

- IPMI -

Addenda, Errata, and Clarifications

Intelligent Platform Management Interface
Second Generation Specification
v2.0, revision 1.0

Intelligent Platform Management Interface
Specification
v1.5, revision 1.1

Addendum Document Revision 2

5/5/05

Copyright © 2002, 2003, 2004, 2005 Intel Corporation, Hewlett-Packard Company, NEC Corporation, Dell Inc., All rights reserved.

INTELLECTUAL PROPERTY DISCLAIMER

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER INCLUDING ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION, OR SAMPLE.

NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED OR INTENDED HEREBY.

INTEL, HEWLETT-PACKARD, NEC, AND DELL DISCLAIM ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF PROPRIETARY RIGHTS, RELATING TO IMPLEMENTATION OF INFORMATION IN THIS SPECIFICATION. INTEL, HEWLETT-PACKARD, NEC, AND DELL, DO NOT WARRANT OR REPRESENT THAT SUCH IMPLEMENTATION(S) WILL NOT INFRINGE SUCH RIGHTS.

I²C is a trademark of Philips Semiconductors. All other product names are trademarks, registered trademarks, or servicemarks of their respective owners.

I²C is a two-wire communications bus/protocol developed by Philips. IPMB is a subset of the I²C bus/protocol and was developed by Intel. Implementations of the I²C bus/protocol or the IPMB bus/protocol may require licenses from various entities, including Philips Electronics N.V. and North American Philips Corporation.

Intel, Hewlett-Packard, NEC, and Dell retain the right to make changes to this document at any time, without notice. Intel, Hewlett-Packard, NEC, and Dell make no warranty for the use of this document and assumes no responsibility for any error which may appear in the document nor does it make a commitment to update the information contained herein.

Contents

Introduction.....	5
Errata Numbers.....	5
6/1/04 Addenda, Errata, and Clarifications	6
E329 Errata - Table 13-8, RMCP/RMCP+ Packet Format for IPMI via Ethernet	6
E330 Clarification - Table 42-3, Sensor Type Codes	6
E331 Clarification/Typo - Table 22 18, Cipher Suite Record Format.....	6
E332 Errata - Table 44-11, Command Number Assignments and Privilege Levels.....	7
E333 Addendum - Table 43 15, Sensor Unit Type Codes.....	7
E334 Typo - Table 44-11, Command Number Assignments and Privilege Levels	8
E335 Errata and Clarification - Table 3-1, Required BMC Functions	8
E336 Errata - Table 44-11, Command Number Assignments and Privilege Levels.....	8
E337 Clarification - Section 32.3, OEM SEL Record - Type E0h-FFh.....	9
E338 Clarification - Section 22.10, Get BT Interface Capabilities Command - applies to IPMI v1.5 and v2.0	9
E339 Addendum - Table 25-4, Serial/Modem Configuration Parameters	10
E340 Clarification - Figure 9-8, Aborting KCS Transactions in-progress and/or Retrieving KCS Error Status.....	10
E341 Addendum - Table 30-9, Alert Immediate Command, and Table H-1, Sub-function Number Assignments	12
E342 Addendum - Table 42-3, Sensor Type Codes	13
E344 Errata - Table 36-3, Sensor Type Codes, Missing Errata E256	13
E345 Errata - Table 17-1, PEF Action Priorities.....	14
E346 Clarification - Section 13.31.4, Integrity Algorithms, Table 22-30, Set Channel Security Keys....	14
E347 Addendum and Clarification - Table 22-17, Get Channel Cipher Suites Command	15
E348 Typos.....	16
E349 Errata - Table 22-12, Get System Interface Capabilities Command	16
E350 Errata and Clarification - Table 13-21, xRC4-Encrypted Payload Fields.....	16
E351 Errata - Table C3-1, Service Processor Management Interface Description Table Format (applies to IPMI v1.5 & v2.0)	17
E352 Addendum - Table 43-13, Entity ID Codes	17
E353 Addendum - (applies to IPMI v1.5 only).....	18
E354 Clarification - Table 43-13, Entity ID Codes	18
E355 Clarification - Table 22-30, Set Channel Security Keys Command.....	18
E356 Errata - Section 6.12.8, Session Sequence Numbers	19
E357 Addendum and Clarification - Table 36-3, Sensor Type Codes	21
E358 Addendum and Clarification - Table 30-9, Alert Immediate Command.....	22
E359 Addendum - Section 17.7, Event Filter Table and Table 30-2, Get PEF Capabilities Command ..	23
E360 Addendum - Section 21, Firmware Firewall & Command Discovery Commands.....	26
E361 Typos and Clarifications - Section 13.28, Authentication, Integrity, and Confidentiality Algorithm Numbers	31
E362 Typos and Clarifications - Section 13.31, RMCP+ Authenticated Key-Exchange Protocol (RAKP)31	31
E366 Errata - Table 22-35, Set User Password Command.....	34

5/5/05 Addenda, Errata, and Clarifications 35

- E363 Typo, Table 5-4, System Software IDs 35
- E364 Clarification - Table 22-17, Get Channel Cipher Suites Command 35
- E365 Typos - Tables 21-7, -8, -9 35
- E367 Addendum - Table 24-2, Activate Payload Command, Table 15-2, SOL Payload Data Format 36
- E368 Typos - Section 21, Firmware Firewall & Command Discovery Commands 38
- E370 Clarification - Table 32-1, SEL Event Records 40
- E372 Addendum - “settable sensors” 41
- E373 Addendum - Table 42-3, Sensor Type Codes 45
- E374 Addendum - Table 28-3, Get Chassis Status Command 46
- E375 Addendum - Table 42-3, Sensor Type Codes 46
- E376 Addendum - Table 42-3, Sensor Type Codes 46
- E377 Clarification - Section 20.1, Get Device ID Command, & Table 20-2, Get Device ID Command ... 47
- E378 Addendum - Set Serial Routing Mux command 48
- E379 Addendum - Command Forwarding 49
- E380 Addendum - Table 25-4, Serial/Modem Configuration Parameters 56
- E381 Addendum - Set / Get System Info Command 57
- E382 Clarification - Table 21-2, Get NetFn Support Command 61
- E383 Clarification - Table 23-4, LAN Configuration Parameters 62
- E384 Clarification - Table 27-3, Set Watchdog Timer Command, Table 27-4, Get Watchdog Timer Command 63
- E385 Clarification/Errata - Section 13.28.4, Integrity Algorithms 64
- E386 Clarification - Table 22-26, Get AuthCode Command 64

Introduction

This document presents errata and clarifications applying to the *Intelligent Platform Management Interface Specification Second Generation Specification, v2.0*, revision 1.0, and *Intelligent Platform Management Interface Specification v1.5*, revision 1.1. For the IPMI v1.5 Specification, this errata document picks up where the *IPMI v1.5 Addenda, Errata, and Clarifications* document, revision 5 document left off.

As of this writing the IPMI specifications are available from the IPMI Web Site at:

<http://www.intel.com/design/servers/ipmi>

The section, table, and figure references are given relative to the given revision of the specification, unless otherwise noted. Where examples are given, text additions are shown with double underlines, and text deletions are shown with strike-through.

Unless noted, errata apply to IPMI v2.0 only.

Errata Numbers

The errata numbers pick up from where numbers for previous errata documents left off. This is done to help avoid confusion when referring to errata across revisions of the specification and errata documents. Some errata numbers are skipped in this document. This is intentional. The errata numbers are derived from numbers used for tracking errata and clarification requests within the IPMI Promoters group. The gaps in the sequence result from requests that have been dropped or that are still in progress.

6/1/04 Addenda, Errata, and Clarifications

E329 Errata - Table 13-8, RMCP/RMCP+ Packet Format for IPMI via Ethernet

The lengths of the Pad Length and Next Header fields were missing in the in RMCP+ column of the table. This has been corrected as follows:

Table 13-8, RMCP/RMCP+ Packet Format for IPMI via Ethernet

Field	Format			Value
	RMCP / IPMI 1.5 26Fh	ASF RMCP 298h	RMCP / IPMI 2.0 "RMCP+" 26Fh	
...				
Pad Length		1	1	indicates how many pad bytes were added so that the amount of non-pad data can be determined.
Next Header		1	1	Reserved in IPMI v2.0. Set to always = 07h for RMCP+ packets defined in this specification.
...				

E330 Clarification - Table 42-3, Sensor Type Codes

The naming for Offset 05h in the Physical Security sensor as "Unauthorized dock/undock" was ambiguous regarding what state would be reported. This has been clarified as follows:

Table 42-3, Sensor Type Codes

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
Physical Security (Chassis Intrusion)	05h	00h	General Chassis Intrusion
		01h	Drive Bay intrusion
		02h	I/O Card area intrusion
		03h	Processor area intrusion
		04h	LAN Leash Lost (system is unplugged from LAN) <i>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.</i>
		05h	Unauthorized dock
		06h	FAN area intrusion (supports detection of hot plug fan tampering)

E331 Clarification/Typo - Table 22 18, Cipher Suite Record Format

The table has been updated to correct typos and clarify that the first field of the Cipher Suite Record starts with either a C0h or C1h byte, followed by one or more ID bytes depending on whether the Cipher Suite is a standard or OEM Cipher Suite, respectively.

Table 22-18, Cipher Suite Record Format

size	Tag bits [7:6]	Tag bits [5:0]
2 or 5	-	<p>This field starts off with either a C0h or C1h "Start of Record" byte, depending on whether the Cipher Suite is a standard Cipher Suite ID or an OEM Cipher Suite, respectively</p> <p>Byte 1: [7:0] = 1100_0000b. Start of Record, Standard Cipher Suite Data following C0h (1100_0000b) start of record byte: Byte 2 - Cipher Suite ID This value is used a numeric way of identifying the Cipher Suite on the platform. It's used in commands and configuration parameters that enable and disable Cipher Suites. See <i>Table 22-19, Cipher Suite IDs</i>.</p> <p>[5:0] = 1100_0001b. Start or Record, OEM Cipher Suite Data following C1h (1100_0001) start of record byte: Byte 2 - OEM Cipher Suite ID See <i>Table 22-19, Cipher Suite IDs</i>.</p> <p>Byte 3:5 - OEM IANA Least significant byte first. 3-byte IANA for the OEM or body that defined the Cipher Suite.</p>

E332 Errata - Table 44-11, Command Number Assignments and Privilege Levels

Incorrect privilege level definition for Activate/Deactivate Payload commands.

Table 44-11, Command Number Assignments and Privilege Levels

	section	NetFn	CMD	C	U	O	A
	...						
Activate Payload	24.1	App	48h		[10]	[10]	[10]
Deactivate Payload	24.2	App	49h		[10]	[10]	[10]

10. The configuration parameters for a given payload type determine the privilege level required to activate / deactivate the payload.

E333 Addendum - Table 43 15, Sensor Unit Type Codes

The IPMI Units table was missing 'grams' as one of the measurement units. In addition, Fatal Errors (which are a special class of uncorrectable error in some bus implementations) were not included. These units have been added to the table as follows:

Table 43-15, Sensor Unit Type Codes

23	minute	57	becquerel	91	fatal error
24	hour	58	PPM (parts/million)	92	grams

E334 Typo - Table 44-11, Command Number Assignments and Privilege Levels

Table 44-11. Command Number Assignments and Privilege Levels had a bad cross-reference. It states that "Set User Access" is in section 22.25, but it isn't. Section 22.25 is "Set Channel Security Keys Command". Section 22.26 is the "Set User Access Command".

E335 Errata and Clarification - Table 3-1, Required BMC Functions

There was a 'cut and paste' error in the specification where text from the SDR Repository requirements was copied to the SEL Interface requirements. This has been corrected by deleting the erroneous text as follows. In addition, since the SEL is critical to post-mortem failure analysis, it is required to be accessible whenever the BMC is accessible. This requirement is also included in the update to table 3-1.

Table 3-1, Required BMC Functions

...

SEL Interface	M	The BMC must provide a System Event Log interface. The event log must hold at least 16 entries. SEL access must be provided via the system interface. The SEL must be fully accessible via all mandatory SEL commands through all supported interfaces to the BMC whenever the system is powered up or in ACPI 'S1' sleep state. SEL read access is always mandatory whenever the BMC is accessible, and through any interface that is operational, regardless of system power state. .-
---------------	---	--

...

E336 Errata - Table 44-11, Command Number Assignments and Privilege Levels

The specification was missing the command number assignments for Firmware Firewall commands. These are defined as follows:

Table 44-11, Command Number Assignments and Privilege Levels

	section	NetFn	CMD	C	U	O	A
IPM Device "Global" Commands							
...							
Get NetFn Support	21.2	App	09h		X		
Get Command Support	21.3	App	0Ah		X		
Get Command Sub-function Support	21.4	App	0Bh		X		
Get Configurable Commands	21.5	App	0Ch		X		
Get Configurable Command Sub-functions	21.6	App	0Dh		X		
unassigned	-	App	0Eh-0Fh	-	-	-	-
Set Command Enables	21.7	App	60h				X
Get Command Enables	21.8	App	61h		X		
Set Command Sub-function Enables	21.9	App	62h				X
Get Command Sub-function Enables	21.10	App	63h		X		

...

E337 Clarification - Section 32.3, OEM SEL Record - Type E0h-FFh

An OEM ID is not part of the record. As with other OEM commands and operations, the OEM ID for the record is inferred from the *Get Device ID* command. Since the record has no mechanism for returning which controller or software logged the record, the ID must be presumed to be the MFR ID from the *Get Device ID* command to the BMC.

32.3 OEM SEL Record - Type E0h-FFh

E0h - FFh Range reserved for non-timestamped OEM SEL records. The SEL Device does not automatically timestamp these records. The four bytes passed in the byte locations normally used for the timestamp will be directly entered into the SEL. *SEL viewer applications should not interpret byte positions 4:7 in this record as a timestamp.* These records are entered via the *Add SEL or Partial Add SEL* commands.

Note that an OEM ID is not part of this record. Since the record also has no mechanism for returning which controller or software logged the record, the OEM ID for this record is presumed to be the MFR ID from the *Get Device ID* command to the BMC.

E338 Clarification - Section 22.10, Get BT Interface Capabilities Command - applies to IPMI v1.5 and v2.0

This applies to both IPMI v1.5 and v2.0. (For IPMI v1.5 this is for section 18.9.) The size definitions for bytes 3 and 4 was ambiguous / misleading. The names implied that the value was returning the size of the buffer (i.e. how many bytes a driver could write to the interface) when instead the value was returning the maximum 'message payload' size that could be accepted. This is clarified as follows:

22.10 Get BT Interface Capabilities Command

The BT interface includes a *Get BT Interface Capabilities* command that returns various characteristics of the interface, including buffer sizes, and multithreaded communications capabilities.

Table 22-13, Get BT Interface Capabilities Command

	byte	data field
Request Data	-	-
Response Data	1	Completion Code
	2	Number of outstanding requests supported (1 based. 0 illegal)
	3	Input (request) buffer message size in bytes. (1 based.) ^[1]
	4	Output (response) buffer message size in bytes. (1 based.) ^[1]
	5	BMC Request-to-Response time, in seconds, 1 based. 30 seconds, maximum.
	6	Recommended retries (1 based). (see text for BT Interface)

1. For Bytes 3 and 4 (Input and Output Buffer size), the buffer message size is the largest value allowed in first byte (length field) of any BT request or response message. For a send, this means if *Get BT Interface Capabilities* returns 255 in byte 3 (input buffer size) the driver can actually write 256 bytes to the input buffer (adding one for the length byte (byte 1) that is sent in with the request.)

E339 Addendum - Table 25-4, Serial/Modem Configuration Parameters

A new, optional, parameter is defined to help speed software detection of bit rate support for the channel. This parameter is specified as follows:

Table 25-4, Serial/Modem Configuration Parameters

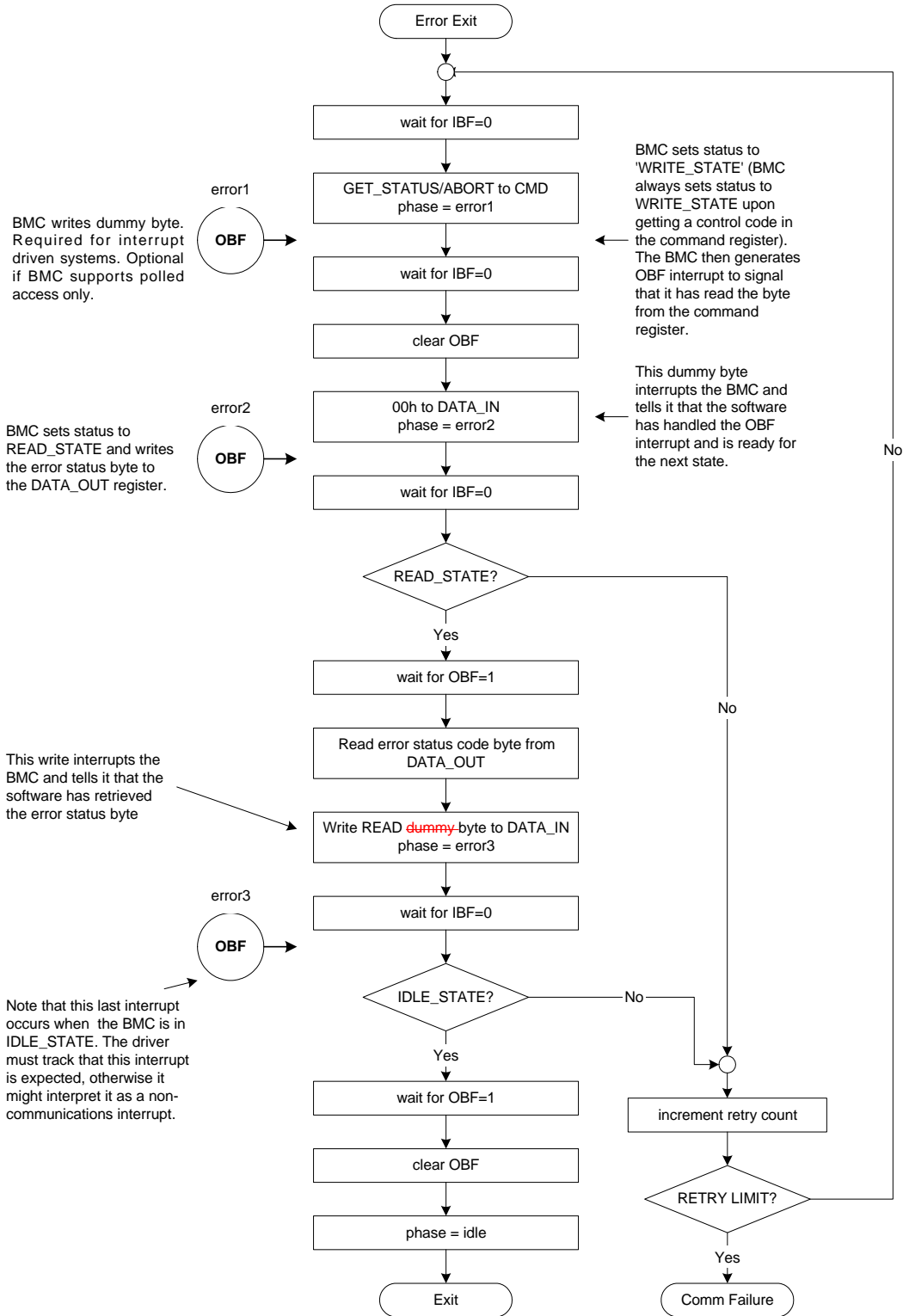
Parameter	#	Parameter Data (non-volatile unless otherwise noted) ⁽¹⁾
...		
Bit Rate Support (READ ONLY, optional)	50	This parameter returns a read-only bitfield indicating which bit rates are supported for this serial channel. <u>data 1</u> - Bit Rate Support [7:6] - reserved [4] - 115.2 kbps [3] - 57.6 kbps [2] - 38.4 kbps [1] - 19.2 kbps (required) [0] - 9600 bps

...

E340 Clarification - Figure 9-8, Aborting KCS Transactions in-progress and/or Retrieving KCS Error Status

The block labelled "Write READ dummy byte to DATA_IN, phase=error3" can be confusing. The READ control code value (68h) must be written, not just any 'dummy' data byte. This is clarified as follows:

Figure 9-8, Aborting KCS Transactions in-progress and/or Retrieving KCS Error Status



E341 Addendum - Table 30-9, Alert Immediate Command, and Table H-1, Sub-function Number Assignments

The *Alert Immediate* command works well for testing alerts, but it cannot effectively be used for generating events that contain event data. The command is extended to allow event data to be delivered with the alert, and table H-1 is updated to allow reporting of this optional capability as a sub-function.

Table 30-9, Alert Immediate Command

Byte data field

	...
	<p>3 Alert String Selector Selects which Alert String, if any, to use with the alert. [7] - 0b = don't send an Alert String 1b = send Alert String identified by following string selector. [6:0] - string selector. 000_0000b = use volatile Alert String. 01h-7Fh = non-volatile string selector.</p>
	<p><i>The following "Platform Event Parameters" (bytes 4:11) can be used to fill in the corresponding event data fields of a Platform Event Trap. When supported, all bytes (4:11) must be supplied. Implementation of this capability is OPTIONAL but highly recommended for IPMI v2.0 implementations. See Table 29-5, Event Request Message Fields, for specification of the individual fields.</i></p>
	4 Generator ID
	5 EvMRev
	6 Sensor Type
	7 Sensor #
	8 Event Dir Event Type
	9 Event Data 1
	10 Event Data 2
	11 Event Data 3
Response Data	<p>1 Completion Code. Generic codes, plus following command-specific completion codes: 81h = Alert Immediate rejected due to alert already in progress. 82h = Alert Immediate rejected due to IPMI messaging session active on this channel. 83h = Platform Event Parameters (4:11) not supported.</p>
	...

Table H-1, Sub-function Number Assignments

	Sub Fn #	NetFn	CMD
	...		
Alert Immediate		S/E	16h
reserved / unspecified	0		
Alert to Channel 1	1		
Alert to Channel 2	2		
Alert to Channel 3	3		

Alert to Channel 4	4		
Alert to Channel 5	5		
Alert to Channel 6	6		
Alert to Channel 7	7		
Platform Event Parameters	8		

...

E342 Addendum - Table 42-3, Sensor Type Codes

The specification primarily covers events for failures related to OS Hangs and startup, but did not cover events for 'normal' OS shutdown or if a power down or reset occurs that is not related to a BMC or pushbutton initiated power down or resets. The specification also did not cover whether or not the soft-shutdown was initiated by PEF, or if a shutdown that was requested via a local s/w agent failed. This is addressed with the following additions:

Table 42-3, Sensor Type Codes

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
...			
System Boot / Restart Initiated	1Dh	00h	Initiated by power up (this would typically be generated by BIOS/EFI)
		01h	Initiated by hard reset (this would typically be generated by BIOS/EFI)
		02h	Initiated by warm reset (this would typically be generated by BIOS/EFI)
		03h	User requested PXE boot
		04h	Automatic boot to diagnostic
		05h	OS / run-time software initiated hard reset
		06h	OS / run-time software initiated warm reset
		07h	System Restart (Intended to be used with Event Data 2 and or 3 as follows): <u>Event Data 2</u> [7:4] - reserved [3:0] - restart cause per <i>Get System Restart Cause</i> command. <u>Event Data 3</u> Channel number used to deliver command that generated restart, per <i>Get System Restart Cause</i> command.
...			
OS Stop / Shutdown	20h	00h	Critical stop during OS load / initialization. Unexpected error during system startup. Stopped waiting for input or power cycle/reset.
		01h	Run-time Critical Stop (a.k.a. 'core dump', 'blue screen')
		02h	OS Graceful Stop (system powered up, but normal OS operation has shut down and system is awaiting reset pushbutton, power-cycle or other external input)
		03h	OS Graceful Shutdown (system graceful power down by OS)
		04h	Soft Shutdown initiated by PEF
05h	Agent Not Responding. Graceful shutdown request to agent via BMC did not occur due to missing or malfunctioning local agent.		

E344 Errata - Table 36-3, Sensor Type Codes, Missing Errata E256

IPMI v1.5 Errata revision 5, E256 - the offset for "Addendum - Timestamp Synch Event" was missed in v2.0 rev 1.0 of the specification. (The note associated with the offset was already part in the spec, however.) This is corrected as follows. The note associated with the offset was already part in the spec.

Table 36-3, Sensor Type Codes

System Event	12h	00h ... 05h	... Timestamp Clock Synch. This event can be used to record when changes are made to the timestamp clock(s) so that relative time differences between SEL entries can be determined. See note ⁽¹⁾ . <u>Event Data 2</u> [7] - first/second 0b = event is first of pair. 1b = event is second of pair. [6:4] - reserved [3:0] - Timestamp Clock Type 0h = SEL Timestamp Clock updated. (Also used when both SEL and SDR Timestamp clocks are linked together.) 1h = SDR Timestamp Clock updated.
--------------	-----	-------------------	--

E345 Errata - Table 17-1, PEF Action Priorities

The priority for ICMB Group Control Operation as a PEF action was not listed in PEF Priority Table. This is corrected as follows:

Table 17-1, PEF Action Priorities

Action	Priority	Additional Information
power down	1	(optional)
power cycle	2	(optional) Will not be executed if a power down action was also selected.
reset	3	(mandatory) Will not be executed if a power down or power cycle action was also selected.
Diagnostic Interrupt	4	(optional) The diagnostic interrupt will not occur if a higher priority action is also selected to occur.
ICMB Group Control	5	(optional) Performs ICMB group control operation according to settings from the Group Control Table parameter in the PEF Configuration Parameters.
Send Alert	6	(mandatory if alerting is supported) Send alerts in order based on the selected Alert Policy. Alert actions will be deferred until after the power down has completed. There is an additional prioritization within alerts being sent: based on the Alert Policy Table entries for the alert. This is described further in <i>Section 17.11, Alert Policy Table</i> .
OEM	OEM	(optional) Priority determined by OEM.

E346 Clarification - Section 13.31.4, Integrity Algorithms, Table 22-30, Set Channel Security Keys

The size of Kg (160 bits) was not explicitly defined, nor were the sizes of Kg and Kr parameters called out in the *Set Channel Security Keys* command. In addition, a recommendation has been added to indicate that a 160-bit user key should be used when "one-key" logins are used. This has been clarified with additions to section 13.31.4 and 22.25 as follows:

13.31.4 Integrity Algorithms

The Integrity Algorithm Number specifies the algorithm used to generate the contents for the AuthCode “signature” field that accompanies authenticated IPMI v2.0/RMCP+ messages once the session has been established.

Unless otherwise specified, the integrity algorithm is applied to the packet data starting with the AuthType/Format field up to and including the field that immediately precedes the AuthCode field itself.

HMAC-SHA1-96 and HMAC-MD5-128 utilize the Session Integrity Key as the key for use in HMAC. For “two key” logins, 160-bit key KG is used in the creation of SIK. For “one key” logins, the user’s key (password) is used in place of KG. To maintain a comparable level of authentication, it is recommended that a full 160-bit user key be used when “one key” logins are enabled for IPMI v2.0/RMCP+.

Table 22-30, Set Channel Security Keys Command

byte	data field
...	
3	Key ID [7:0] - key ID. 00h = RMCP+ “KR” key (20 bytes). The “KR” key is used as a unique value for random number generation. Note: A BMC implementation is allowed to share a single KR value across all channels. A utility can set KR and lock it for one channel, and then verify it has been set and locked for any other channels by using this command to read the key from other channels and checking the ‘lock status’ field to see if it matches and is unlocked. 01h = RMCP+ “KG” key (20 bytes). “KG” key acts as a value that is used for key exchange for the overall channel. This key is not lockable in order to enable a password/key configuration utility to set its value. This value is used in conjunction with the user key values (passwords) in RAKP-HMAC-SHA1 and RAKP-HMAC-MD5 authentication. I.e. the remote console needs to have a-priori knowledge of both this key value and the user password setting, in order to establish a session. all other = reserved
...	

E347 Addendum and Clarification - Table 22-17, Get Channel Cipher Suites Command

The specification did not indicate the format of data returned when the “list supported algorithms” parameter was selected. This is corrected as follows:

Table 22-17, Get Channel Cipher Suites Command

byte	data field
...	
3	List Index. [7] - 1b = list algorithms by Cipher Suite 0b = list supported algorithms ¹⁾ [6] - reserved [5:0] - List index (00h-3Fh). 0h selects the first set of 16, 1h selects the next set of 16, and so on. 00h = Get first set of algorithm numbers. The BMC returns sixteen (16) bytes at a time per index, starting from index 00h, until the list data is exhausted, at which point it will 0 bytes or <16 bytes of list data.
...	

1. When listing numbers for supported algorithms, the BMC returns a list of the algorithm numbers for each algorithm that the BMC supports on a given channel. Each algorithm is

only listed once. There is no requirement that the BMC return the algorithm numbers in any specific order.

E348 Typos

- Cross references to “Table 37-12, Entity ID Codes” should be “Table 43-14, Entity ID Codes”
- Cross references to “Table 37-11, IPMB/I 2C Device Type Codes” should be “Table 43- 12, IPMB/I 2C Device Type Codes”
- Formatting error: A bad cross-reference made it appear as if the last sentence of the first paragraph of section 12.12, Discovering SSIF, was truncated after “(see”. The corrected formatting is:
[...the existence and slave address of the SSIF \(see Appendix C1 - Locating IPMI System Interfaces via SM BIOS Tables\).](#)
- Bad cross-reference formatting in section 13.32.2, Encryption with AES
- Table 13-8, RMCP/RMCP+ Packet Format for IPMI via Ethernet didn't consistently list field sizes in all columns for parts of Ethernet packet and IP Header that are common across IPMI v1.5/RMCP, ASF/RMCP, and IPMI v2.0/RMCP+
- Formatting: IPMI Session Header in tables 13-9 and 13-10 were not shaded.
- Table 13-11, ‘Maximum Requested Privilege Level’ ala IPMI v1.5. → ‘Maximum Requested Privilege Level’ as in IPMI v1.5.
- *Set Security Keys* → *Set Channel Security Keys*
- Table 44-11, Command Number Assignments and Privilege Levels. Bad cross reference for “Get BT Interface Capabilities” 22.9 → 22.10
- Appendix Table Captions. A number of the caption numbers for appendix tables were incorrect. E.g. *Table C3- 44-8, ...* instead of *Table C3-1, ...* The captions have been fixed.
- Corrected section references for *Set User Access* and *Set Channel Security Keys* commands in Table 44-11, Command Number Assignments and Privilege Levels.

E349 Errata - Table 22-12, Get System Interface Capabilities Command

The table did not list the parameters returned for SMIC. This is corrected as follows:

Table 22-12, Get System Interface Capabilities Command

byte data field

...

For System Interface Type = KCS or SMIC	
3	[7:3] - reserved [2:0] - System Interface Version 000b = version 1 (conformant with KCS or SMIC interface as defined in this specification).

...

E350 Errata and Clarification - Table 13-21, xRC4-Encrypted Payload Fields

The last sentence of the description for the “Data offset” field was truncated. In addition, it wasn't emphasized that in xRC4 encrypted payloads the Confidentiality Header is not encrypted, just the Payload Data. This is corrected as follows:

Table 13-21, xRC4-Encrypted Payload Fields

Field	Size	Sub field	Description
Confidentiality Header (not encrypted)	4	Data offset	This value advances 'N' counts for every N-bytes of new payload data that is encrypted. The value for the first packet of payload data is 0000_0000h. If the first packet contains 12 bytes of payload data, the data offset for the second packet will be 12 (0Ch). If the second packet contained 8 bytes of payload data, the offset for the third packet will be 20 (14h), and so on. The xRC4 algorithm operates in a manner similar to a large pseudo-random number generator. Therefore, decryption can handle missed packets by advancing the state machine by the number of steps to the offset for the data and decrypt from that point.
	16	Initialization Vector	The Initialization Vector is a 128-bit random number that is used in conjunction key information for the session to initialize the state machine for xRC4. The Initialization Vector is only passed when the xRC4 state machine is initialized or is reinitialized (data offset = 0000_0000h). This field is absent when the data offset is non-zero.
Payload Data	variable	Payload Data	Payload data. Encrypted per xRC4 algorithm.
Confidentiality Trailer	0	none	xRC4 does not add use a confidentiality trailer.

E351 Errata - Table C3-1, Service Processor Management Interface Description Table Format (applies to IPMI v1.5 & v2.0)

Byte offsets 36 and 37 were swapped. In addition, there should be a reserved byte 64. This is corrected as follows:

Table C3-1, Service Processor Management Interface Description Table Format

Field	Byte Length	Byte Offset	Description
			...
Interface Type	1	36	Indicates the type of IPMI interface: 0 Reserved 1 Keyboard Controller Style (KCS) 2 Server Management Interface Chip (SMIC) 3 Block Transfer (BT) 4 SMBus System Interface (SSIF) 5-255 Reserved
Reserved	1	37	This field must always be 01h to be compatible with any software that implements previous versions of this spec.
			...
Reserved	1	64	This field must always be null (0x00) to be compatible with any software that implements previous versions of this spec. This field is a deprecated "SPMI ID Field". Implementations based on pre-IPMI v2.0 versions of SPMI may contain a null-terminated string here.

E352 Addendum - Table 43-13, Entity ID Codes

A new Entity ID for Real Time Clock has been added.

Table 43-13, Entity ID Codes

Code	Entity
53	35h Real Time Clock (RTC)

...

...

E353 Addendum - (applies to IPMI v1.5 only)

The following addendum enables the SSIF to be used with IPMI v1.5 systems while retaining IPMI v1.5 specification conformance.

1.6.16 System Interfaces

...The IPMI system interfaces are:

Keyboard Controller Style (KCS) The bit definitions, and operation of the registers follows that used in the Intel 8742 Universal Peripheral Interface microcontroller. The term ‘Keyboard Controller Style’ reflects the fact that the 8742 interface was used as the legacy keyboard controller interface in PC architecture computer systems. ...

...

SMBus System Interface (SSIF) The SMBus System Interface is defined as part of the IPMI v2.0 or later specifications. However, this interface can be used with IPMI v1.5 implementations. IPMI v1.5 implementations that use SSIF will be considered to be conformant to the IPMI v1.5 specification if the SSIF implementation meets the IPMI v2.0 or later specifications for the interface. Use of the SSIF with an IPMI v1.5 implementation requires using the IPMI v2.0 or later specification and complying with any licensing requirements for implementing those specifications.

E354 Clarification - Table 43-13, Entity ID Codes

Entity ID for BIOS is renamed “System Firmware” to indicate it’s applicable to legacy BIOS and new technologies such as EFI .

Table 43-13, Entity ID Codes

Code	Entity
34	22h System Firmware (e.g. BIOS / EFI)

...

...

E355 Clarification - Table 22-30, Set Channel Security Keys Command

The text indicated that software could check to see whether the locking of KR on one channel would lock it for all channels. The last sentence stated that after locking it on one channel, software could check the lock status on other channels to see if it was ‘unlocked’. It is clearer and more robust to indicate that software should verify to see whether it is the status returns “locked” on the other channels. Additional text changes are made to explain what happens if the command is executed for a channel that does not

support RMCP+, why KG cannot be locked, and that KG must be settable on a per-channel basis. These changes are made to the table as follows:

Table 22-30, Set Channel Security Keys Command

	byte	data field
Request Data	1	Channel Number [7:4] - reserved [3:0] - Channel Number (Note: this command only applies to channels that support RMCP+, if the channel does not support RMCP+ the command will return an error completion code.)
	...	
	3	Key ID [7:0] - key ID. 00h = RMCP+ "KR" key (20 bytes). The "KR" key is used as a unique value for random number generation. Note: A BMC implementation is allowed to share a single KR value across all channels. A utility can set KR and lock it for one channel, and then verify it has been set and locked for any other channels by using this command to read the key from other channels and checking the 'lock status' field for each channel to see if it matches and is locked. 01h = RMCP+ "KG" key (20 bytes). "KG" key acts as a value that is used for key exchange for the overall channel. This key cannot be locked. This is to ensure a password/key configuration utility can set its value. This value is used in conjunction with the user key values (passwords) in RAKP-HMAC-SHA1 and RAKP-HMAC-MD5 authentication. I.e. the remote console needs to have a-priori knowledge of both this key value and the user password setting, in order to establish a session. KG must be individually settable on each channel that supports RMCP+. all other = reserved
	...	

E356 Errata - Section 6.12.8, Session Sequence Numbers

When the v2.0 document was created, text and edits on sequence number handing that were in the draft were accidentally left out. This is corrected as follows:

6.12.8 Session Sequence Numbers

The session sequence number is a 32-bit, unsigned, value. The session sequence number is *not* used for matching IPMI requests and responses. The IPMI Sequence (Seq) field or similar field in the particular payload is used for that purpose. The sender of the packet increments the session sequence number for every packet that gets transmitted even if the payload of the content is a 'retry'. Session Sequence Numbers are generated and tracked on a per-session basis. I.e. there are separate sets of sequence numbers and sequence number handling for each session.

Sequence numbers only apply to packets that are transmitted within the context of an IPMI session. Certain IPMI commands and protocol messages are accepted 'outside of a session'. When sent outside a session, the sequence number fields for these packets are always set to 0000_0000h.

6.12. 9 IPMI v1.5 Session Sequence Number Handling

For IPMI v1.5 sessions, there are two Session Sequence Numbers: the *Inbound* Session Sequence Number and the *Outbound* Session Sequence Numbers. The inbound and outbound directions are defined with respect to the BMC. Inbound messages are from the remote console to the BMC, while outbound messages are from the BMC to the remote console.

Inbound messages use the inbound session sequence number, while outbound messages use the outbound session sequence number. The inbound and outbound sequence numbers are updated and tracked independently, and are

unique to each session. Since the number of incoming packets and outgoing packets will typically vary, the inbound and outbound sequence numbers will not stay in lock step with one another.

The BMC and the remote console independently select the starting session sequence number for the messages they receive. Typically, this is done using a random number in order to further reduce the likelihood of a playback attack. The remote console sets the starting values for the outbound session sequence number when it sends the first *Activate Session* command for an authenticated session. The remote console must increment the inbound session sequence number by one (1) for each subsequent message it sends to the BMC. The *Activate Session* response is the first authenticated outbound (BMC to remote console) message. This response message uses the initial outbound session sequence number value that the remote console delivered in the prior *Activate Session* command request. The BMC must increment the outbound session sequence number by one (1) for each subsequent outbound message from the BMC.

16.12.10 IPMI v1.5 Inbound Session Sequence Number Tracking and Handling

-At a minimum, the BMC is required to track that the inbound sequence number is increasing, and to silently discard the packet if the sequence number is **eight** counts or more from than the last value received. (An implementation is allowed to contain a proprietary configuration option that enables a larger sequence number difference, as long as the standard of +eight can be restored.)

An implementation can elect to terminate the session if it receives a number of sequence numbers that are more than eight counts from the last value received.

Valid packets (packets with good data integrity checks and signature) to a given session that have the same inbound sequence number as an earlier packet are considered to be duplicate packets and are silently discarded (even if the message content is different).

16.12.11 IPMI v1.5 Out-of-order Packet Handling

In order to avoid closing a session because a packet was received out-of-order, the BMC must implement one of two options:

Option 1: Advancing eight-count (or greater) window. Recommended. Track which packets have been received that have sequence numbers up to eight counts *less* than the highest last received sequence number, tracking which of the prior eight sequence numbers have been received. Also accept packet with sequence numbers that are up to eight counts greater than the last received sequence number, and set that number as the new value for the highest sequence number received. This option is illustrated in *Appendix A - Previous Sequence Number Tracking*.

Option 2: Drop any packets with sequence numbers that are lower than the last valid value received. While simpler than option 1, this option is not recommended except for resource-constrained implementations due to the fact that any out-of-order packets will require the remote console to timeout and retransmit.

Sequence number wrap-around must be taken into account for both options. When a sequence number advances from FFFF_FFFFh to 0000_0000h, the value FFFF_FFFFh represents the lesser sequence number.

16.12.12 IPMI v1.5 Outbound Session Sequence Number Tracking and Handling

The remote console is required to handle outbound session sequence number tracking in the same manner as the BMC handles the inbound session sequence number, except that Option 2 (above) should not be used as a means of handling out-of-order packets.

16.12.13 IPMI v2.0 RMCP+ Session Sequence Number Handling

For IPMI v2.0 RMCP+ sessions, there are two *sets* of Session Sequence Numbers for a given session. One set of inbound and outbound sequence numbers is used for authenticated (signed) packets, and the other set is used for unauthenticated packets. The inbound and outbound sequence numbers for authenticated packets are updated and tracked independently from the inbound and outbound sequence numbers for unauthenticated packets.

IPMI v2.0 RMCP+ Session Sequence Numbers are used for rejecting packets that may have been duplicated by the network or intentionally replayed.

The individual Session Sequence Numbers is are initialized to zero whenever a session is created and incremented by one at the start of outbound processing for a given packet (i.e. the first transmitted packet has a '1' as the sequence number, not 0). Session Sequence numbers are incremented for every packet that is transmitted by a given sender, regardless of whether the payload for the packet is a 'retry' or not.

When dropping packets because of sequence number, any packet with an illegal, duplicate, or out-of-range sequence can be dropped without having to verify the packet integrity data (AuthCode) signature first. When accepting packets, the BMC must apply any packet integrity and authentication code checks before accepting the packet's sequence number.

16.12.14 IPMI v2.0 RMCP+ Sliding Window

IPMI v2.0 RMCP+ uses a 'sliding window' for tracking sequence numbers for received packets. This sliding window is used for rejecting packets that have sequence numbers that are significantly out-of-range with respect to the sequence number for the most recently accepted packet while allowing a number of out-of-order packets to be accepted.

In order for a packet to be accepted by the BMC, its sequence number must fall within a 32-count sliding window, where packets will be accepted if they are within plus 15 or minus 16 counts of the highest sequence number that was previously accepted, and they are not duplicates of any previously received sequence numbers.

E357 Addendum and Clarification - Table 36-3, Sensor Type Codes

The specification did not provide the ability to report sensor or FRU device failures. This is added to Table 36-3 as follows. Also, clarifications were added to indicate the difference between degraded, unavailable, off-line, and failure states:

Table 36-3, Sensor Type Codes

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
			...
Management Subsystem Health	28h	00h	sensor access degraded or unavailable (A sensor that is degraded will still return valid results, but may be operating with a slower response time, or may not detect certain possible states. A sensor that is unavailable is not able to return any results (scanning is disabled,))
		01h	controller access degraded or unavailable (The ability to access the controller has been degraded, or access is unavailable, but the party that is doing the monitoring cannot determine which.)
		02h	management controller off-line (controller cannot be accessed for normal operation because it has been intentionally taken off-line for a non-error condition. Note that any commands that are available must function according to specification.)
		03h	management controller unavailable (controller cannot be accessed because of an error condition)
		04h	Sensor failure (the sensor is known to be in error. It may still be accessible by software) <u>Event Data 2</u> The Event Data 2 field for this offset can be used to provide additional information on the type of failure with the following definition: [7:0] - Sensor Number. Number of the failed sensor corresponding to event offset 04h or 00h.

		05h	<p>FRU failure</p> <p>The Event Data 2 and 3 fields for this offset can be used to provide additional information on the type of failure with the following definition:</p> <p><u>Event Data 2</u></p> <p>[7] - logical/physical FRU device 0b = device is not a logical FRU Device 1b = device is logical FRU Device (accessed via FRU commands to mgmt. controller)</p> <p>[6:5] - reserved. [4:3] - LUN for Master Write-Read command or FRU Command. 00b if device is non-intelligent device directly on IPMB. [2:0] - Private bus ID if bus = Private. 000b if device directly on IPMB, or device is a logical FRU Device.</p> <p><u>Event Data 3</u></p> <p>For LOGICAL FRU DEVICE (accessed via FRU commands to mgmt. controller):</p> <p>[7:0] - FRU Device ID within controller that generated the event. FFh = reserved.</p> <p>For non-intelligent FRU device:</p> <p>[7:1] - 7-bit I2C Slave Address of FRU device . This is relative to the bus the device is on. For devices on the IPMB, this is the slave address of the device on the IPMB. For devices on a private bus, this is the slave address of the device on the private bus.</p> <p>[0] - reserved.</p>
--	--	-----	---

E358 Addendum and Clarification - Table 30-9, Alert Immediate Command

The *Alert Immediate* command did not provide a mechanism for testing an alert for particular event data. This meant that fields of the PET trap could not be filled in using this command. The following OPTIONAL parameters are added to the command. In addition, the sub-function assignments are extended to enable reporting the presence or absence of this optional capability via the command discovery commands. An error completion code on the *Alert Immediate* command can also report whether this capability is supported or not.

Table 30-9, Alert Immediate Command

Request Data	Byte	data field
	1	Channel number. (This value is required to select which configuration parameters are to be used to send the Alert.) [7:4] - reserved [3:0] - Channel number. Note: BMC stores the 'Alert immediate status' for each channel that can send alert.
...		
<i>The following "Platform Event Parameters" (bytes 4:11) can be used to fill in the corresponding event data fields of a Platform Event Trap. When supported, all bytes (4:11) must be supplied. Implementation of this capability is OPTIONAL but highly recommended for IPMI v2.0 implementations. See Table 29-5, Event Request Message Fields, for specification of the individual fields.</i>		
	4	Generator ID
	5	EvMRev
	6	Sensor Type
	7	Sensor #

Response Data	8	Event Dir Event Type
	9	Event Data 1
	10	Event Data 2
	11	Event Data 3
	1	Completion Code. Generic codes, plus following command-specific completion codes: 81h = Alert Immediate rejected due to alert already in progress. 82h = Alert Immediate rejected due to IPMI messaging session active on this channel. 83h = Platform Event Parameters (4:11) not supported.

...

Table H-1, Sub-function Number Assignments

	Sub Fn #	NetFn	CMD
...			
Alert Immediate reserved / unspecified	0	S/E	16h
Alert to Channel 1	1		
Alert to Channel 2	2		
Alert to Channel 3	3		
Alert to Channel 4	4		
Alert to Channel 5	5		
Alert to Channel 6	6		
Alert to Channel 7	7		
Platform Event Parameters	8		

...

E359 Addendum - Section 17.7, Event Filter Table and Table 30-2, Get PEF Capabilities Command

The following additions are made to Section 17.7 and Table 17-2 to enable the OPTION for an implementation to support PEF filtering on OEM Events. Correspondingly, the *Get PEF Capabilities* command is updated to report whether OEM Event Record Filtering is supported.

Table 30-2, Get PEF Capabilities Command

	byte	data field
Request Data	-	-
Response Data	1	Completion Code
	2	PEF Version (BCD encoded, LSN first, 51h for this specification. 51h → version 1.5)
	3	[7] - 1b = OEM Event Record Filtering supported [6] - reserved Action Support [5] - 1b = diagnostic interrupt [4] - 1b = OEM action [3] - 1b = power cycle [2] - 1b = reset [1] - 1b = power down [0] - 1b = Alert
	4	Number of event filter table entries (1 based)

17.7 Event Filter Table

The Event Filter Table consists of a set of rows or ‘entries’ that define each filter. The following table specifies the fields that comprise a row in the Event Filter Table....

There are two things that can kick off PEF: the arrival of a new event or BMC startup with pending events.

PEF filters for event data that corresponds to the Type 02h System Event Record format. IPMI v2.0 introduces an OPTION to enable an implementation to also filter Type C0h-DFh, and Type E0h-FFh OEM Event Records. When this option is available, the filter table can be used to filter on the OEM Record Type value and the first six non-timestamp OEM data bytes as exact matches. The next three bytes in the OEM Event Record can be filtered with mask-based comparisons the function in the same manner as the matching for the Event Data 1, 2, and 3 fields of a Type 02h System Event Record.

If filtering of OEM Event Records is supported, the *Add SEL Entry* command can be used for adding OEM Events to the SEL.

Table 17-2, Event Filter Table Entry

Byte	Field	Description
1	Filter Configuration	[7] - 1b = enable filter 0b = disable filter [6:5] - 11b = reserved 10b = manufacturer pre-configured filter. The filter entry has been configured by the system integrator and should not be altered by software. Software is allowed to enable or disable the filter, however. 01b = reserved 00b = software configurable filter. The filter entry is available for configuration by system management software. [4:0] - Record type 0h = Record 02h 1h = OEM Record C0h-DFh (timestamped. Includes Mfr ID as first three non-timestamp data bytes in record.) 2h = OEM Record E0h-FFh (non-timestamped. Mfr ID from <i>Get Device ID</i> command for BMC.)
...		

5	Generator ID Byte 1 / OEM Record Type	Slave Address or Software ID from Event Message, or OEM Event Record Type ^[1] . FFh = match any
6	Generator ID Byte 2 / OEM data 1	Channel Number / LUN to match, or first non-timestamp OEM record data byte following the Record Type byte. (I.e. For E0h-FFh records, OEM data 1 corresponds to byte 4, for C0h-DFh records OEM data 1 corresponds to byte 7) FFh = match any see section 32, <i>SEL Record Formats</i> .
7	Sensor Type / OEM data 2	Type of sensor, or 2nd OEM record data byte following the timestamp. FFh = match any
8	Sensor # / OEM data 3	FFh = match any
9	Event Trigger (Event/Reading Type) / OEM data 4	FFh = match any
10, 11	Event Data 1 Event Offset Mask / OEM data 5:6	This bit field is used to match different values of the least significant nibble of the Event Data 1 field. This enables a filter to provide a match on multiple event offset values. Bit positions 15 through 0 correspond to the offset values Fh - 0h, respectively. A 1 in a given bit position will cause a match if the value in bits 3:0 of the Event Data 1 hold the corresponding value for the bit position. Multiple mask bits can be set to 1, enabling a match to multiple values. A match must be made with this field in order to have a match for the filter. <u>data 1</u> 7:0 - mask bit positions 7 to 0, respectively. <u>data 2</u> 15:8 - mask bit positions 15 to 8, respectively. For OEM record matching: <u>data 1</u> = OEM data 5 (FFh = match any) <u>data 2</u> = OEM data 6 (FFh = match any)
12	Event Data 1 AND Mask / OEM Data 7 AND Mask	This value is applied to the entire Event Data 1 byte. The field is Used to indicate 'wildcarded' or 'compared' bits. This field must be used in conjunction with Compare 2. To match any Event Data field value, just set the corresponding AND Mask, Compare 1, and Compare 2 fields to 00h. (See <i>Section 17.8, Event Data 1 Event Offset Mask</i> for more information). Note that the Event Data 1 AND mask, Compare 1 mask, and Compare 2 masks will typically be set to wild-card the least significant of Event Data 1 in order to allow the Event Data 1 Event Mask field to determine matches to the event offset. Bits 7:0 all have the following definition: 0 = Wildcard bit. (drops this bit position in the Event Data byte out of the comparison process) Corresponding bit position must be a 1 in Compare 1, and a 0 in Compare 2. (Note - setting a 0 in this bit, a 1 and Compare 1 and a 1 in Compare 2 guarantees that you'll <i>never</i> have a match.) 1 = use bit for further 'exact' or 'non-exact' comparisons based on Compare 1 and Compare 2 values.
13	Event Data 1 Compare 1 / OEM Event Data 7 Compare 1	Used to indicate whether each bit position's comparison is an exact comparison or not. (See <i>Section 17.8, Event Data 1 Event Offset Mask</i> for more information). Here, 'test value' refers to the Event Data value <i>after</i> the AND Mask has been applied. Bits 7:0 all have the following definition: 1 = match bit in test value exactly to correspond bit position in Compare 2 0 = contributes to match if corresponding bit in test value matches corresponding bit in Compare 2.

14	Event Data 1 Compare 2 / OEM Event Data 7 Compare 2	(See <i>Section 17.8, Event Data 1 Event Offset Mask</i> for more information). Here, 'test value' refers to the Event Data value <i>after</i> the AND Mask has been applied. Bits 7:0 all have the following definition: 1 = match a '1' in corresponding bit position in test value. 0 = match a '0' in corresponding bit position in test value.
15	Event Data 2 AND Mask / OEM Event Data 8 AND Mask	
16	Event Data 2 Compare 1 / OEM Event Data 8 Compare 1	
17	Event Data 2 Compare 2 / OEM Event Data 8 Compare 2	
18	Event Data 3 AND Mask / OEM Event Data 9 AND Mask	
19	Event Data 3 Compare 1 / OEM Event Data 9 Compare 1	
20	Event Data 3 Compare 2 / OEM Event Data 9 Compare 2	

E360 Addendum - Section 21, Firmware Firewall & Command Discovery Commands

The "Group Extension" and "OEM/Group" Network Function codes utilize a byte or IANA that identifies the party that has defined command functionality under the given code. The Firmware Firewall commands did not provide a mechanism to discover these codes, nor return support for commands defined under those codes. This is corrected by defining new, optional, commands and optional parameters for existing commands to enable getting and setting the command support for the codes under these network functions. A new command number assignment is also made and the new command is also added to Table H-1, Sub-function Number Assignments.

21.2b Get OEM NetFn IANA Support Command

This command returns the IANA Enterprise Number that is used to identify the OEM or Group that has defined functionality under Network Function codes 2Ch/2Dh, or 2Eh/2Fh. The command can be iterated if there is more than one IANA associated with the given Network Function code.

Table 21-2, Get OEM NetFn IANA Support Command

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-7h, Fh = channel numbers Eh = retrieve information for channel this request was issued on.
	2	Network Function (NetFn) code [7:6] - reserved. [5:0] - Network Function to get OEM IANA info for. Legal values are: 2Ch = "Group Extension" Network Function (codes 2Ch,2Dh) 2Eh = "OEM/Group" Network Function (codes 2Eh, 2Dh) all other = reserved
	3	List Index [7:6] - reserved [5:0] - List Index. 0 gets first IANA. Increment until last IANA is returned
IPMI Response Data	1	Completion Code
	2	[7] - 1b = last IANA [6:0] - reserved
	3	LUN support [7:6] - LUN 3 (11b) support 00b = no commands supported on LUN 3 (11b) 01b = commands follow base IPMI specification. Commands exist on LUN, but no special restriction of command functions. Commands follow standard Optional/Mandatory specifications. 10b = commands exist on LUN, but some commands/operations may be restricted by firewall configuration. 11b = reserved [5:4] - LUN 2 (10b) support <i>Note that a BMC uses LUN 10b for message bridging. The message bridging capability is enabled/disabled by enabling/disabling the Send Message command.</i> 00b = no commands supported on LUN 2 (10b) 01b = commands follow base IPMI specification. Commands exist on LUN, but no special restriction of command functions. Commands follow standard Optional/Mandatory specifications. 10b = commands exist on LUN, but some commands/operations may be restricted by firewall configuration. 11b = reserved [3:2] - LUN 1 (01b) support [1:0] - LUN 0 (00b) support
	<i>For Network Function = 2Ch:</i>	
	(4)	Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)
	<i>For Network Function = 2Eh:</i>	
(4:6)	OEM or group IANA supported for given Network Function code on returned LUNs. LS byte first. (See description for Network Function 2Eh/2Fh in Table 5-1, Network Function Codes)	

Table 21-3, Get Command Support Command

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-7h, Fh = channel numbers Eh = retrieve information for channel this request was issued on.
	...	
	2	[7:6] - Operation 00b = return support mask for commands 00h through 7Fh. 01b = return support mask for commands 80h through FFh. 10b, 11b = reserved. [5:0] - NetFn. Network function code to look up command support for. The management controller will return the same values for odd or even NetFn values. I.e. the value for bit [0] is ignored.
	3	[7:2] - reserved [1:0] - LUN
<i>For Network Function = 2Ch:</i>		
	(4)	Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)
<i>For Network Function = 2Eh:</i>		
	(4:6)	OEM or group IANA supported for given Network Function code on returned LUNs. LS byte first. (See description for Network Function 2Eh/2Fh in Table 5-1, Network Function Codes)
IPMI Response Data	1	Completion Code
...		

Table 21-4, Get Command Sub-function Support Command

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-7h, Fh = channel numbers Eh = retrieve information for channel this request was issued on.	
	...		
	<i>For Network Function = 2Ch:</i>		
		5	Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)
<i>For Network Function = 2Eh:</i>			
	6:8	OEM or group IANA supported for given Network Function code on returned LUNs. LS byte first. (See description for Network Function 2Eh/2Fh in Table 5-1, Network Function Codes)	
IPMI Response Data	1	Completion Code	
...			

Table 21-5, Get Configurable Commands Command

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-7h, Fh = channel numbers Eh = retrieve information for channel this request was issued on.
	...	
<i>For Network Function = 2Ch:</i>		
	(4)	Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)
<i>For Network Function = 2Eh:</i>		
	(5:7)	OEM or group IANA supported for given Network Function code on returned LUNs. LS byte first. (See description for Network Function 2Eh/2Fh in Table 5-1, Network Function Codes)
IPMI Response Data	1	Completion Code
...		

Table 21-6, Get Configurable Command Sub-functions Command

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-7h, Fh = channel numbers Eh = retrieve information for channel this request was issued on.
	...	
<i>For Network Function = 2Ch:</i>		
	(5)	Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)
<i>For Network Function = 2Eh:</i>		
	(6:8)	OEM or group IANA supported for given Network Function code on returned LUNs. LS byte first. (See description for Network Function 2Eh/2Fh in Table 5-1, Network Function Codes)
IPMI Response Data	1	Completion Code
...		

Table 21-7, Set Command Enables Command

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-7h, Fh = channel numbers Eh = retrieve information for channel this request was issued on.
	...	
<i>For Network Function = 2Ch:</i>		
	(19)	Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)
<i>For Network Function = 2Eh:</i>		
	(19:21)	OEM or group IANA supported for given Network Function code on returned LUNs. LS byte first. (See description for Network Function 2Eh/2Fh in Table 5-1, Network Function Codes)
IPMI Response Data	1	Completion Code Generic, plus following command-specific codes: 80h = attempt to enable an unsupported or un-configurable command.

Table 21-7, Set Command Enables Command

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-7h, Fh = channel numbers Eh = retrieve information for channel this request was issued on.
	...	
<i>For Network Function = 2Ch:</i>		
	(19)	Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)
<i>For Network Function = 2Eh:</i>		
	(19:21)	OEM or group IANA supported for given Network Function code on returned LUNs. LS byte first. (See description for Network Function 2Eh/2Fh in Table 5-1, Network Function Codes)
IPMI Response Data	1	Completion Code Generic, plus following command-specific codes: 80h = attempt to enable an unsupported or un-configurable command.

Table 21-1, Firmware Firewall Commands

Command	Section Defined	O/M
Get NetFn Support	21.2	O ^[1,3]
Get Command Support	21.3	O ^[1,3]
Get Command Sub-function Support	21.4	O ^[1,3]
Get Configurable Commands	21.5	O ^[2]
Get Configurable Command Sub-functions	21.6	O ^[2]
Set Command Enables	21.7	O
Get Command Enables	21.8	O ^[2]
Set Command Sub-function Enables	21.9	O ^[2]
Get Command Sub-function Enables	21.10	O ^[2]
Get OEM NetFn IANA Support	21.11	O ^[1,3,4]

1. Mandatory on any channel/interface to the BMC on which a typically mandatory command can be or is disabled for firmware firewall purposes.
2. Mandatory on any channel/interface to the BMC on which the *Set Command Enables* command is implemented. The *Set Command Enables*, *Get Command Enables*, *Set Command Sub-function Enables*, and *Get Command Sub-function Enables* commands must be implemented as a set.
3. The *Get NetFn Support*, *Get Command Support*, and *Get Command Sub-function Support* commands must be implemented as a set.
4. Mandatory if OEM network functions 2Ch-2Dh or 2Eh-2Fh are utilized on management controller and firmware firewall is implemented.

Table H-1, Sub-function Number Assignments

	Sub Fn #	NetFn	CMD
IPM Device "Global" Commands			
reserved		App	00h
Get Device ID		App	01h
...			
Set Command Enables		App	60h
Get Command Enables		App	61h
Set Command Sub-function Enables		App	62h
Get Command Sub-function Enables		App	63h
Get OEM NetFn IANA Support		App	64h
...			

Table G-1, Command Number Assignments and Privilege Levels

	section	NetFn	CMD	C	U	O	A
IPM Device “Global” Commands							
reserved	-	App	00h	-	-	-	-
...							
Get OEM NetFn IANA Support	21.11	App	64h		X		
...							

E361 Typos and Clarifications - Section 13.28, Authentication, Integrity, and Confidentiality Algorithm Numbers

The text referred to RAKP Message 3 and 4 when it should have referred to 2 and 3, respectively. In addition, clarification were added to indicate the size of the SIK and additional keying material if the RAKP-HMAC-MD5 Algorithm is used, and to clarify the size of the Message Authentication Code fields. This is corrected as follows:

13.28.1 RAKP-HMAC-SHA1 Authentication Algorithm

RAKP-HMAC-SHA1 specifies the use of RAKP messages for the key exchange portion of establishing the session, and that HMAC-SHA1 (per [RFC2101]) is used to create 20-byte Key Exchange -Authentication Code fields in RAKP Message 2 and RAKP Message 3. HMAC-SHA1-96 (per [RFC2404]) is used for generating a 12-byte Integrity Check Value field for RAKP Message 4.

...

13.28.3 RAKP-HMAC-MD5 Authentication Algorithm

This authentication algorithm operates the same way as RAKP-HMAC-SHA1 except that HMAC with MD5 (per [RFC2104]) is used for RAKP authentication operations in place of SHA-1. Thus, the Key Exchange Authentication Code fields in RAKP Message 2 and RAKP Message 3 and the Integrity Check Value field in RAKP Message 4 are all 16-byte fields (128-bit MD5). Since MD5 requires fewer computational steps than SHA-1, this option can be used to offer a quicker session activation, particularly on management controllers that have limited computational resources. When the SIK and additional keying material (K1, K2, etc.) are generated (per sections 13.31, RMCP+ Authenticated Key-Exchange Protocol (RAKP), and 13.32, Generating Additional Keying Material) the MD5 algorithm is used in the HMAC algorithm, resulting in 16-byte (128-bit) keys.

E362 Typos and Clarifications - Section 13.31, RMCP+ Authenticated Key-Exchange Protocol (RAKP)

The section was written only describing the use of the RAKP-HMAC-SHA1 authentication algorithm. Since other algorithms are available, the section has been generalized and now references RAKP-HMAC-SHA1 as an example authentication algorithm that uses RAKP. In addition, the Message Authentication Code fields and the Integrity Check Value field in the RAKP messages were not called out as the fields that would hold certain calculated values for RAKP using in RMCP+. Lastly, there were typos where RFC2404 was called out instead of RFC2104.

This is corrected as follows:

13.31 RMCP+ Authenticated Key-Exchange Protocol (RAKP)

RMCP+ can support a number of different authentication and key exchange protocols during its *Creation* (session activation) phase. For this specification, the mandatory-to-implement authentication and key exchange protocol is the RMCP+ Authenticated Key-Exchange Protocol (RAKP). RAKP (defined below) was developed based on the Authenticated Key Exchange Protocol (AKEP) defined by Bellare and Rogaway in [BR1].

RAKP-- uses pre-shared symmetric keys -to mutually authenticate a remote console to a given managed system and to generate pair-wise unique symmetric keying material that can be used with a number of integrity and confidentiality algorithms to provide protection for RMCP messages. The use of RAKP with the different authentication and integrity algorithms available for IPMI v2.0/RMCP+ is described in 13.28, *Authentication, Integrity, and Confidentiality Algorithm Numbers*. For example, the RAKP-HMAC-SHA1 authentication algorithm uses the HMAC-SHA1 integrity algorithm defined in [RFC2104] in the RAKP authentication process, and the HMAC-SHA1-96 integrity algorithm defined in [RFC2404] for data integrity.

RAKP also supports the concept of remote console user “roles” and optionally “usernames” (e.g. operator “x” or administrator “y”), which are established by RAKP when a session is created.

...

Once this and other necessary RMCP-related data is installed in the managed system and the managed system is initialized, the remote console can initiate sessions with the managed system. Following the exchange of RMCP Presence Ping/Pong and RMCP+ Open Session Request/Response messages (exchanging Session IDs and selecting RAKP for use), the remote console starts the RAKP protocol. First, the remote console selects a random number, **R_M**, a requested role, **Role_M**, a user name length, **ULength_M**, a user name (optional - denoted by < > below), **UName_M**, and the managed system’s Session ID, **SID_C**, and sends them to the managed system as Message 1.

Message 1: Remote Console -> Managed System

SID_C, R_M, Role_M, ULength_M, < UName_M >

After receiving Message 1, the managed system verifies that the value **SID_C** is active and that a session can be created using **Role_M**, **ULength_M**, and (optional), **UName_M** for the given selections for security algorithms.

If the request is valid, the managed system then selects a random number, **R_C**, and sends to the remote console as Message 2 the values **SID_M**, **R_C**, and **GUID_C** as well as the HMAC per [RFC2104] of the values (**SID_M**, **SID_C**, **R_M**, **R_C**, **GUID_C**, **Role_M**, **ULength_M**, < **UName_M** >) generated using key **K_[UID]** associated with the given username, **UName_M**, and role, **Role_M**.

Message 2: Managed System -> Remote Console

**SID_M, R_C, GUID_C,
HMAC_{K[UID]}(SID_M, SID_C, R_M, R_C, GUID_C, Role_M, ULength_M, < UName_M >)**

Where:

Parameter	bytes	Name
SID _M	4	Remote_Console_Session_ID
SID _C	4	Managed_System_Session_ID
R _M	16	Remote Console Random_Number
R _C	16	Managed System Random Number
GUID _C	16	Managed_System_GUID
Role _M	1	Requested Privilege Level (Role) (this is the <i>entire</i> byte holding the Requested Privilege Level field)
ULength _M	1	User Name Length byte (number of bytes of UName _M = 0 for 'null' username)
UName _M	var	User Name bytes (absent for 'null' username)

Where $\text{HMAC}_{\text{K}[\text{UID}]}(\text{SID}_M, \text{SID}_C, \text{R}_M, \text{R}_C, \text{GUID}_C, \text{Role}_M, \text{ULength}_M, \langle \text{UName}_M \rangle)$ represents the value for the Key Exchange Authentication Code field in RAKP Message 2. (The $\text{HMAC}_{\text{K}[\text{UID}]}$ notation indicates use of the HMAC algorithm per [RFC2104] with the hashing function (e.g. SHA-1, MD5) that is specified for the selected authentication algorithm (See 13.28, *Authentication, Integrity, and Confidentiality Algorithm Numbers*) over the concatenation of the indicated fields where $\text{K}[\text{UID}]$ is the user-specific key that is associated with the given username and role. Note that some authentication algorithms may substitute a different algorithm than HMAC for generating the Key Exchange Authentication Code.)

After receiving RAKP Message 2, the remote console verifies that the value SID_M is active and that GUID_C matches the managed system that the remote console is expecting to communicate with. The remote console then validates the Key Exchange Authentication Code from the message. If the code is valid, the remote console creates the Session Integrity Key (SIK) by generating an HMAC per [RFC2104] of the concatenation of R_M , R_C , Role_M , ULength_M , and (optional) UName_M using 160-bit key K_G (note - no truncation).

The hashing algorithm used for this HMAC, and the ones following, is specified by the particular authentication algorithm being used. (Note that $\text{K}[\text{UID}]$ is used in place of K_G if 'one-key' logins are being used. See 13.28.4, *Integrity Algorithms*)

$$\text{SIK} = \text{HMAC}_{\text{K}_G}(\text{R}_M \mid \text{R}_C \mid \text{Role}_M \mid \text{ULength}_M \mid \langle \text{UName}_M \rangle)$$

Then the remote console sends to the managed system as Message 3 the value SID_C and (for the RAKP-HMAC-SHA1 algorithm) the HMAC per [RFC2104] of the values (R_C , SID_M , Role_M , ULength_M , $\langle \text{UName}_M \rangle$) generated using key $\text{K}[\text{UID}]$ selected by the username, UName_M , and role Role_M .

Message 3: Remote Console -► Managed System

$$\text{SID}_C, \text{HMAC}_{\text{K}[\text{UID}]}(\text{R}_C, \text{SID}_M, \text{Role}_M, \text{ULength}_M, \langle \text{UName}_M \rangle)$$

Where $\text{HMAC}_{\text{K}[\text{UID}]}(\text{R}_C, \text{SID}_M, \text{Role}_M, \text{ULength}_M, \langle \text{UName}_M \rangle)$ represents the value for the Key Exchange Authentication Code for RAKP Message 3. After receiving Message 3, the managed system verifies that the value SID_C is active and then validates the message authentication code. If the HMAC is valid, the managed system creates the SIK by generating an HMAC per [RFC2104] of the concatenation of R_M , R_C , Role_M , ULength_M , and (optional) UName_M using 160-bit key K_G (note - no truncation, and that $\text{K}[\text{UID}]$ is used in place of K_G if 'one-key' logins are being used. See 13.28.4, *Integrity Algorithms*).

$$\text{SIK} = \text{HMAC}_{\text{K}_G}(\text{R}_M \mid \text{R}_C \mid \text{Role}_M \mid \text{ULength}_M \mid \langle \text{UName}_M \rangle)$$

The managed system then sends to the management console as Message 4 the values SID_M , and (for the RAKP-HMAC-SHA1 algorithm) the HMAC per [RFC2404] of the values (R_M , SID_C , GUID_C) generated using key SIK .

Message 4: Managed System -► Mgmt Console

$$\text{SID}_M, \text{HMAC}_{\text{SIK}}(\text{R}_M, \text{SID}_C, \text{GUID}_C)$$

Where $\text{HMAC}_{\text{K}[\text{UID}]}(\text{R}_C, \text{SID}_M, \text{Role}_M, \text{ULength}_M, \langle \text{UName}_M \rangle)$ represents the value in the Integrity Check Value field for RAKP Message 4. After receiving Message 4, the management console verifies that the value SID_M is active and then validates the Integrity Check Value. If the value is valid, the management console has verification that mutual authentication with the managed system was successful and that the same pair-wise unique SIK was successfully generated on both ends of the connection. The management console then transitions into the *Message Transfer* session state (the session is now active and, if authentication or authentication/encryption have been enabled, the transfer of authenticated and authenticated/encrypted payloads can commence).

The same RAKP steps are followed for session activation even if the Cipher Suite indicates that there are no integrity or encryption algorithms required for the session.

E366 Errata - Table 22-35, Set User Password Command

A cut-and-paste error caused the definition for bit 7 that was in the pre-release Adopter review document to be missed in the release specification. This bit explicitly indicates whether the password is to be saved as 16- or 20-bytes. This is corrected as follows:

Table 22-35, Set User Password Command

Request Data	byte	data field
	1	User ID. For IPMI v2.0, the BMC shall support 20-byte passwords (keys) for all supported user IDs that have configurable passwords. The BMC shall maintain an internal tag that indicates whether the password was set as a 16-byte or as a 20-byte password. ... The 'test password' operation can be used to determine whether a password has been stored as 16-bytes or 20-bytes. - [7] - password size 1b = set 20-byte user password/key. 0b = set 16-byte user password/key (IPMI v1.5 backward compatible) [6] - reserved. [5:0] - User ID. 000000b = reserved. (User ID 1 is permanently associated with User 1, the null user name).

...

5/5/05 Addenda, Errata, and Clarifications

E363 Typo, Table 5-4, System Software IDs

There was a typo in the "Resultant 8-bit value" for the System Management Software row in the table. The last value should be 5Fh instead of 6Fh. This is corrected as follows:

Table 5-4, System Software IDs

System Software Type	IDs (7-bit)	bit 0 ¹	Resultant 8-bit value ¹
...			
System Management Software	20h-2Fh	1b	41h, 43h, 45h, ... 5Fh
...			

E364 Clarification - Table 22-17, Get Channel Cipher Suites Command

This clarification further specifies the response format when bit7 byte3 request is 0 (list supported algorithms). The BMC just returns the algorithm numbers consecutively with no required ordering. Note 1 is updated as follows:

1. When listing numbers for supported algorithms, the BMC returns a list of the algorithm numbers for each algorithm that the BMC supports on a given channel. Each algorithm is listed consecutively and only listed once. There is no requirement that the BMC return the algorithm numbers in any specific order.

E365 Typos - Tables 21-7, -8, -9

The offsets in the tables were in error. The values are corrected as follows:

Table 21-7, Set Command Enables Command

...	
4:19	Enable/Disable Mask
...	
<i>For Network Function = 2Ch:</i>	
(20)	Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)
<i>For Network Function = 2Eh:</i>	
(20:22)	OEM or group IANA supported for given Network Function code on returned LUNs. LS byte first. (See description for Network Function 2Eh/2Fh in Table 5-1, Network Function Codes)
...	

Table 21-8, Get Command Enables Command

IPMI Response Data	1	Completion Code
	2:17	Enable/Disable Mask
	...	
	<i>For Network Function = 2Ch:</i>	
	(18)	Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)
<i>For Network Function = 2Eh:</i>		
(18:20)	OEM or group IANA supported for given Network Function code on returned LUNs. LS byte first. (See description for Network Function 2Eh/2Fh in Table 5-1, Network Function Codes)	

Table 21-10, Get Configurable Command Sub-function Enables Command

IPMI Response Data	1	Completion Code Generic, plus following command-specific completion codes: 80h = attempt to enable an unsupported or un-configurable sub-function.
	2:5	Sub-Function Enables (ls-byte first) These sixteen bits form a bitfield where each bit indicates support for a particular sub-function for the given command. The bit offset corresponds to the number of the sub-function. See <i>Table H-1, Sub-function Number Assignments</i>

E367 Addendum - Table 24-2, Activate Payload Command, Table 15-2, SOL Payload Data Format

This addendum adds an optional SOL ‘test mode’ to IPMI v2.0. This mode enables remote console software to monitor the state of RTS and DTR, and to directly control the DCD and DSR state for implementations that can support manipulation and monitoring of these signals. This can be used for supporting hardware compatibility tests for the 16550 interface that run on the managed system.

To support this, the following changes are made to the *Activate Payload* command and SOL payload format:

Table 24-2, Activate Payload Command

3:6	<p>Auxiliary Request Data. Additional payload-specific parameters to configure behavior of the payload when it becomes activated. Ignored if no auxiliary data is specified for given payload type.</p> <p>For Payload Type = SOL: byte 1 [7] - Encryption Activation ... [5] - Test Mode (optional). Enables DCD/ and DSR to be manually controlled by the remote console and the reporting of RTS and DTR state via the SOL Operation/Status byte. This can be used to facilitate software testing of the 16550 UART interface.</p>
-----	---

Response Data		<p>1b = activate test mode. If test mode is not supported, bit [0] of the auxiliary response data will be returned as 0b. 0b = deactivate test mode [4] - reserved [3:2] - Shared Serial Alert Behavior ... byte 2:4 reserved - write a 00h</p>
	1	Completion Code ...
	2:5	<p>Auxiliary Response Data. LS-byte first. For Payload = SOL: [31:1] - reserved. [0] - 0b = test mode not supported / enabled 1b = test mode enabled</p>

...

Table 15-2, SOL Payload Data Format

Field	Size	Description	
		...	
Operation / Status	1	<p>BMC to Remote Console: Operations are executed <i>before</i> character data is transferred. [7] reserved [6] 1b: Packet is being NACK'd. The BMC is unable to accept all character data from packet. Note: Operation field is still accepted even if packet is NACK'd. 0b: ACK. BMC ready to accept next packet of character data. [5]^[1] A NACK packet with this status will be automatically sent one time after this bit changes state. (Whenever the system enters or leaves a power state where character transfers to the system serial controller are possible) A NACK packet with "Character transfer is unavailable" status will also be sent for each character transfer request from the remote console when the system is in a powered-down or sleep state. 1b: Character transfer is unavailable because system is in a powered-down or sleep state. 0b: SOL character transfer is available. [4]^[2] A NACK packet with this status will be automatically sent one sent once, just before the BMC deactivates SOL because of a front panel power-button or a reset. 1b: SOL is deactivated/deactivating.</p>	<p>Remote Console to BMC: Note: Operations are executed <i>before</i> character data is transferred. [7] reserved [6] ACK/NACK 1b: NACK. Packet is being NACK'd by the remote console. 0b: ACK. Packet is being ACK'd by the remote console. [5] Ring/WOR Assert RI (may not be supported on all implementations) - Goal is to allow this to be used for generating a WOR. [4] Break 1b: Generate BREAK (300 ms, nominal) [3] CTS 1b: Deassert CTS (clear to send) to the baseboard serial controller. (This is the default state when SOL is deactivated.) 0b: If test mode = inactive, Let BMC control CTS. If test mode = active, assert CTS. [2] DCD/DSR for test mode = inactive: 1b: Deassert DCD/DSR to baseboard serial controller 0b: Assert DCD/DSR to baseboard serial controller. for test mode = active: 1b: Deassert DCD to baseboard serial controller 0b: Assert DCD to baseboard serial controller.</p>

	<p>[Remote console can use this to tell if SOL was deactivated by some other party, or by local pushbutton reset or power on/off].</p> <p>0b: SOL is active.</p> <p>[3] Transmit Overrun</p> <p>1b: characters were dropped between transmitting this packet and the previous packet, because the system did not pay attention to hardware flow control.</p> <p>0b: no characters were lost between this packet and the preceding packet.</p> <p>[2] Break</p> <p>1b: A break condition from the system has been detected. The BMC will generate this only on one packet at the start of the break.</p> <p>0b: no break detected</p> <p>[1:0] For test mode = inactive: Reserved</p> <p>For test mode = active: [1] - 1b = RTS asserted [0] - 1b = DTR asserted</p> <p>A packet with this status will be automatically sent whenever RTS or DTR changes state. Note that this packet may not contain character data. If no character data is available, this will be a NACK packet. Otherwise, the ACK/NACK state follows the definition for bit [6], above.</p>	<p>[1] Flush Inbound</p> <p>for test mode = inactive: 1b: Drop (flush) data from remote console to BMC [not including data carried in this packet, if any]</p> <p>for test mode = active: 1b: Deassert DSR to baseboard serial controller 0b: Assert DSR to baseboard serial controller.</p> <p>[0] Flush Outbound</p> <p>for test mode = inactive: 1b: Flush Outbound Character Data (flush data from BMC to remote console)</p> <p>for test mode = active: reserved. Write as 0b.</p>
--	--	---

...

E368 Typos - Section 21, Firmware Firewall & Command Discovery Commands

The following shows corrections for typos that were in command tables for the firmware firewall commands:

Table 21-4, Get Command Sub-function Support Command

...

<i>For Network Function = 2Eh:</i>	
5:7	OEM or group IANA supported for given Network Function code on returned LUNs. LS byte first. (See description for Network Function 2Eh/2Fh in Table 5-1, Network Function Codes)

...

(5:8)	Support Mask 1 (ls-byte first) These thirty-two bits form a bitfield where each bit indicates support for a particular sub-function for the given command. ...
-------	---

Table 21-5, Get Configurable Commands Command

...

2:17	Support Mask ... Depending on the value of the "Operation" parameter passed in the request: ...
------	---

Table 21-6, Get Configurable Command Sub-functions Command

...

2:5	Support Mask (ls-byte first) These thirty-two bits form a bitfield where each bit indicates support for a particular sub-function for the given command. ...
-----	---

Table 21-7, Set Command Enables Command

...

2	[7:6] - Operation. ... [5:0] - NetFn. Network function code to set command support for. The management controller will set the same values for odd or even NetFn values. I.e. the value for bit [0] is ignored.
---	--

...

4:19	Enable/Disable Mask Depending on the value of the "Operation" parameter passed in the request: ...
------	--

Table 21-8, Get Command Enables Command

...

2:17	Enable/Disable Mask These sixteen bytes form a 128-bit bitfield where each bit returns the enable/disable of a particular command value under the given NetFn. If a command is not supported at all, a 0b will be returned. ...
------	---

Table 21-9, Set Configurable Command Sub-function Enables Command

...

2	[7:6] - reserved [5:0] - NetFn. Network function code to set command support for. The management controller will set the same values for odd or even NetFn values. I.e. the value for bit [0] is ignored.
---	--

...

5:8	Sub-Function Enables (ls-byte first). The enable/disable settings are <i>non-volatile</i> and take effect on successful completion of the command. The management controller must reject all new settings (must not change present settings) if there is any error in the command (non-zero completion code returned). These thirty-two bits form a bitfield where each bit indicates support for a particular sub-function for the given command. The bit offset corresponds to the number of the sub-function. ...
-----	--

...

Table 21-10, Get Configurable Command Sub-function Enables Command

...

2:5	Sub-Function Enables (ls-byte first) These thirty-two bits form a bitfield where each bit indicates support for a particular sub-function for the given command. The bit offset corresponds to the number of the sub-function. See <i>Table H-1, Sub-function Number Assignments</i> . 1b sub-function is enabled 0b sub-function is disabled or is un-configurable/reserved. [31] - bit for sub-function 31. [30] - bit for sub-function 30. ... [1] - bit for sub-function 1. [0] - bit for sub-function 0.
-----	---

E370 Clarification - Table 32-1, SEL Event Records

Table 32-1, SEL Event Records doesn't refer to IPMI v2.0 for the EvM rev field. This (Note: The value of EvmRev did not change for IPMI v2.0, since event record formats did not change.)

Table 32-1, SEL Event Records

Byte	Field	Description
...		
1	EvM Rev	Event Message format version (=04h for events in this specification, 03h for IPMI v1.0 Event Messages.) <i>Note: the BMC must accept Platform Event request messages that are in IPMI v1.0 format (EvMRev=03h) and log them as IPMI v1.5 / v2.0 Records by setting the EvMRev field to 04h and setting the Channel Number in the Generator ID field appropriately for the channel that the event was received from.</i>

...

E372 Addendum - “settable sensors”

This addendum adds a new command and affects Table 43-1, Full Sensor Record - SDR Type 01h, Table 43-2, Compact Sensor Record - SDR Type 02h, Table G-1, Command Number Assignments and Privilege Levels, Table H-1, Sub-function Number Assignments, and Table 35-1, Sensor Device Commands, as follows:

35.17 Set Sensor Reading and Event Status Command

This command enables software to set the present reading and event status for sensors that support this command. This can be used to create sensors where the data comes from software, such as a BIOS SMI handler, rather than being directly polled or accessed by BMC hardware. The Type 01h and Type 02h SDRs include an optional bit that allows those records to report a sensor is settable.

Table 35-18, Set Sensor Reading and Event Status Command

Request Data	1	sensor number (FFh = reserved)
	2	<p><u>Operation</u></p> <p>[7:6] - <u>Event Data Bytes operation</u> This field controls whether associated event data bytes are written or left unchanged for the given sensor. These event data bytes will be returned in any event message generated by the sensor.</p> <p>11b = reserved 10b = Write given values to event data bytes, excluding bits [3:0] of Event Data 1. (If values trigger an event, BMC will automatically generate bits [3:0] based on the sensor reading and event status.) 01b = Write given values to event data bytes, including bits [3:0] of Event Data 1 (bits [3:0] written to Event Data 1 will override BMC generation of the event offset value on next event generated by the given sensor.) 00b = Don't use Event Data bytes from this command. BMC will generate it's own Event Data bytes based on its sensor implementation.</p> <p>[5:4] - <u>Assertion bits operation</u> This field controls whether the corresponding assertion event status bits in the given sensor get set cleared according to the assertion event status parameters in this command, or are left unchanged. If the parameter for the assertion bits is absent from this command, the corresponding assertion bits in the sensor (if any) will remain unchanged regardless of the selected operation.</p> <p>11b = A 0b in a given bit position in the given parameter causes corresponding bit position to be cleared. A 1b causes no change to the corresponding 10b = A 1b in a given bit position causes corresponding bit position to be set to 1b. A 0b 01b = write given value to assertion event status bytes 00b = don't change assertion event status bytes</p> <p>[3:2] - <u>Deassertion bits operation</u> This field controls whether the deassertion event status bits in the given sensor get set, cleared according to the deassertion event status parameters in this command, or are left unchanged. If the parameter for the deassertion bits is absent from this command, the corresponding assertion bits in the sensor (if any) will remain unchanged regardless of the selected operation.</p> <p>11b = A 0b in a given bit position in the given parameter causes corresponding bit position to be cleared. A 1b causes no change to the corresponding 10b = A 1b in a given bit position causes corresponding bit position to be set to 1b. A 0b 01b = write given value to assertion event status bytes</p>

	<p>00b = don't change assertion event status bytes</p> <p>[1:0] - <u>Sensor Reading operation</u> This field controls whether the sensor reading byte is written or left unchanged according to the sensor 10b, 11b = reserved 01b = write given value to sensor reading byte 00b = don't change sensor reading byte</p>
3	<p>Sensor Reading <u>Byte 1: byte of reading.</u></p>
(4)*	<p><u>For sensors with threshold based events:</u> (High-going events are asserted when value first becomes \geq threshold. Low-going events are asserted when value first becomes \leq corresponding threshold.)</p> <p>[7] - 1b = assertion event condition for upper non-critical going high occurred [6] - 1b = assertion event condition for upper non-critical going low occurred [5] - 1b = assertion event condition for lower non-recoverable going high occurred [4] - 1b = assertion event condition for lower non-recoverable going low occurred [3] - 1b = assertion event condition for lower critical going high occurred [2] - 1b = assertion event condition for lower critical going low occurred [1] - 1b = assertion event condition for lower non-critical going high occurred [0] - 1b = assertion event condition for lower non-critical going low occurred</p> <p><u>For sensors with discrete events:</u> [7] - 1b = state 7 assertion event occurred [6] - 1b = state 6 assertion event occurred [5] - 1b = state 5 assertion event occurred [4] - 1b = state 4 assertion event occurred [3] - 1b = state 3 assertion event occurred [2] - 1b = state 2 assertion event occurred [1] - 1b = state 1 assertion event occurred [0] - 1b = state 0 assertion event occurred</p>
(5)*	<p><u>For sensors with threshold based events:</u> [7:4] - reserved. Write as 0000b. [3] - 1b = assertion event condition for upper non-recoverable going high occurred [2] - 1b = assertion event condition for upper non-recoverable going low occurred [1] - 1b = assertion event condition for upper critical going high occurred [0] - 1b = assertion event condition for upper critical going low occurred</p> <p><u>For sensors with discrete events:</u> (00h otherwise) [7] - reserved. Ignore on read. [6] - 1b = state 14 assertion event occurred [5] - 1b = state 13 assertion event occurred [4] - 1b = state 12 assertion event occurred [3] - 1b = state 11 assertion event occurred [2] - 1b = state 10 assertion event occurred [1] - 1b = state 9 assertion event occurred [0] - 1b = state 8 assertion event occurred</p>

	<p>(6)* <u>For sensors with threshold based events:</u> (High-going events are deasserted when value goes less than the corresponding threshold minus the positive-going hysteresis value. Low-going events are deasserted when value goes greater than the corresponding threshold plus the negative-going hysteresis value.) [7] - 1b = deassertion event condition for upper non-critical going high occurred [6] - 1b = deassertion event condition for upper non-critical going low occurred [5] - 1b = deassertion event condition for lower non-recoverable going high occurred [4] - 1b = deassertion event condition for lower non-recoverable going low occurred [3] - 1b = deassertion event condition for lower critical going high occurred [2] - 1b = deassertion event condition for lower critical going low occurred [1] - 1b = deassertion event condition for lower non-critical going high occurred [0] - 1b = deassertion event condition for lower non-critical going low occurred</p> <p><u>For sensors with discrete events:</u> [7] - 1b = state 7 deassertion event occurred [6] - 1b = state 6 deassertion event occurred [5] - 1b = state 5 deassertion event occurred [4] - 1b = state 4 deassertion event occurred [3] - 1b = state 3 deassertion event occurred [2] - 1b = state 2 deassertion event occurred [1] - 1b = state 1 deassertion event occurred [0] - 1b = state 0 deassertion event occurred</p>
	<p>(7)* <u>For sensors with threshold based events:</u> [7:4] - reserved. Write as 0000b. [3] - 1b = deassertion event condition for upper non-recoverable going high occurred [2] - 1b = deassertion event condition for upper non-recoverable going low occurred [1] - 1b = deassertion event condition for upper critical going high occurred [0] - 1b = deassertion event condition for upper critical going low occurred</p> <p><u>For sensors with discrete events:</u> (0h otherwise) [7] - reserved. Ignore on read. [6] - 1b = state 14 deassertion event occurred [5] - 1b = state 13 deassertion event occurred [4] - 1b = state 12 deassertion event occurred [3] - 1b = state 11 deassertion event occurred [2] - 1b = state 10 deassertion event occurred [1] - 1b = state 9 deassertion event occurred [0] - 1b = state 8 deassertion event occurred</p>
	<p>(8)* <u>Event Data 1</u> (See <i>Table 29 6, Event Request Message Event Data Field Contents</i>). Note: bits 3:0 of Event Data 1 are the event offset. It is up to the party issuing this command to ensure that any values written to the event offset field are consistent with values written to the Reading and State fields. The Event Data Bytes operation field in byte 1 of this request can be used to select whether the BMC automatically generates the event offset bits or uses values passed in this byte.</p>
	<p>(9)* <u>Event Data 2</u></p>
	<p>(10)* <u>Event Data 3</u></p>
<p>Response Data</p>	<p>1 Completion Code. Generic plus the following command-specific completion codes: 80h: Attempt to change reading or set or clear status bits that are not settable via this command 81h: Attempted to set Event Data Bytes, but setting Event Data Bytes is not supported for this sensor.</p>

* = Devices must accept a variable number of request data bytes (4 to 10). This requirement is to allow a reduction in the number of data bytes that must be transferred.

The following shows the additions to the Type 01h SDR for the *Set Sensor Reading and Event Status* command. The same changes apply to Table 43-2 for Type 02h SDRs.

Table 43-1, Full Sensor Record - SDR Type 01h

byte	Field Name	size	Description
11	Sensor Initialization	1	<p>... [7] - Settable Sensor 1b = Sensor is settable (Support the <i>Set Sensor Reading</i> command) note: using this bit to report settable sensors is <i>optional</i>. I.e. it is ok to report a settable sensor as 'not settable' in the SDR if it is desired to not report this capability to s/w) 0b = Sensor is not settable [6] - Init Scanning 1b = enable scanning (this bit=1 implies that the sensor accepts the 'enable/disable scanning' bit in the <i>Set Sensor Event Enable</i> command). [5] - Init Events 1b = enable events (per Sensor Event Message Control Support bits in Sensor Capabilities field, and per the Event Mask fields, below). [4] - Init Thresholds 1b = initialize sensor thresholds (per settable threshold mask below). [3] - Init Hysteresis 1b = initialize sensor hysteresis (per Sensor Hysteresis Support bits in the Sensor Capabilities field, below). [2] - Init Sensor Type 1b = initialize Sensor Type and Event / Reading Type code <u>Sensor Default (power up) State</u> Reports how this sensor comes up on device power up and hardware/cold reset. The Initialization Agent does not use this bit. This bit solely reports to software how the sensor comes prior to being initialized by the Initialization Agent. [1] - 0b = event generation disabled, 1b = event generation enabled [0] - 0b = sensor scanning disabled, 1b = sensor scanning enabled</p>

...

Table G-1 (Command Table) Additions:

Table G-1, Command Number Assignments and Privilege Levels

	section	NetFn	CMD	C	U	O	A
...							
Get Sensor Type	35.16	S/E	2Fh		X		
Set Sensor Reading and Event Status	35.17	S/E	30h			X	
FRU Device Commands							

...

Table H-1 Additions:

Table H-1, Sub-function Number Assignments

	Sub Fn #	NetFn	CMD
...			
Get Sensor Type		S/E	2Fh
Set Sensor Reading and Event Status		S/E	30h
FRU Device Commands			

...

Table 35-1 Additions:

Table 35-1, Sensor Device Commands

Command	Section	O/M
...		
Get Sensor Type	35.16	O ^[4]
Set Sensor Reading and Event Status	35.17	O

...

E373 Addendum - Table 42-3, Sensor Type Codes

Added Sensor-specific drive status to the Drive Slot (Bay) sensor type 0Dh. These states mirror the drive slot states used in the SAF-TE and ANSI SES specifications.

Table 42-3, Sensor Type Codes

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
...			
Drive Slot (Bay)	0Dh	00h	Drive Presence
		01h	Drive Fault
		02h	Predictive Failure
		03h	Hot Spare
		04h	Consistency Check / Parity Check in progress
		05h	In Critical Array
		06h	In Failed Array
		07h	Rebuild/Remap in progress
		08h	Rebuild/Remap Aborted (was not completed normally)

...

E374 Addendum - Table 28-3, Get Chassis Status Command

The *Get Chassis Status* command has been updated to include the optional capability for reporting support for the *Chassis Identify* command and the present chassis identify state.

Table 28-3, Get Chassis Status Command

	byte	data field
Request Data	-	-
...		
Response Data	1	Completion Code
	4	Misc. Chassis State
		[7] - reserved
		[6] - 1b = Chassis Identify command and state info supported (Optional) 0b = Chassis Identify command support unspecified via this command. (The <i>Get Command Support</i> command, if implemented, would still indicate support for the <i>Chassis Identify</i> command)
		[5:4] - Chassis Identify State. Mandatory when bit [6] = 1b, reserved (return as 00b) otherwise. Returns the present chassis identify state. Refer to the <i>Chassis Identify</i> command for more info. 00b = chassis identify state = Off 01b = chassis identify state = Temporary (timed) On 10b = chassis identify state = Indefinite On 11b = reserved
		[3] - 1b = Cooling/fan fault detected
		[2] - 1b = Drive Fault
		[1] - 1b = Front Panel Lockout active (power off and reset via chassis push-buttons disabled.)
		[0] - 1b = Chassis intrusion active
...		

E375 Addendum - Table 42-3, Sensor Type Codes

Added sensor-specific offsets for reporting whether a cable is connected or not, and for indicate if a cable is misconnected or interconnection is incorrect, as follows:

Table 42-3, Sensor Type Codes

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
...			
Cable / Interconnect	1Bh	00h 01h	Cable/Interconnect is connected Configuration Error - Incorrect cable connected / Incorrect interconnection
...			

E376 Addendum - Table 42-3, Sensor Type Codes

Added sensor-specific offset to indicate whether memory is being throttled.

Table 42-3, Sensor Type Codes

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
-------------	------------------	------------------------	-------

...			
Memory	0Ch	00h	Correctable ECC / other correctable memory error
		01h	Uncorrectable ECC / other uncorrectable memory error
		02h	Parity
		03h	Memory Scrub Failed (stuck bit)
		04h	Memory Device Disabled
		05h	Correctable ECC / other correctable memory error logging limit reached
		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	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 doesn't match other memory modules).
		08h	Spare. Indicates entity associated with the sensor represents a 'spare' unit of memory.
		09h	<p><i>The Event Data 3 field can be used to provide an event extension code, with the following definition:</i></p> <p><u>Event Data 3</u></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> <p>Memory Automatically Throttled (memory throttling triggered by a hardware-based mechanism operating independent from system software, such as automatic thermal throttling or throttling to limit power consumption.)</p>

E377 Clarification - Section 20.1, Get Device ID Command, & Table 20-2, Get Device ID Command

The description of IPMI version has been clarified to indicate that 02h is used as the IPMI version for implementations that provide IPMI v2.0 capabilities, as follows:

Table 20-2, Get Device ID Command

	byte	data field
Request Data	-	-
Response Data	1	Completion Code
...		
	6	IPMI Version. Holds IPMI Command Specification Version. BCD encoded. 00h = reserved. Bits 7:4 hold the Least Significant digit of the revision, while bits 3:0 hold the Most Significant bits. E.g. a value of 51h indicates revision 1.5 functionality. 02h for implementations that provide IPMI v2.0 capabilities per this specification.

...

Also, the text in section 20.1 describing the IPMI Version field is updated as follows:

IPMI Version

This field holds the version of the IPMI specification that the controller is compatible with. This indicates conformance with this document, including event message formats and mandatory command support. *This field is BCD encoded with bits 7:4 holding the Least Significant digit of the revision and bits 3:0 holding the Most Significant bits.*

The value shall be 02h for implementations that provide IPMI v2.0 capabilities per this specification.

E378 Addendum - Set Serial Routing Mux command

A new command has been added to facilitate the use of Add-in / Adjunct management controllers that can be used to augment BMC functionality. The new command definition and corresponding updates to Table G-1, Command Number Assignments and Privilege Levels, Table H-1, Sub-function Number Assignments, and Table 25-1, IPMI Serial/Modem Commands, as follows.

25.13 Set Serial Routing Mux Command

This optional command supports implementations where an add-in card can take over responsibility for Serial Port Sharing from the BMC. The command enables an add-in card or adjunct management controller to direct the BMC to route serial connections to the add-in or allow them to be handled by the BMC. Logically, this action can be viewed as controlling a hardware multiplexer (serial routing mux) that routes the serial signals between the BMC and the add-in, though this specification does not describe or require a particular hardware implementation for supporting this capability. The command also returns the present setting of the serial routing mux.

For BMC implementations, the setting is volatile with respect to BMC initialization. The BMC ‘power on default’ shall be “BMC controlled”. Otherwise, the BMC must retain this setting across systems resets and power cycles as long as the BMC remains powered (with the exception of actions such as BMC Cold Resets or firmware updates, where the setting is allowed to return to the power-on default).

Table 25-15, Set Serial Routing Mux Command

	byte	data field
Request Data	1	Channel number. This must correspond to the channel number that the desired serial/modem routing mux is associated with. [7:4] - reserved [3:0] - Channel number.
	2	Serial Port Association entry This value matches up with the Serial Port Association Entry value used as the set selector for the System Serial Port Association parameter in the serial configuration parameters for the given channel. This enables support for implementations where different IPMI serial capabilities are associated with different ports or system serial controllers. For example, an implementation where SOL is associated with a different system serial controller than IPMI serial port sharing or IPMI over Serial. (SEE E380)
	3	Mux setting <VOLATILE> The BMC can override these settings on power down, power on, and system resets, and change it during system operation when a serial/modem connection is activated or deactivated. [7:4] - reserved [3:0] - 0h = get present mux setting/status only 1h = serial routing is BMC controlled 2h = force switch of mux to route “ System to Add-In” 3h = force switch of mux to route “Connector to System” 4h = force switch of mux to route “Connector to Add-in”

Response Data	1	Completion Code
	2	Mux setting. This returns the present state of the mux and the mux change bits from the last <i>Set Mux Control</i> command. present mux setting [7:4] - reserved [3:0] - 0h = reserved 1h = routing under BMC control 2h = routing set to "System to Add-in" 3h = routing set to "Connector to System" 4h = routing set to "Connector to Add-in"

Table G-1 (Command Table) Additions:

Table G-1, Command Number Assignments and Privilege Levels

	section	NetFn	CMD	C	U	O	A
...							
Get User Callback Options	25.12	Transport	1Bh		X		
Set Serial Routing Mux	25.13	Transport	1Ch				X
SOL Activating	26.1	Transport	20h	b2	b2	b2	b2

...

Table H-1 Additions:

Table H-1, Sub-function Number Assignments

	Sub Fn #	NetFn	CMD
...			
Get User Callback Options		Transport	1Bh
Set Serial Routing Mux		Transport	1Ch
SOL Activating		Transport	20h

...

Table 25-1 Additions:

Table 25-1, IPMI Serial/Modem Commands

Command	Section Defined	O/M
...		
Get User Callback Options	25.12	O ^[5]
Set Serial Routing Mux	25.13	O

...

E379 Addendum - Command Forwarding

This addendum defines a new set of commands to enable a capability called "Command Forwarding". Please refer to the text of the addendum for details. This addendum adds new sections and commands to the specification, and updates to Table G-1, Command Number Assignments and Privilege Levels, Table H-1, Sub-function Number Assignments, as follows:

35b Command Forwarding Commands

Command Forwarding is an optional capability that can be used to support add-in cards or auxiliary management controllers. This functionality enables the specified commands on a given interface to be forwarded from the BMC to the add-in instead of being processed directly by the BMC. The BMC accomplishes this by encapsulating the forwarded command within a *Forwarded Command* command that it then sends to the target controller on the add-in. Correspondingly, the controller on the add-in can use the *Forwarded Command* to return forwarded command responses to the BMC.

Only requests from the source to the target need to be forwarded. If the target (add-in) needs to deliver a request to a particular channel, it can use the *Send Message* command to do so. Bridging in the BMC will then handle the routing of the response back to the target. Thus, the *Forwarded Command* command is only used to forward *request messages* to the target. Correspondingly, the BMC does not itself accept *Forwarded Command* requests, just responses.

This is similar in operation to the *Send Message* command. The general process for initializing and using Command Forwarding is:

- The *Set Forwarded Commands* command is used to select which commands are to be forwarded from a given channel. In this section, channels that receive commands that are to be forwarded are referred to as sources for Command Forwarding.
- The *Enable Forwarded Commands* command is used to configure which controller will receive the forwarded commands, and also to enable (activate) Command Forwarding. In this section, the controller that receives and processes forwarded commands is referred to as the target controller for Command Forwarding.
- Subsequently, when the BMC receives a command over a channel, it checks to see if Command Forwarding is enabled for that channel, and whether the command is to be Forwarded.
- If the command is to be forwarded, the BMC encapsulates the IPMI common command fields (i.e. NetFn, LUN, CMD) in a *Forwarded Command* request message to the target controller.
- When the BMC issues the *Forwarded Command* command, it temporarily records the sequence number that was used to send that command, along with information necessary to format and route the corresponding response data back to the source channel.
- The target receives the *Forwarded Command* request, processes it, and returns a *Forwarded Command* response. This response contains the encapsulated IPMI message data the original, forwarded, request. The BMC uses the sequence number in this response to look up how to route and format the response data for the particular source channel.

Table 35b-1, Command Forwarding Commands

Command	Section Defined	O/M
Get Forwarded Commands	35b.1	O ⁽¹⁾
Set Forwarded Commands	35b.2	O ⁽¹⁾
Enable Forwarded Commands	35b.3	O ⁽¹⁾
Forwarded Command	35b.4	O ⁽¹⁾

35b.1 Get Forwarded Commands Command

This command enables software to determine which commands are presently enabled for command forwarding from a given channel on the BMC.

Table 35b-2 Get Forwarded Commands Command

	Byte	Data field
Request Data	1	Source Channel Number (number for the channel that is the source of forwarded commands)
	2	[7:6] Operation 00b = return forwarded mask for commands 00h through 7Fh 01b = return forwarded mask for commands 80h through FFh 10b, 11b = reserved [5:0] NetFn
	3	[7:2] reserved [1:0] LUN
Response Data	1	Completion code
	2:17	Forwarded Commands mask These sixteen bytes form a 128-bit bitfield where each bit indicates a particular command value under the given NetFn for which forwarding is enabled. For each bit in the bitfield: 0b = indicates the command is not forwarded 1b = indicates the command is forwarded Depending on the value of the "Operation" parameter passed in the request: Byte 1, bit 0 corresponds to command 00h or command 80h Byte 16, bit 7 corresponds to command 7Fh or command FFh

35b.2 Set Forwarded Commands Command

This command enables software to set which commands are presently enabled for command forwarding from a given channel on the BMC.

Table 35b-3, Set Forwarded Commands Command

	Byte	Data field
Request Data	1	Source Channel Number (number for the channel that is the source of forwarded commands) <i>All supported source channels are configured independently.</i>
	2	[7:6] Operation 00b = set forwarded command mask for commands 00h through 7Fh 01b = set forwarded command mask for commands 80h through FFh 10b = disable Command Forwarding from this channel 11b = enable Command Forwarding form this channel [5:0] NetFn
	3	[7:2] reserved [1:0] LUN
	4:19	Forwarded Command mask These sixteen bytes forms a 128-bit bitfield where each bit indicates a particular command value under the given NetFn for which forwarding is enabled For each bit in the bitfield: 0b = indicates the command is not forwarded 1b = indicates the command is forwarded Depending on the value of the "Operation" parameter passed in the request: Byte 1, bit 0 corresponds to command 00h or command 80h Byte 16, bit 7 corresponds to command 7Fh or command FFh
Response Data	1	Completion code

35b.3 Enable Forwarded Commands Command

This command allows enabling and disabling Command Forwarding, and also provides the ability to configure which interface (channel) the BMC sends forwarded commands to and receives forwarded command responses from.

Note: a given BMC may not support Command Forwarding over all channels. The command returns which channels command forwarding can be targeted to.

Table 35b-4, Enable Forwarded Commands Command

	Byte	Data field
Request Data	1	[7:2] - reserved [1:0] - Operation: 00b = get present configuration 01b = set target controller channel, slave address and LUN 10b = enable Command Forwarding from given channel 11b = disable Command Forwarding from given channel
	2	Channel Number to be used for channel between BMC and target controller [3:0] - channel number.

Response Data

	0h-7h = channel numbers 08h-0Fh = reserved
3	Target Controller LUN [7:2] - reserved [1:0] - target controller LUN
4	Target Controller Slave Address [7:1] - controller slave address [0b] - reserved. Write as 0b.
5	Forwarded command time-out, in 10's of ms. 1-based. 30 ms, min. Sets the minimum time the BMC will wait before timing out waiting for a response to a <i>Forwarded Command</i> command. 00h-02h = reserved. 03h-FFh = timeout in 10's of ms. E.g. 03h = 30 ms.
1	Completion code
2	Command Forwarding Status [7:2] - reserved [1:0] - command forwarding status 11b = command forwarding disabled 10b = command forwarding enabled all other = reserved
3	Channel Number to used for channel between BMC and target controller. [3:0] - channel number. 0h-7h = channel numbers 08h-0Fh = reserved
4	Target Controller LUN [7:2] - reserved [1:0] - target controller LUN
5	Target Controller Slave Address [7:1] - controller slave address [0b] - reserved. Write as 0b.
6	Forwarded command time-out, in 10's of ms. 1-based. 30 ms, min. Sets the minimum time the BMC will wait before timing out waiting for a response to a <i>Forwarded Command</i> command. 00h-02h = reserved. 03h-FFh = timeout in 10's of ms. E.g. 03h = 30 ms.
7:8	Source Channel Support bitfield indicating which channel numbers are available for use for Command Forwarding <u>sources</u> . <i>The implementation must allow all supported source channels for command forwarding to be enabled and used for command forwarding simultaneously.</i> <u>byte 1:</u> [7] - 1b = channel 7 supported for Command Forwarding [6] - 1b = channel 6 supported for Command Forwarding [5] - 1b = channel 5 supported for Command Forwarding [4] - 1b = channel 4 supported for Command Forwarding [3] - 1b = channel 3 supported for Command Forwarding [2] - 1b = channel 2 supported for Command Forwarding [1] - 1b = channel 1 supported for Command Forwarding [0] - 1b = channel 0 (primary IPMB) supported for Command Forwarding <u>byte 2:</u> [7] - 1b = channel Fh (system interface) supported for Command Forwarding. [6:0] - reserved

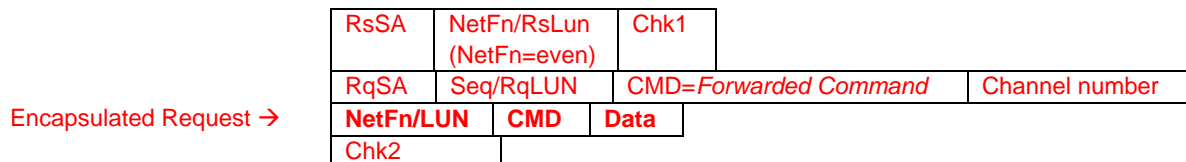
9:10	<p>Target Channel Support bitfield indicating which channel numbers are available for selection as the target channel for forwarded commands.</p> <p><i>Note:</i></p> <ul style="list-style-type: none"> • Only one channel at a time can be set as the target channel per this version of the specification. • Only channels of type IPMB or PCI-SMBus are supported as targets with this version of the specification. • OEM channel use is allowed, but the mechanism used for handling forwarded commands on an OEM channel is outside this specification. <p>byte 1:</p> <p>[7] - 1b = channel 7 supported for Command Forwarding [6] - 1b = channel 6 supported for Command Forwarding [5] - 1b = channel 5 supported for Command Forwarding [4] - 1b = channel 4 supported for Command Forwarding [3] - 1b = channel 3 supported for Command Forwarding [2] - 1b = channel 2 supported for Command Forwarding [1] - 1b = channel 1 supported for Command Forwarding [0] - 1b = channel 0 (primary IPMB) supported for Command Forwarding</p> <p>byte 2: reserved</p>
------	---

35b.4 Forwarded Command Command

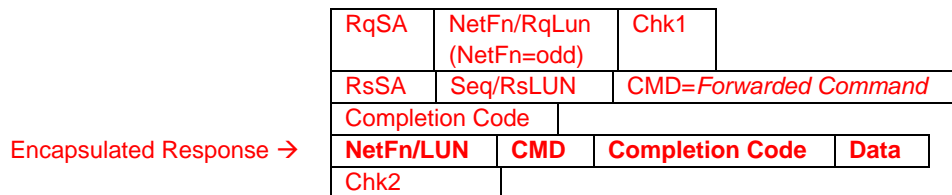
This command is used to encapsulate the forwarded command data between the add-in and the BMC. Below are examples of the format of this command used to forward a request to a target controller on IPMB.

Note that for IPMB this encapsulation adds at least three bytes of overhead to forwarded requests, since there are two occurrences of NetFn/LUN and CMD bytes, plus a field for the source channel number. (If the request is from a session-based channel, two additional bytes of overhead are required.) For responses, there are three bytes of overhead because the completion code byte is also duplicated. Thus, to support this command, the BMC must include sufficient additionally buffering to accept this additional overhead for all interfaces that support using the *Forwarded Command* message to deliver a message to a given target.

Example: Format of *Forwarded Command* request message used to carry a forwarded request from BMC to target controller (add-in) via IPMB:



Example: Format of *Forwarded Command* response message from target controller (add-in) to BMC via IPMB:



The BMC will *time out* and return an FFh error completion code to the requester if the target controller does not return a matching *Forwarded Command* response message within the timeout set by the *Enable Forwarded Commands* command.

The *Forwarded Command* command is only sent out by the BMC as a request. It is not accepted as a request by the BMC itself.

Table 35b-5, Forwarded Command Command

Request Data	1	[7] - 1b = forwarded request is from a session-based channel [6:4] - reserved [3:0] - Channel number.
	2 ^[1]	[7:6] - reserved. [5:0] - User ID. Use 000000b for single-session channels.
	3 ^[1]	[7:4] - User Maximum Privilege Level ^[2] [3:0] - User Operating Privilege Level ^[2] (present privilege level User that originated request is operating at)
	4 ^[1]	Session Handle. Use 00h for single-session channels.
	x:N	Forwarded Command Request Data
Response Data	1	Completion Code
	2:M	Forwarded Command Response Data

1. These fields present only if request is forwarded from a session-based channel.
2. Value is captured at time that the request is received and interpreted by the BMC.

Table G-1 (Command Table) Additions:

Key for Command Privilege Levels Table:

...
b3 = command only generated by BMC, can only be delivered to a session-less channel.

Table G-1, Command Number Assignments and Privilege Levels

	section	NetFn	CMD	C	U	O	A
...							
Command Forwarding Commands							
Forwarded Command	35b.4	Transport	30h	b3	b3	b3	b3
Set Forwarded Commands	35b.1	Transport	31h				X
Get Forwarded Commands	35b.2	Transport	32h		X		
Enable Forwarded Commands	35b.3	Transport	33h				X

...

Table H-1 Additions:

Table H-1, Sub-function Number Assignments

	Sub Fn #	NetFn	CMD
...			
Forwarded Command (NOTE: This command is a byproduct of the Command Forwarding capability being enabled on one or more channels and cannot be directly enabled/disabled via Firmware Firewall)		Transport	30h
Set Forwarded Commands		Transport	31h
Get Forwarded Commands		Transport	32h
Enable Forwarded Commands		Transport	33h

...

E380 Addendum - Table 25-4, Serial/Modem Configuration Parameters

This addendum adds the optional capability of reporting the interconnect topology and naming of serial channels and connectors used for IPMI -over-serial and SOL.

Table 25-4, Serial/Modem Configuration Parameters

Parameter	#	Parameter Data (non-volatile unless otherwise noted) ⁽¹⁾
...		
System Serial Port Association (optional) (This parameter is allowed to be READ ONLY for implementations where the serial port configuration for IPMI is fixed)	51	<p>This parameter can be used to tell which serial controller channel is connected to a given physical connector. It can also indicate whether that serial connector is used with Serial Port Sharing, used for IPMI over Serial</p> <p><u>data 1</u> - set selector = Serial Port Association Entry. 0-based. The set selector is only required to cover entries for serial connectors and/or serial channels that are used with IPMI.</p> <p><u>data 2</u> - serial connector number (A number for the physical connector. The choice of this number is implementation specific. For example, connector '1' may correspond to a connector on the rear of a chassis for one system, and an internal header on another.)</p> <p>[7:4] - IPMI channel number (when connector is used for IPMI over serial) 0h = connector is not used with IPMI over Serial</p> <p>[3:0] - serial connector number 0h = no connector (e.g. when serial controller channel is used with IPMI SOL but is not shared with a serial connector)</p> <p><u>data 3</u> - serial controller channel number (a number for the system serial controller that is presently connected to the connector. The choice of this number is implementation specific.)</p> <p>[7] - serial controller channel is used with IPMI Serial Port Sharing (note: if this bit is 1b then bits [7:4] of data 2 must hold a valid IPMI channel number.)</p> <p>[6] - serial controller channel is used with IPMI SOL</p> <p>[5:4] - reserved</p> <p>[3:0] - serial controller channel number 0h = no channel. (e.g. when a serial connector is just used for IPMI over Serial, and is not shared with the system)</p>

System Connector Names (optional)	52	<p>This parameter can be used to store strings for the serial connector names associated with the system serial association entries described in parameter 51.</p> <p><u>data 1</u> - set selector. 0-based. This matches up with the set selector for parameter 51.</p> <p><u>data 2:17</u> - serial connector name or label. It is recommended that this match up with the connector labeling on the chassis or system board. The first byte of this data indicates the encoding of the string, as follows:</p> <p><u>string data 1:</u> [7:4] - reserved [3:0] - encoding 0h = ASCII+LATIN 1. String is null terminated with 00h. 1h = UTF-8. Is-byte first. String is null terminated with 0000h. 2h = UNICODE. Is-byte first. String is null terminated with 0000h. all other = reserved.</p>
System Serial Channel Names (optional)	53	<p>This parameter can be used to store a string for the serial controller channel names associated with the system serial association entries described in parameter 51.</p> <p><u>data 1</u> - set selector. 0-based. This matches up with the set selector for parameter 51.</p> <p><u>data 2:17</u> - serial channel name or label. The first byte of this data indicates the encoding of the string, as follows:</p> <p><u>string data 1:</u> [7:4] - reserved [3:0] - encoding 0h = ASCII+LATIN 1. String is null terminated with 00h. 1h = UTF-8. Is-byte first. String is null terminated with 0000h. 2h = UNICODE. Is-byte first. String is null terminated with 0000h. all other = reserved.</p>

...

E381 Addendum - Set / Get System Info Command

This addendum defines a new set of optional commands to enable reporting information about the system firmware (BIOS) and operating system of the managed system. This addendum adds new sections and command tables to the specification, and updates to Table 22-1, IPMI Messaging Support Commands, Table G-1, Command Number Assignments and Privilege Levels, and Table H-1, Sub-function Number Assignments, as follows:

22.14a Set System Info Command

This command is used for setting system information parameters such as system name and BIOS/system firmware revision information.

Table 22-16a, Set System Info Parameters Command

	byte	data field
Request Data	1	Parameter selector
	2:N	Configuration parameter data, per <i>Table 22-16c, System Info Parameters</i>
Response Data	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

22.14b Get System Info Command

This command is used for retrieving system information parameters from the *Set System Info Parameters* command.

Table 22-16b, Get System Info Parameters Command

	byte	data field
Request Data	1	[7] - 0b = get parameter 1b = get parameter revision only. [6:0] - reserved
	2	Parameter selector
	3	Set Selector. Selects a given set of parameters under a given Parameter selector value. 00h if parameter doesn't use a Set Selector.
	4	Block Selector (00h if parameter does not require a block number)
Response Data	1	Completion Code. Generic codes, plus following command-specific completion code(s): 80h = parameter not supported.
	2	[7:0] - Parameter revision. Format: MSN = present revision. LSN = oldest revision parameter is backward compatible with. 11h for parameters in this specification. <i>The following data bytes are not returned when the 'get parameter revision only' bit is 1b.</i>
	3:N	Configuration parameter data, per Table 22-16c, System Info Parameters If the rollback feature is implemented, the BMC makes a copy of the existing parameters when the 'set in progress' state becomes asserted (See the Set In Progress parameter #0). While the 'set in progress' state is active, the BMC will return data from this copy of the parameters, plus any uncommitted changes that were made to the data. Otherwise, the BMC returns parameter data from non-volatile storage.

Table 22-16c, System Info Parameters

Parameter	#	Parameter Data (non-volatile unless otherwise noted) ^[1]
Set In Progress (volatile)	0	<p><u>data 1</u> - 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 that 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] - 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 'set in progress' value is present on these bits.</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>
System Firmware version	1	<p>System Firmware Version string in text.</p> <p>System firmware that requires multiple strings to represent version information can separate those strings using 00h as the delimiter for ASCII+LATIN1 and UTF-8 encoded string data, or 0000h for UNICODE encoded string data.</p> <p>For IA32 and EMT64 utilizing non-EFI system firmware, it is recommended that four blocks (64 bytes) of storage be provided. For EFI-based systems, 256 bytes is recommended.</p>

		<p>Note: System firmware may optionally include a routine that checks during POST to see if this parameter is up-to-date with the present firmware version, and if not, update it automatically. Other implementations may elect to automatically update this parameter when system firmware updates occur.</p> <p><u>data 1</u> - set selector = 16-byte data block number to access, 0 based. Two data blocks (32-bytes) for string data required, at least three recommended. Number of effective characters will be dependent on the encoding selected in string data byte 1.</p> <p><u>data 2:17</u> - 16-byte block for system firmware name string data</p> <p>For the first block of string data (set selector = 0), the first two bytes indicate the encoding of the string and its overall length as follows: <u>string data byte 1:</u> [7:4] - reserved [3:0] - encoding 0h = ASCII+Latin1 1h = UTF-8 2h = UNICODE all other = reserved. <u>string data byte 2:</u> [7:0] - string length (in bytes, 1-based)</p>
System name	2	<p>System Name. A name for the overall system to be associated with the BMC. This may or may not match other names that are used for the system.</p> <p><u>data 1</u> - set selector = 16-byte data block number to access, 0 based. Two data blocks (32-bytes) for string data required, at least three recommended. Number of effective characters will be dependent on the encoding selected in string data byte 1.</p> <p><u>data 2:17</u> - 16-byte block for system name string data</p> <p>For the first block of string data (set selector = 0), the first two string data bytes indicate the encoding of the string and its overall length as follows. There is no required value to be set or used for any bytes that are past the string length. <u>string data byte 1:</u> [7:4] - reserved [3:0] - encoding 0h = ASCII+Latin1 1h = UTF-8 (ls-byte first) 2h = UNICODE (ls-byte first) all other = reserved. <u>string data byte 2:</u> [7:0] - string length (in bytes, 1-based)</p>

<p>Primary Operating System Name (non-volatile)</p>	<p>3</p>	<p>Primary Operating system name. The OS that the system boots to for this BMC according to the default configuration of its system firmware. (Note: in systems that may have multiple physical partitions, this reflects the OS for the partition that the given BMC is in. For systems that have virtual machine capability being utilized [where more than one virtual systems may be sharing a physical BMC], it is recommended that this value hold the name of the virtual machine monitor (VMM) software or VMM type)</p> <p><u>data 1</u> - set selector = 16-byte data block number to access, 0 based. Two data blocks (32-bytes) for string data required, at least three recommended. Number of effective characters will be dependent on the encoding selected in string data byte 1.</p> <p><u>data 2:17</u> - 16-byte block for system name string data</p> <p>For the first block of string data (set selector = 0), the first two bytes indicate the encoding of the string and its overall length as follows. There is no required value to be set or used for any bytes that are past the string length.</p> <p><u>string data byte 1:</u> [7:4] - reserved [3:0] - encoding 0h = ASCII+Latin1 1h = UTF-8 2h = UNICODE all other = reserved.</p> <p><u>string data byte 2:</u> [7:0] - string length (in bytes, 1-based)</p>
<p>Operating System Name (volatile)</p>	<p>4</p>	<p>Present Operating system name. The name of the operating system that is presently running and able to access this BMC's system interface. The BMC automatically clears this value by zeroing out the string length on system power cycles and resets.</p> <p>(Note: in systems that may have multiple physical partitions, this reflects the OS for the partition that the given BMC is in. For systems that have virtual machine capability being utilized [where more than one virtual systems may be sharing a physical BMC], it is recommended that this value hold the name of the virtual machine monitor (VMM) software or VMM type)</p> <p><u>data 1</u> - set selector = 16-byte data block number to access, 0 based. Two data blocks (32-bytes) for string data required, at least three recommended. Number of effective characters will be dependent on the encoding selected in string data byte 1.</p> <p><u>data 2:17</u> - 16-byte block for system name string data</p> <p>For the first block of string data (set selector = 1), the first two bytes indicate the encoding of the string and its overall length as follows:</p> <p><u>string data byte 1:</u> [7:4] - reserved [3:0] - encoding 0h = ASCII+Latin1 1h = UTF-8 2h = UNICODE all other = reserved.</p> <p><u>string data byte 2:</u> [7:0] - string length (in bytes, 1-based)</p>
<p>OEM</p>	<p>192 ... 255</p>	<p>This range is available for special OEM system information parameters.</p>

- Choice of system manufacturing defaults for non-volatile parameters is left to the system manufacturer unless otherwise specified.

Table 22-1, IPMI Messaging Support Commands, Additions:

Table 22-1, IPMI Messaging Support Commands

Command	Section Defined	O/M
Set BMC Global Enables	22.1	M
...		
Get System GUID	22.14	O ^[5]
Set System Info	22.14a	O
Get System Info	22.14b	O ^[9]
...		
Set User Password Command	22.30	O ^[4]

1. Optional if the System Interface is the only channel that's implemented.

...

9. Mandatory if Set System Info command is implemented.

Table G-1 (Command Table) Additions:

Table G-1, Command Number Assignments and Privilege Levels

	section	NetFn	CMD	C	U	O	A
...							
BMC Device and Messaging Commands							
...							
Set System Info	22.14a	App	58h				X
Get System Info	22.14b	App	59h		X		
...							

Table H-1 Additions:

Table H-1, Sub-function Number Assignments

	Sub Fn #	NetFn	CMD
...			
Set System Info		App	58h
Get System Info		App	59h
...			

E382 Clarification - Table 21-2, Get NetFn Support Command

There was some ambiguity in the specification of the bit ordering returned by the Get NetFn Support command regarding whether the bytes were returned in the order "NetFn 0h-Fh for LUN 0", "NetFn 10h-1Fh for LUN 0", etc. or in the order "NetFn 0h-Fh for LUN 0", "NetFn 0h-Fh for LUN 1", etc. I.e. whether the data was indexed first by NetFn then LUN, or LUN then NetFn. This is clarified with additional example text in the command as follows:

Table 21-2, Get NetFn Support Command

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-7h, Fh = channel numbers Eh = retrieve information for channel this request was issued on.
IPMI Response Data	1	Completion Code
...		
	3:18	<p>There are 32 possible Network Function (NetFn) pairs. The following bytes are treated as bitfields where each bit indicates the support for a given Network Function pair. Thus, it takes 4 bytes to fully list support for NetFn values under a given LUN. Since there are four possible LUNs for a management controller, a total of 16 bytes will return the settings for all four possible LUNs. 0b = NetFn pair is not used, 1b = NetFn pair is used</p> <p>byte 1, bit 0 corresponds to NetFn pair 0h,1h for LUN 00b byte 1, bit 7 corresponds to NetFn pair Eh,Fh for LUN 00b byte 2, bit 0 corresponds to NetFn pair 10h,11h for LUN 00b byte 2, bit 7 corresponds to NetFn pair 1Eh, 1Fh for LUN 00b ... byte 16, bit 0 corresponds to NetFn pair 30h, 31h for LUN 11b byte 16, bit 7 corresponds to NetFn pair 3Eh, 3Fh for LUN 11b</p>

E383 Clarification - Table 23-4, LAN Configuration Parameters

The specification did indicate how VLAN IDs for alerts should be handled and reported in implementations that can only support a single setting for VLAN ID for all IPMI traffic (alert and non-alert) through it's management network interface. This is clarified in the LAN Configuration Parameters as follows:

Table 23-4, LAN Configuration Parameters

...		
Destination Address VLAN TAGs (can be READ ONLY, see description)	25	<p>Sets/Gets the VLAN IDs (if any) addresses that a LAN alert can be sent to. This parameter is not present if the Number of Destinations parameter is 0, or if the implementation does not support the use of VLAN IDs for alerts. Otherwise, the number of VLAN TAG entries matches the number of Alert Destinations.</p> <p>An implementation may only be able to send alerts using the same VLAN TAG configuration as specified by parameters 20 and 21, in which case this parameter is allowed to be READ ONLY, where data 3-4 reflects the settings of parameters 20 and 21, and data 2 [7:4] indicates that VLAN TAGs are being used for alerts. If the implementation <i>does</i> support configurable VLAN TAGs for alert destinations, it must support configuring unique TAG information for all destinations on the given channel.</p> <p><u>data 1</u> - Set Selector = Destination Selector. [7:4] - reserved [3:0] - Destination selector. Destination 0 is always present as a volatile destination that is used with the <i>Alert Immediate</i> command.</p> <p><u>data 2</u> - Address Format [7:4] - Address Format. 0h = VLAN ID not used with this destination 1h = 802.1q VLAN TAG [3:0] - reserved</p> <p>For Address Format = 1h: <u>data 3-4</u> - VLAN TAG [7:0] - VLAN ID, least-significant byte [11:8] - VLAN ID, most-significant nibble [12] - CFI (Canonical Format Indicator. Set to 0b) [15:13] - User priority (000b, typical)</p>

E384 Clarification - Table 27-3, Set Watchdog Timer Command, Table 27-4, Get Watchdog Timer Command

It was unclear that SMI and NMI interrupt support was optional for the watchdog timer. This is clarified as follows.

Table 27-3, Set Watchdog Timer Command

byte data field

...	
2	<p>Timer Actions [7] - reserved [6:4] - pre-timeout interrupt (logged on expiration when "don't log" bit = 0b) 000b = none 001b = SMI (optional) 010b = NMI / Diagnostic Interrupt (optional) 011b = Messaging Interrupt (this is the same interrupt as allocated to the messaging interface, if communications interrupts are supported for the system interface) 100b,111b = reserved [3] - reserved [2:0] - timeout action 000b = no action 001b = Hard Reset 010b = Power Down 011b = Power Cycle 100b,111b = reserved</p>

...

Table 27-4, Get Watchdog Timer Command

byte data field

...

3	Timer Actions [7] - reserved [6:4] - pre-timeout interrupt 000b = none 001b = SMI (if implemented) 010b = NMI / Diagnostic Interrupt (if implemented) 011b = Messaging Interrupt (this would be the same interrupt as allocated to the messaging interface) 100b,111b = reserved [3] - reserved [2:0] - timeout action 000b = no action 001b = Hard Reset 010b = Power Down 011b = Power Cycle 100b,111b = reserved
---	---

...

E385 Clarification/Errata - Section 13.28.4, Integrity Algorithms

The specification did not clearly indicate the usage of K1. K1 is used in the HMAC-SHA1-96 and HMAC-MD5-128 integrity algorithms for RMCP+. This is clarified as follows.

13.28.4 Integrity Algorithms

The Integrity Algorithm Number specifies the algorithm used to generate the contents for the AuthCode “signature” field that accompanies authenticated IPMI v2.0/RMCP+ messages once the session has been established.

Unless otherwise specified, the integrity algorithm is applied to the packet data starting with the AuthType/Format field up to and including the field that immediately precedes the AuthCode field itself.

HMAC-SHA1-96 and HMAC-MD5-128 take the Session Integrity Key and use it to generate K1. K1 is then used as the key for use in HMAC for data integrity. For “two-key” logins, 160-bit key KG is used in the creation of SIK. For “one-key” logins, the user’s key (password) is used in place of KG. To maintain a comparable level of authentication, it is recommended that a full 160-bit user key be used when “one-key” logins are enabled for IPMI v2.0/RMCP+.

E386 Clarification - Table 22-26, Get AuthCode Command

The specification was not clear that the User Password was to be used as the starting key when RMCP+ Integrity Algorithms were used to generate the hash for the Get AuthCode command. Also the completion code still referred to straight-password checking, which was deleted for IPMI v2.0. This is clarified as follows:

Table 22-26, Get AuthCode Command

	byte	data field
IPMI Request Data	1	<p>[7:6] - Authentication Type / Integrity Algorithm Number 00b = IPMI v1.5 AuthCode Algorithms 01b = IPMI v2.0/RMCP+ Algorithm Number</p> <p>For [7:6] = 00b, IPMI v1.5 AuthCode Number: [5:4] - reserved [3:0] - hash type 0h = reserved 1h = MD2 2h = MD5 3h = reserved 4h = Reserved (change from IPMI v1.5). This shall result in an error completion code. 5h = OEM proprietary all other = reserved</p> <p>For [7:6] = 01b, IPMI v2.0/RMCP+ Integrity Algorithm Number [5:0] - Integrity Algorithm Number. See <i>Table 13-18, Integrity Algorithm Numbers</i>. The User Password is used as the starting key for the Integrity Algorithm, instead of session-dependent keys such as the Session Integrity Key. The "none" Integrity Number (0) is illegal and shall result in an error completion code.</p>
		...
IPMI Response Data	1	Completion Code

...

Last Page