- 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 4

June 12, 2009

Contents

Introduction

This document presents cumulative 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.

Revision 1 (6/1/04) Addenda, Errata, and Clarifications

E269 (See 2/15/06 Addenda, Errata, and Clarifications, below)

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~~ 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) *The Event Data 2 field can be used to identify which network controller the leash was lost on where 00h corresponds to the first (or only) network controller.*
		05h	Unauthorized dock~~/undock~~
		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	~~11b~~	~~Start Of Record~~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 Byte 1: [~~5~~7:0] = 1100_0000b. Start of Record, ~~Start of record followed by Cypher Suite ID tag~~Standard Cipher Suite~~.~~ Data following C0h (1100_0000b) ~~tag~~ start of record byte: Byte ~~1~~ 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 *Table 22-19, Cipher Suite IDs*. [5:0] = 1100_0001b. Start or Record, ~~Start of record followed by Cipher Suite ID + OEM IANA tag~~OEM Cipher Suite~~.~~ Data following C1h (1100_0001) ~~tag~~ start of record byte: Byte ~~1~~ 2 - OEM Cipher Suite ID See *Table 22-19, Cipher Suite IDs*. Byte ~~2~~3:~~4~~ 5 - OEM IANA Least significant byte first. 3-byte IANA for the OEM or body that defined the Cipher Suite.

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	~~X~~[b]	[10]	[10]	[10]
Deactivate Payload	24.2	App	49h	~~X~~[b]	[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. ~~If an IPMB is provided, the SDR Repository must be accessible via that interface as well.~~ ~~SDR Repository access when the system is powered up or in ACPI 'S1' sleep is mandatory, but access when the system is powered down or in a >S1 sleep state is optional.~~

...

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:

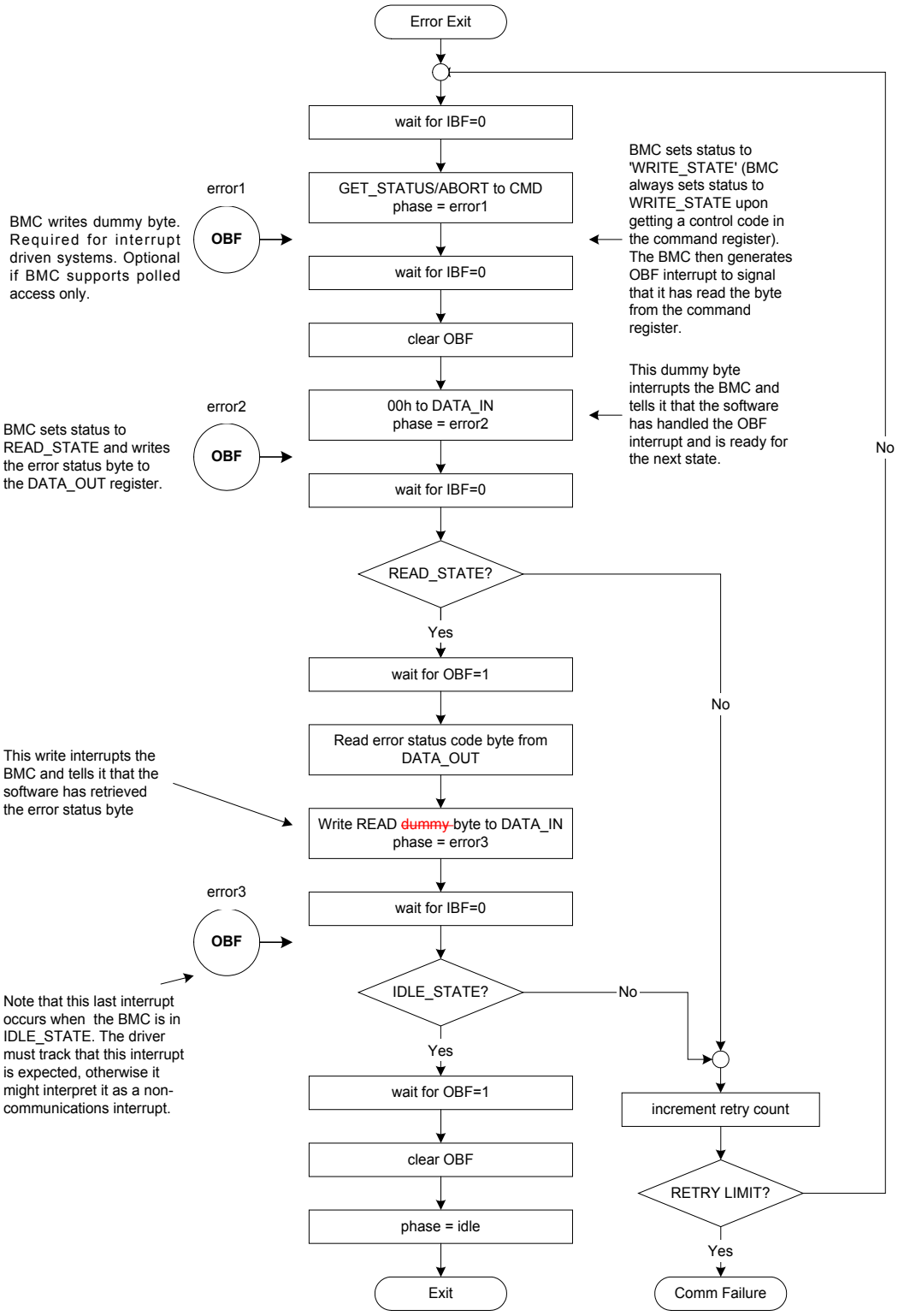
Table 25-4, Serial/Modem Configuration Parameters

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. data 1 - 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
	...	
	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.
		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.*
	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	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		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 of 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:) Event Data 2 [7:4] - reserved [3:0] - restart cause per *Get System Restart Cause* command. Event Data 3 Channel number used to deliver command that generated restart, per *Get System Restart Cause* command.
...			
OS ~~Critical~~ Stop / Shutdown	20h	00h	Critical s~~S~~top 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]. Event Data 2 [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	~~5~~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 *Section 17.11, Alert Policy Table*.
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 K_G (160 bits) was not explicitly defined, nor were the sizes of K_G and K_R 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 K_G is used in the creation of SIK. For "one key" logins, the user's key (password) is used in place of K_G. 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+ "K_R" key (20 bytes). The "K_R" key is used as a unique value for random number generation. Note: A BMC implementation is allowed to share a single K_R value across all channels. A utility can set K_R 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+ "K_G" key (20 bytes). "K_G" 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

...

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/I2C Device Type Codes" should be "Table 43- 12, IPMB/I2C 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, Disovering 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' ~~ala~~ 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
…			
~~Reserved~~	~~1~~	~~36~~	~~This field must always be 01h to be compatible with any software that implements previous versions of this spec.~~
Interface Type	1	~~37~~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 ~~three~~ 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 ~~unlocked~~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 ~~is not lockable~~cannot be locked. This is to ensure ~~in order to enable~~ a password/key configuration utility ~~to~~ 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.

~~**Session Sequence Number Generation**~~

~~Session sequence numbers are generated on a per-session basis. The inbound and outbound sequence numbers are updated and tracked independently. The BMC and the remote console independently select the starting session sequence number for the messages they receive.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 *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

~~Session sequence numbers are tracked on a per-session basis.~~ 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) Event Data 2 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	FRU failure 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: Event Data 2 [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) [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. Event Data 3 For LOGICAL FRU DEVICE (accessed via FRU commands to mgmt. controller): [7:0] - FRU Device ID within controller that generated the event.FFh = reserved. For non-intelligent FRU device: [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. [0] - reserved.

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

	Byte	data field
Request Data	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.
		...

		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 0-1, Event Request Message Fields, for specification of the individual fields.
	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	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		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		
...			

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	~~Action Support~~ [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 comparisions 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] - ~~reserved~~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 *Get Device ID* 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, SEL Record Formats.*
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. data 1 7:0 - mask bit positions 7 to 0, respectively. data 2 15:8 - mask bit positions 15 to 8, respectively.\ For OEM record matching: data 1 = OEM data 5 (FFh = match any) data2 = 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 *Section 17.8, Event Data 1 Event Offset Mask* 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 *never* 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 *Section 17.8, Event Data 1 Event Offset Mask* for more information). Here, 'test value' refers to the Event Data value *after* 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 *Section 17.8, Event Data 1 Event Offset Mask* for more information). Here, 'test value' refers to the Event Data value *after* 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. Comands 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 *Note that a BMC uses LUN 10b for message bridging. The message bridging capability is enabled/disabled by enabling/disabling the Send Message command.* 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. Comands 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
		For Network Function = 2Ch:
	(4)	Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)
		For Network Function = 2Eh:
	(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
		For Network Function = 2Ch:
	(4)	Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)
		For Network Function = 2Eh:
	(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.
	...	
		For Network Function = 2Ch:
	5	Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)
		For Network Function = 2Eh:
	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.
	...	
		For Network Function = 2Ch:
	(4)	Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)
		For Network Function = 2Eh:
	(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.
	...	
		For Network Function = 2Ch:
	(5)	Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)
		For Network Function = 2Eh:
	(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.
	...	
		For Network Function = 2Ch:
	(19)	Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)
		For Network Function = 2Eh:
	(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.
	…	
		For Network Function = 2Ch:
	(19)	Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)
		For Network Function = 2Eh:
	(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 ~~the~~ 20-byte Key Exchange ~~20-byte~~ 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 ~~the~~ HMAC with MD5 ~~is~~ (per [RFC2104] is used for RAKP authentication operations in place of SHA-1. Thus, the Key Exchange Authentication Code fields in RAKP Message ~~3~~ 2 and RAKP Message ~~4~~ 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-~~HMAC-SHA1~~ uses pre-shared symmetric keys ~~and HMAC-based integrity algorithms~~ 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 ~~this specification~~example, the RAKP-HMAC-SHA1 authentication algorithm ~~shall~~ 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, $\mathbf{R_M}$, a requested role, $\mathbf{Role_M}$, a user name length, $\mathbf{ULength_M}$, a user name (optional - denoted by < > below), $\mathbf{UName_M}$, and the managed system's Session ID, $\mathbf{SID_C}$, 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 $\mathbf{SID_C}$ is active and that a session can be created using $\mathbf{Role_M}$, $\mathbf{ULength_M}$, and (optional), $\mathbf{UName_M}$ for the given selections for security algorithms.

If the request is valid, the managed system then selects a random number, $\mathbf{R_C}$, and sends to the remote console as Message 2 the values $\mathbf{SID_M}$, $\mathbf{R_C}$, and $\mathbf{GUID_C}$ as well as the HMAC per [~~RFC2404~~RFC2104] of the values ($\mathbf{SID_M}$, $\mathbf{SID_C}$, $\mathbf{R_M}$, $\mathbf{R_C}$, $\mathbf{GUID_C}$, $\mathbf{Role_M}$, $\mathbf{ULength_M}$, < $\mathbf{UName_M}$ >) generated using key $\mathbf{K_{[UID]}}$ ~~selected by~~ associated with the given username, $\mathbf{UName_M}$, and role, $\mathbf{Role_M}$.

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 *entire* byte holding the Requested Privilege Level field)
$ULength_M$	1	User Name Length byte (number of bytes of $UName_M$ = 0 for 'null' username)
$UName_M$	~~1~~var	User Name bytes (absent for 'null' username)

Where $HMAC_{K[UID]}$ (SID_M, SID_C, R_M, R_C, $GUID_C$, $Role_M$, $ULength_M$, < $UName_M$ >) represents the value for the Key Exchange Authentication Code field in RAKP Message 2. (The $HMAC_{K[UID]}$ 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 K[UID] is the user-specific key that is associated with the given usernname 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 $\mathbf{SID_M}$ is active and that $\mathbf{GUID_C}$ matches the managed system that the remote console is expecting to communicate with. The remote console then validates the ~~HMAC~~Key Exchange Authentication Code from the message. If the ~~HMAC~~ code is valid, the remote console creates the Session Integrity Key (SIK) by generating an HMAC per [RFC2104] of the concatenation of $\mathbf{R_M}$, $\mathbf{R_C}$, $\mathbf{Role_M}$, $\mathbf{ULength_M}$, and (optional) $\mathbf{UName_M}$ using 160-bit key $\mathbf{K_G}$ (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 $\mathbf{K_{[UID]}}$ is used in place of Kg if 'one-key' logins are being used. See *13.28.4, Integrity Algorithms*)

$$\mathbf{SIK = HMAC_{KG}\ (R_M \mid R_C \mid Role_M \mid ULength_M \mid < UName_M >)}$$

Then the remote console sends to the managed system as Message 3 the value $\mathbf{SID_C}$ and (for the RAKP-HMAC-SHA1 algorithm) the HMAC per [~~RFC2404~~RFC2104] of the values ($\mathbf{R_C}$, $\mathbf{SID_M}$, $\mathbf{Role_M}$, $\mathbf{ULength_M}$, $\mathbf{< UName_M >}$) generated using key $\mathbf{K_{[UID]}}$ selected by the username, $\mathbf{UName_M}$, and role $\mathbf{Role_M}$.

Message 3: Remote Console -► Managed System

$$\mathbf{SID_C,\ HMAC_{K[UID]}\ (R_C,\ SID_M,\ Role_M,\ ULength_M,\ < UName_M >)}$$

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

$$\mathbf{SIK = HMAC_{KG}\ (R_M \mid R_C \mid Role_M \mid ULength_M \mid < UName_M >)}$$

The managed system then sends to the management console as Message 4 the values $\mathbf{SID_M}$, and (for the RAKP-HMAC-SHA1 algorithm) the HMAC per [RFC2404] of the values ($\mathbf{R_M}$, $\mathbf{SID_C}$, $\mathbf{GUID_C}$) generated using key **SIK**. ~~The managed system then transitions into the *Message Transfer* session state.~~

Message 4: Managed System -► Mgmt Console

$$\mathbf{SID_M,\ HMAC_{SIK}\ (R_M,\ SID_C,\ GUID_C)}$$

Where $HMAC_{K[UID]}$(R_C, SID_M, $Role_M$, $ULength_M$, < $UName_M$>) represents the value in the Integrity Check Value field for RAKP Message 4. After receiving Message 4, the management console verifies that the value $\mathbf{SID_M}$ is active and then validates the ~~HMAC~~Integrity Check Value. If the ~~HMAC~~ 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

	byte	data field
Request Data	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. ~~[7:6] - reserved.~~ [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).

...

Revision 2 (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, … ~~6Fh~~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:~~18~~19	Enable/Disable Mask …
	For Network Function = 2Ch:
(~~19~~20)	Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)
	For Network Function = 2Eh:
(~~19:21~~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:~~18~~17	Enable/Disable Mask ...
	For Network Function = 2Ch:
(18)	Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)
	For Network Function = 2Eh:
(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.
~~5:6~~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 *Table H-1, Sub-function Number Assignments*. ...

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	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. 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.

		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
Response Data	1	Completion Code ...
	2:5	Auxiliary Response Data. LS-byte first. For Payload = SOL: [31:1] - reserved. ~~Return as 00_00_00_00h.~~ [0] - 0b = test mode not supported / enabled 1b = test mode enabled

...

Table 15-2, SOL Payload Data Format

Field	Size	Description	
		...	
Operation / Status	1	**BMC to Remote Console:** Operations are executed *before* 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.	**Remote Console to BMC:** Note: Operations are executed *before* 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 ~~Pause~~ 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] ~~Drop~~ 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.

		[Remote console can use this to tell if SOL was deactivated by some other party, or by local pushbutton reset or power on/off]. 0b: SOL is active. [3] Transmit Overrun 1b: characters were dropped between transmitting this packet and the previous packet, because the system did not pay attention to hardware flow control. 0b: no characters were lost between this packet and the preceding packet. [2] Break 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. 0b: no break detected [1:0] For test mode = inactive: Reserved For test mode = active: [1] - 1b = RTS asserted [0] - 1b = DTR asserted 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.	[1] Flush Inbound for test mode = inactive: 1b: Drop (flush) data from remote console to BMC [not including data carried in this packet, if any] for test mode = active: 1b: Deassert DSR to baseboard serial controller 0b: Assert DSR to baseboard serial controller. [0] Flush Outbound for test mode = inactive: 1b: Flush Outbound Character Data (flush data from BMC to remote console) for test mode = active: reserved. Write as 0b.

…

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

…

	For Network Function = 2Eh:
~~6:8~~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 ~~sixteen~~ 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 "~~Set Selector~~Operation" parameter passed in the request: ...

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

...

2:5	Support Mask (ls-byte first) These ~~sixteen~~ 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 ~~look~~ set ~~up~~ command support for. The management controller will ~~return~~ set the same values for odd or even NetFn values. I.e. the value for bit [0] is ignored.

...

4:~~18~~19	Enable/Disable Mask ... Depending on the value of the "~~Set Selector~~Operation" parameter passed in the request: ...

Table 21-8, Get Command Enables Command

...

2:~~18~~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 ~~look up~~ set command support for. The management controller will ~~return~~ 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 *non-volatile* 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 ~~sixteen~~ 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

...

~~5:6~~2:5	Sub-Function Enables (ls-byte first) These ~~sixteen~~ 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 *Table H-1, Sub-function Number Assignments*. 1b sub-function is enabled 0b sub-function is disabled or is un-configurable/reserved. [~~15~~31] - bit for sub-function ~~15~~31. [~~14~~30] - bit for sub-function ~~14~~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.) *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.*
...		

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	Operation [7:6] - Event Data Bytes operation 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. 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. [5:4] - Assertion bits operation 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. 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 [3:2] - Deassertion bits operation 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. 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

	[1:0] - Sensor Reading operation 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
3	Sensor Reading Byte 1: byte of reading.
(4)*	For sensors with threshold based events: (High-going events are asserted when value first becomes ≥ threshold. Low-going events are asserted when value first becomes ≤ corresponding threshold.) [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 For sensors with discrete events: [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
(5)*	For sensors with threshold based events: [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 For sensors with discrete events: (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

	(6)*	For sensors with threshold based events: (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 For sensors with discrete events: [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
	(7)*	For sensors with threshold based events: [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 For sensors with discrete events: (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
	(8)*	Event Data 1 (See *Table 29 6, Event Request Message Event Data Field Contents*). 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.
	(9)*	Event Data 2
	(10)*	Event Data 3
Response Data	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.

* = 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	[7] - Settable Sensor 1b = Sensor is settable (Support the *Set Sensor Reading* command) note: using this bit to report settable sensors is *optional*. 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 *Set Sensor Event Enable* 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 Sensor Default (power up) State 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
			...

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 [7:4] - reserved [6] - 1b = Chassis Identify command and state info supported (Optional) 0b = Chassis Identify command support unspecified via this command. (The *Get Command Support* command, if implemented, would still indicate support for the *Chassis Identify* 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 *Chassis Identify* 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. *The Event Data 3 field can be used to provide an event extension code, with the following definition:* Event Data 3 [7:0] - Memory module/device (e.g. DIMM/SIMM/RIMM) identification, relative to the entity that the sensor is associated with (if SDR provided for this sensor).
		09h	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.)

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 bit s.* *The value shall be **02h** for implementations that provide IPMI v2.0 capabilities per this specification.~~**51h** indicating conformance with this specification, version 1.5.~~*

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 *Set Mux Control* 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 uses 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) *All supported source channels are configured independently.*
	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. 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 *Forwarded Command* command. 00h-02h = reserved. 03h-FFh = timeout in 10's of ms. E.g. 03h = 30 ms.
Response Data	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 *Forwarded Command* 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 sources. *The implementation must allow all supported source channels for command forwarding to be enabled and used for command forwarding simultaneously.* byte 1: [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 byte 2: [7] - 1b = channel Fh (system interface) supported for Command Forwarding. [6:0] - reserved

9:10	Target Channel Support bitfield indicating which channel numbers are available for selection as the target channel for forwarded commands. *Note:* • *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.* byte 1: [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 byte 2: reserved

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:

Encapsulated Request →

RsSA	NetFn/RsLun (NetFn=even)	Chk1	
RqSA	Seq/RqLUN	CMD=*Forwarded Command*	Channel number
NetFn/LUN	**CMD**	**Data**	
Chk2			

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

Encapsulated Response →

RqSA	NetFn/RqLun (NetFn=odd)	Chk1	
RsSA	Seq/RsLUN	CMD=*Forwarded Command*	
Completion Code			
NetFn/LUN	**CMD**	**Completion Code**	**Data**
Chk2			

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 Firmwall)		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	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 data 1 - 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. data 2 - 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.) [7:4] - IPMI channel number (when connector is used for IPMI over serial) 0h = connector is not used with IPMI over Serial [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) data 3 - 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.) [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.) [6] - serial controller channel is used with IPMI SOL [5:4] - reserved [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)

System Connector Names (optional)	52	This parameter can be used to store strings for the serial connector names associated with the system serial association entries described in parameter 51. data 1 - set selector. 0-based. This matches up with the set selector for parameter 51. data 2:17 - 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: string data 1: [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.
System Serial Channel Names (optional)	53	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. data 1 - set selector. 0-based. This matches up with the set selector for parameter 51. data 2:17 - serial channel name or label. The first byte of this data indicates the encoding of the string, as follows: string data 1: [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.

...

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 *Table **22-16c**, System Info Parameters*
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.
		The following data bytes are not returned when the 'get parameter revision only' bit is 1b.
	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	data 1 - This parameter is used to indicate when any of the following parameters are being updated, and when the updates are completed. The bit is primarily provided to alert software that some other software or utility is in the process of making changes to the data. 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. If used, the roll back shall restore all parameters to their previous state. Otherwise, the change shall take effect when the write occurs. [7:2] - reserved [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. 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. 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. 11b = reserved
System Firmware version	1	System Firmware Version string in text. 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. 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.

		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. data 1 - 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. data 2:17 - 16-byte block for system firmware name string data 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: string data byte 1: [7:4] - reserved [3:0] - encoding 0h = ASCII+Latin1 1h = UTF-8 2h = UNICODE all other = reserved. string data byte 2: [7:0] - string length (in bytes, 1-based)
System name	2	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. data 1 - 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. data 2:17 - 16-byte block for system name string data 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. string data byte 1: [7:4] - reserved [3:0] - encoding 0h = ASCII+Latin1 1h = UTF-8 (ls-byte first) 2h = UNICODE (ls-byte first) all other = reserved. string data byte 2: [7:0] - string length (in bytes, 1-based)

Primary Operating System Name (non-volatile)	3	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) data 1 - 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. data 2:17 - 16-byte block for system name string data 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. string data byte 1: [7:4] - reserved [3:0] - encoding 0h = ASCII+Latin1 1h = UTF-8 2h = UNICODE all other = reserved. string data byte 2: [7:0] - string length (in bytes, 1-based)
Operating System Name (volatile)	4	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. (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) data 1 - 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. data 2:17 - 16-byte block for system name string data 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: string data byte 1: [7:4] - reserved [3:0] - encoding 0h = ASCII+Latin1 1h = UTF-8 2h = UNICODE all other = reserved. string data byte 2: [7:0] - string length (in bytes, 1-based)
OEM	192 … 255	This range is available for special OEM system information parameters.

1. 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	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 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

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	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. 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 *does* support configurable VLAN TAGs for alert destinations, it must support configuring unique TAG information for all destinations on the given channel. data 1 - 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 *Alert Immediate* command. data 2 - Address Format [7:4] - Address Format. 0h = VLAN ID not used with this destination 1h = 802.1q VLAN TAG [3:0] - reserved For Address Format = 1h: data 3-4 - 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)

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	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
...	

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 ~~utilize~~ 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	[7:6] - Authentication Type / Integrity Algorithm Number 00b = IPMI v1.5 AuthCode Algorithms 01b = IPMI v2.0/RMCP+ Algorithm Number 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 For [7:6] = 01b, IPMI v2.0/RMCP+ Integrity Algorithm Number [5:0] - Integrity Algorithm Number. See *Table 13-18, Integrity Algorithm Numbers.* 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.
		...
IPMI Response Data	1	Completion Code ~~If authentication type = straight password, BMC returns ‘OK’ if password was correct for specified user, or error completion code if it is not.~~

...

Revision 3 (2/15/06) Addenda, Errata, and Clarifications

E269 Addendum and Clarification - Table 22-32, Get User Access Command

The specification did not provide a mechanism to positively indicate that a particular User ID was completely enabled or disabled using the *Set User Password* command. Bits [5:4] only actually reported whether a given User ID might was presently configured to be enabled for an IPMI User authentication purpose. This is deficiency is corrected by defining bit [7] to be a bit that positively indicates the enabled/disabled state of the User ID. The description of how the "number of enabled User IDs" is determined in the description of bits [5:4] is also updated accordingly.

Table 22-31, Get User Access Command

	byte	data field
Request Data	1	[7:4] - reserved [3:0] - Channel Number
	2	[7:6] - reserved [5:0] - User ID. 000000b = reserved.
Response Data	1	Completion Code. Note: an implementation will not return an error completion code if the user access level is set higher than the privilege limit for a given channel. If it is desired to bring attention to this condition, it is up to software to check the channel privilege limits and provide notification of the mis-match.
	2	Maximum number of User IDs. 1-based. Count includes User 1. A value of 1 indicates only User 1 is supported. [7:6] - reserved [5:0] - maximum number of user IDs on this channel
	3	Count of currently enabled User IDs (1-based). A value of 0 indicates that all users, including User 1, are disabled. This is equivalent to disabling access to the channel. [7:6] - User ID Enable status (for IPMI v2.0 errata 3 and later implementations). 00b = User ID enable status unspecified. (For backward compatibility with pre-errata 3 implementations. IPMI errata 3 and later implementations should return the 01b and 10b responses.) 01b = User ID enabled via *Set User Password* command. 10b = User ID disabled via *Set User Password* command. 11b = reserved. ~~reserved~~ [5:0] - count of currently enabled user IDs on this channel (Indicates how many User ID slots are presently in use.)
	4	Count of User IDs with fixed names, including User 1(1-based). Fixed names in addition to User 1 are required to be associated with sequential user IDs starting from User ID 2. [7:6] - reserved. [5:0] - count of user IDs with fixed names on this channel

5	Channel Access [7] - reserved. [6] - 0b = user access available during call-in or callback direct connection 1b = user access available only during callback connection For pre- IPMI v2.0 errata 3 implementations: bits 5:4, following, are used for determining the 'count of currently enabled user IDs' in byte 3. Either bit being set to 1b represents an 'enabled user ID'. For IPMI v2.0 errata 3 and later implementations: The 'count of enabled User IDs' is based on the User IDs that are presently enabled as reflected in byte 3, bits [7:6], User ID Enable status. Note: Some pre- IPMI v2.0 errata 3 implementations may automatically clear bits [5:4], and may also prevent them from being set, while the User ID is disabled. IPMI v2.0 errata 3 and later implementations should not alter bits [5:4] based on whether a User ID is enabled or not. [5] - 0b = user disabled for link authentication 1b = user enabled for link authentication [4] - 0b = user disabled for IPMI Messaging 1b = user enabled for IPMI Messaging [3:0] - User Privilege Limit for given Channel 0h = reserved 1h = Callback 2h = User 3h = Operator 4h = Administrator 5h = OEM Proprietary Fh = NO ACCESS (Note: this value does not add to, or subtract from, the number of enabled user IDs)

E387 Addendum - Table 22-24, Close Session Command

This addendum is to enable using a Session Handle with the *Close Session* command. At present, there is no straightforward way to enable closing another session, although the text for the *Close Session* command clearly indicates that was intended to be supported. The *Close Session* command takes a Session ID, but since *Get Session Info* does not provide a way to obtain the session ID, there's no direct way to get the Session ID to use for closing another session. Rather than changing the *Get Session Info* command, the change is to allow the *Close Session* command to take a Session Handle, as follows:

Table 22-24, Close Session Command

	byte	data field
Request Data	1:4	Session ID. For IPMI v2.0/RMCP+ this is the Managed System Session ID value that was generated by the BMC, not the ID from the remote console. If Session ID = 0000_0000h then an implementation can optionally enable this command to take an additional byte of parameter data that allows a session handle to be used to close a session.
	(5)	Session Handle. (only present if Session ID = 0000_0000h)
Response Data	1	Completion Code 87h = invalid Session ID in request 88h = invalid Session Handle in request

E388 Typos - Section 6.13.2, Send Message Command From System Interface, and Table 28-14, Boot Option Parameters

Section 6.13.2, Send Message Command From System Interface, has incorrect command name. In the last paragraph, *Get Channel Sessions* should be *Get Session Info*, as follows:

> If the target channel uses sessions, the *Send Message* command data will require a *Session Handle* value to select which session on the channel the message will be sent to. Software can use the *Get Channel Info* and ~~*Get Channel Sessions*~~ *Get Session Info* commands to determine what channels are present and to obtain the Session Handle for a given session.

A similar typo is in Table 28-14:

Table 28-14, Boot Option Parameters

Parameter	#	Parameter Data (non-volatile unless otherwise noted)
…		
boot initiator info (semi-volatile) [1]	6	Address & Identity information for the party that initiated the boot. The party that initiates the boot writes this parameter and the boot info acknowledge parameter prior to issuing the command that causes the system power up, power cycle, or reset. This data is normally written by the remote console application, not the BMC. boot source data 1- Channel Number. Channel that will deliver the boot command (e.g. chassis control). BIOS and boot software (e.g. service partition or OS loader) can use the ~~*Get Channel Sessions*~~*Get Session Info* command to find out information about the party that initiated the boot. [7:4] - reserved [3:0] - Channel Number data 2:5 - Session ID. Session ID for session that the boot command will be issued over. This value can be used with the ~~*Get Channel Sessions*~~*Get Session Info* command to find out information about the party that initiated the boot. For IPMI v2.0/RMCP+ this is the Managed System Session ID that was generated by the BMC when the session was activated.
…		

E389 Errata - Table 24-7, Get Payload Instance Info Command

The offset bytes for the Payload-specific Information field were incorrect. This is corrected as follows:

Table 24-7, Get Payload Instance Info Command

	byte	data field
Request Data	1	Payload Type Number - Type number of the standard payload type or OEM Payload Handle to retrieve status for. See *Table 13-16, Payload Type Numbers*.
	2	Payload Instance. 1-based. 0h = reserved.
Response Data	1	Completion Code
	2:5	Session ID - ID of session that instance is presently activated on. (The Managed System Session ID that the BMC generated when the session was activated). 00_00_00_00h if given instance is not activated. Remote software can use this information with the *Get Session Info* command to identify the remote console that presently is using a given payload type.
	4:11~~6~~ :13	Payload-specific information (8-bytes) For Payload Type = SOL: Byte 1: Port Number A number representing the system serial port that is being redirected. 1-based. 0h = unspecified. Used when more than one port can be redirected on a system. Byte 2:8 = reserved.

E390 Addendum and Clarification - Table 43-13, Entity ID Codes

The specification had text that helped indicate that the Entity ID for power unit could also be used for grouping entities and sensors under a logical power domain, but the same text was not applied to the naming and description for the 'cooling unit' Entity ID.

In addition, the example was somewhat misleading in that it may have implied that power unit and cooling unit were only used for grouping physical entities. The Entity descriptions has been extended as follows:

Table 43-13, Entity ID Codes

Code		Entity
0	00h	unspecified
…		
19	13h	power unit / power domain - This Entity ID is typically used as a pre-defined logical entity for grouping power supplies and/or sensors that are associated in monitoring a particular logical power domain.
…		
30	1Eh	cooling unit / cooling domain - This Entity ID can be used as a pre-defined logical entity for grouping fans or other cooling devices and/or sensors that are associated in monitoring a particular logical cooling domain.
…		

E391 Addendum - Section 6.13.4, Bridged Request Example

This addendum loosens the specification to allow implementations to elect to either use the BMC address or the Target device address when sending the second response of a bridged request. The fifth paragraph of the section is updated as follows:

> For example, suppose a *Get Device ID* command has been encapsulated in a *Send Message* command directed to the IPMB from a LAN channel. The BMC will immediately send a response to the *Send Message* command back on LAN. The BMC will extract the encapsulated *Get Device ID* message content and format it as a *Get Device ID* request for IPMB. The target device on IPMB responds with a *Get Device ID* response message in IPMB format. The BMC takes the tracking information that was stored when the *Send Message* command was issued, and uses it to create a *Get Device ID* response in LAN format. The Responder's address information in that response ~~will~~ can either be that of the BMC, ~~not~~ or the address of the device on IPMB that the request was targeted to, at the choice of the BMC implementation. *Parties that initiate this type of bridged request using the Send Message command should accept responses from the BMC that use either address.*

E392 Addendum - Table 42-3, Sensor Type Codes

Certain busses, such as PCI-E, may operate in a degraded performance condition as a mechanism for handling error conditions such as insufficient electrical margins or higher than normal error rates. A new offset for 'Bus Degraded' has been added to the Critical Interrupt sensor type to cover busses or interconnects which may be put into a lower performance state for error handling purposes.

Table 42-3 is updated as follows:

Table 42-3, Sensor Type Codes

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
...			
Critical Interrupt	13h	00h	Front Panel NMI / Diagnostic Interrupt
		01h	Bus Timeout
		02h	I/O channel check NMI
		03h	Software NMI
		04h	PCI PERR
		05h	PCI SERR
		06h	EISA Fail Safe Timeout
		07h	Bus Correctable Error
		08h	Bus Uncorrectable Error
		09h	Fatal NMI (port 61h, bit 7)
		0Ah	Bus Fatal Error
		0Bh	Bus Degraded (bus operating in a degraded performance state)
...			

E393 Typo - Table 43-13, Entity ID Codes

The previous IPMI v2.0 Markup included a cut-and-paste error when E352 was added to the markup. The System Firmware entity type (22h) was copied in the table as Entity ID 36h. The row containing the second occurrence of the System Firmware Entity ID is deleted, as follows.

Table 43-13, Entity ID Codes

Code		Entity
0	00h	unspecified

...

~~54~~	~~36h~~	~~System Firmware (E.g. BIOS / EFI).~~

...

E394 Addendum - Table 23-2, Set LAN Configuration Parameters Command, Table 26-3, Set SOL Configuration Parameters Command, Table 30-4, Set PEF Configuration Parameters Command

Some configuration commands are missing completion codes. The completions codes are added to make the commands consistent, as follows.

Table 23-2, Set LAN Configuration Parameters Command

	byte	data field
Request Data	1	[7:4] - reserved [3:0] - Channel number.
	2	Parameter selector
	3:N	Configuration parameter data, per *Table 23-4, LAN Configuration Parameters*
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 83h = attempt to read write-only parameter

Table 26-3, Set SOL Configuration Parameters Command

	byte	data field
Request Data	1	[7:4] - reserved [3:0] - Channel number.
	2	Parameter selector
	3:N	Configuration parameter data, per *Table 26-5, SOL Configuration Parameters*
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 83h = attempt to read write-only parameter

Table 30-4, Set PEF Configuration Parameters Command

	byte	data field
Request Data	1	Parameter selector [7] - reserved [6:0] - Parameter selector
	2:N	Configuration parameter data, per *Table 24-6, PEF Configuration Parameters.*
Response Data	1	Completion Code. Generic plus the following command-specific completion codes: 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 83h = attempt to read write-only parameter

E395 Errata - Table 43-13, Entity ID Codes

The description for Entity Type 45 (2Dh) was incorrect. This is corrected as follows.

Table 43-13, Entity ID Codes

Code		Entity
…		
44	2Ch	I/O module (a module that contains the main elements of an I/O interface)
45	2Dh	Processor/ IO module (a ~~module that contains the main elements of an I/O interface~~combination processor and i/O module)
…		

E396 Addendum and Clarification on E372 "Settable Sensors"

The following additions and clarifications are made to the command in order to clarify the management controller implementation requirements, and software interaction with the command.

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 (or satellite management controller) hardware. The Type 01h and Type 02h SDRs include an optional bit that allows those records to report a sensor is settable.

The command sets the event state and data values for the sensor directly into the management controller. The management controller simply takes the parameters that are given to it and generates events based on the event state settings. The management controller is not required to autonomously update the sensor event state based on reading values. There is also no requirement for the BMC to make sure the event state and reading are in synch with one another, though an implementation is allowed to reject 'illegal' combinations.

For example, if a sensor is threshold-based, the implementation is not required to update threshold state based on the data value. Thus, software should always set the event state whenever it wants to cause events to be generated based on data that is set with this command.

Since the management controller is not required to automatically update sensor event state, this means it is not required to automatically clear or rearm event state once a given event state has been set. Therefore, if software asserts an event state using this command, it will need to issue a separate command to explicitly deassert that state before another event can be generated.

E397 Addendum - Get / Set SEL Time UTC Offset Commands

The following commands are added to enable a UTC offset that is associated with the SEL Time to be set and returned.

31.11a Get SEL Time UTC Offset

This command is used to retrieve the SEL Time UTC Offset that was set using the *Set SEL Time UTC Offset* command. See *Set SEL Time UTC Offset* command for additional information.

Table 31-11a, Get SEL Time UTC Offset Command

	byte	data field
Request Data	-	-
Response Data	1	Completion Code
	2:3	16-bit, 2s-complement signed integer for the offset in minutes from UTC to SEL Time. LS-byte first. (ranges from -1440 to 1440) 07FFh = 'unspecified'. Interpret SEL time as local time.

31.11b Set SEL Time UTC Offset

This command initializes and retrieve a UTC offset (timezone) that is associated with the SEL Time (see the *Set SEL Time* command). The offset is the number of minutes difference between the local time zone and Universal Coordinated Time (UTC).

If you know what the UTC time is, you get local time by *adding* the offset to the UTC time. To get UTC time from local time (SEL Time) you *subtract* the offset from the local time. For example, the offset for United States Pacific Standard Time is -8 (minus 8) hours. If the UTC time were 10am (10:00 hours), then Pacific Standard Time would be 2am (02:00 hours). The offset for Tokyo, Japan is +9 (plus 9), so if the UTC time were 10am, then the Tokyo, Japan time is 7pm (19:00 hours).

Note that the UTC offset varies with DAYLIGHT SAVINGS TIME. Therefore, if this is capability is used, software may be required to ensure the offset gets updated appropriately.

In order to retain backward compatibility with the original 'local only' definition of SEL Time, the UTC Offset is **automatically reset to 'unspecified'** whenever the *Set SEL Time* command is executed. Thus, the UTC Offset must be re-written after using the *Set SEL Time* command.

Table 31-11b, Set SEL Time UTC Offset Command

	byte	data field
Request Data	1:2	16-bit, 2s-complement signed integer for the offset in minutes from UTC to SEL Time. LS-byte first. (ranges from -1440 to 1440) 07FFh = 'unspecified'. Interpret SEL time as local time.
Response Data	1	Completion Code

Table 31-1, SEL Device Commands, Additions:

Table 31-1, SEL Device Commands

Command	Section	O/M
...		
Set SEL Time	31.11	M
Get SEL Time UTC Offset	31.11a	O[4]
Set SEL Time UTC Offset	31.11b	O
Get Auxiliary Log Status	31.12	O
Set Auxiliary Log Status	31.13	O[3]

1. Mandatory if multiple entities have overlapping access to the SEL. If system mechanisms or conventions are defined that preclude this operation, then this command is optional.
2. Either *Add SEL Entry* or *Partial Add SEL Entry* must be provided. Providing both is optional.
3. *Set Auxiliary Log Status* cannot be implemented without also supporting *Get Auxiliary Log Status*. However, *Get Auxiliary Log Status* is allowed to be implemented without *Set Auxiliary Log Status*.
4. Mandatory if *Set SEL Time UTC Offset* 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							
...							
Get SEL Time UTC Offset	31.11a	Storage	5Ch		X		
Set SEL Time UTC Offset	31.11b	Storage	5Dh			X	
...							

Table H-1 Additions:

Table H-1, Sub-function Number Assignments

	Sub Fn #	NetFn	CMD
...			
Get SEL Time UTC Offset		Storage	5Ch
Set SEL Time UTC Offset		Storage	5Dh
...			

E398 Addendum - Additional Completion Code for Section 35b.4, Forwarded Command Command

An additional completion code is supported as an option for the Forwarded Command command, as follows:

The BMC will *time out* and return an FFh or C3h (Timeout while processing command. Response unavailable.) 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.

Response Data	x:N	Forwarded Command Request Data
	1	Completion Code. BMC shall return FFh or C3h (Timeout while processing command. Response Unavailable) completion code if the target controller does not return a matching *Forwarded Command* response message within the timeout set by the *Enable Forwarded Commands* command.
	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.

E399 Clarification - Table 22-2, Set BMC Global Enables Command, Table 22-3, Get BMC Global Enables Command

The specification did not clearly indicate how the *Set* and *Get BMC Global Enables* commands should handle the case where the Message Buffer Full and/or Receive Message Queue interrupts are not supported by the particular BMC implementation. This is clarified as follows:

Table 22-2, Set BMC Global Enables Command

	byte	data field
Request Data	1	This field is set to xxxx_100xb on power-up and system resets. If the implementation allows the receive message queue interrupt to be enabled/disabled, the default for bit 0 should be 0b~~, otherwise it should always be 1b~~. [7] OEM 2 Enable. Generic system mgmt. software must do a 'read-modify-write' using the *Get BMC Global Enables* and *Set BMC Global Enables* to avoid altering this bit. [6] OEM 1 Enable. Generic system mgmt. software must do a 'read-modify-write' using the *Get BMC Global Enables* and *Set BMC Global Enables* to avoid altering this bit. [5] OEM 0 Enable. Generic system mgmt. software must do a 'read-modify-write' using the *Get BMC Global Enables* and *Set BMC Global Enables* to avoid altering this bit. [4] reserved [3] 1b = Enable System Event Logging (enables/disables logging of events to the SEL - with the exception of events received over the system interface. Event reception and logging via the system interface is always enabled.) SEL Logging is enabled by default whenever the BMC is first powered up. It's recommended that this default state also be restored on system resets and power on. [2] 1b = Enable Event Message Buffer. Error completion code returned if written as '1' and Event Message Buffer not supported. [1] 1b = Enable Event Message Buffer Full Interrupt [0] 1b = Enable Receive Message Queue Interrupt (this bit also controls whether KCS communication interrupts are enabled or disabled. An implementation is allowed to have this interrupt always enabled.) Note: If the Event Message Buffer Full or Receive Message Queue interrupt are not supported, an implementation can elect to return a CCh error completion code for the *Set BMC Global Enables* command if an attempt is made to enable the interrupt (this is the recommended implementation). Alternatively, the implementation can accept the command, but must return 0b for the corresponding bit in the *Get BMC Global Enables*.
Response Data	1	Completion Code.

	byte	data field
Request Data	-	-
Response Data	1	Completion Code
	2	[7] - 1b = OEM 2 Enabled. [6] - 1b = OEM 1 Enabled. [5] - 1b = OEM 0 Enabled. [4] - reserved [3] - 1b = System Event Logging Enabled [2] - 1b = Event Message Buffer Enabled [1] - 1b = Event Message Buffer Full Interrupt Enabled [0] - 1b = Receive Message Queue Interrupt Enabled (this bit also indicates whether KCS communication interrupt are enabled or disabled.) Note: If the Receive Message Queue and/or Event Message Full interrupts are not implemented the corresponding Interrupt Enabled status bit should always be returned as 0b.

E400 Addendum - Writable SDRs Optional

This addendum extends the specification to support implementations that only support a 'read-only' SDR Repository. That is, an SDR Repository that cannot be written to using standard IPMI commands. This is covered by changes to SDR Repository -related sections and tables, as follows:

Table 3-1, Required BMC Functions

Function	M/O	Description
IPM Device	M	The BMC must implement the mandatory IPM Device commands. If an IPMB is provided, the mandatory commands must be accessible from the IPMB unless otherwise noted.
System Interface	M	The implementation must provide BMC access via one of the specified IPMI system interfaces.
SDR Repository	M	The BMC must provide a SDR Repository to hold Sensor, Device Locator, and Entity Association records for all sensors in the platform management subsystem. This does not need to include SDRs for sensors that only generate events. If the SDR Repository is writable, it~~.~~ It is recommended that at least 20% additional space is provided for add-in platform management extensions. The SDR Repository must be accessible via the system interface. If an IPMB is provided, the SDR Repository must be readable via that interface as well. SDR update via the IPMB interface is optional. SDR Repository access when the system is powered up or in ACPI 'S1' sleep is mandatory, but access when the system is powered-down or in a >S1 sleep state is optional.

…

33. SDR Repository

This section describes the logical SDR Repository Device, and the commands that are used to access the SDR Repository. This section also describes a companion set of functionality, the Internal Sensor Initialization Agent, that is part of a system that implements this platform and sensor instrumentation specification.

…

SDRs are kept in a single, centralized Sensor Data Record Repository to simplify the ability for out-of-band applications to get information about the platform management subsystem. This eliminates the need for out-of-band applications, which may be over slow transports, to perform discovery actions. It also is a better mechanism to ensure that the information actually represents what's supposed to be in the system, instead of just what was discovered.

The SDR Repository implementation can be writable via standard IPMI commands, or it can be 'read-only'. Supporting a writable SDR Repository provides a common way to support adding Sensor Data Records for 3rd party add-in devices and sensors, such as sensors provided by satellite management controllers on IPMB, Depending on the sensor implementation, writable SDRs can also be used to provide a non-volatile mechanism for changing the default behavior of sensors, such as whether they are scanning or generate events and what thresholds they use.

…

33.2 Modal and Non-modal SDR Repositories

There are two possible writable SDR Repository implementations: modal and non-modal. A non-modal SDR Repository can be written to at any time. Writing to the SDR does not impact the operation of other commands in the management controller.

…

33.3 Populating the SDR Repository

Most systems are fundamentally static with respect to their platform management configuration once the system integrator has put the system together. Thus, the typical model for ~~the~~ an implementation that supports a writable SDR Repository is that it is manually updated using a utility or other piece of software if the platform management configuration is changed in the field.

For example, suppose a system could be upgraded to accept a new RAID backplane that had extra fans and temperature sensors. Part of the upgrade process would be to run a utility, supplied by the system integrator, that updated the SDR Repository with the new SDRs.

33.7 SDR Repository Device Commands

The following sections describe the commands that an SDR Repository Device provides for accessing the SDR Repository.

….

Refer to *Appendix G - Command Assignments* for the specification of the Network Function and Command (CMD) values and privilege levels for these commands.

Table 33-2, SDR Repository Device Commands

Command	Section	O/M
Get SDR Repository Info	33.9	M
Get SDR Repository Allocation Info	33.10	O
Reserve SDR Repository	33.11	M
Get SDR	33.12	M[5]
Add SDR	33.13	M[1]
Partial Add SDR	33.14	M[1][5]
Delete SDR	33.15	O[5]
Clear SDR Repository	33.16	M[5]
Get SDR Repository Time	33.17	O/M[2]
Set SDR Repository Time	33.18	O/M[2]
Enter SDR Repository Update Mode	33.19	O[3]
Exit SDR Repository Update Mode	33.20	O[3]
Run Initialization Agent	33.21	O[4]

1. If a writable Sensor Data Record Repository is implemented, e~~E~~ither *Add SDR* or *Partial Add SDR* command must be provided via the system interface. Providing both via the system interface is optional. For the IPMB, the *Add SDR* and *Partial Add SDR* commands are optional.
2. If the SEL Device and SDR Repository Device are implemented in separate controllers, then both these commands are Mandatory for the SDR Repository Device. If the SDR Repository Device shares the same controller as the SEL Device (This is normally indicated in the IPM Device Support field of the Get Device ID command), then the SDR device uses the SEL Device's Timestamp Clock. In this case, the Get SDR Repository Time command is optional, and the Set SDR Repository Time command is *not used*.
3. Support for both these commands is mandatory ~~for~~ if a modal SDR Repository is implemented. The *Enter SDR Repository Update Mode* command is mandatory when in 'operational' mode, while the *Exit SDR Repository Update Mode* is mandatory when in 'update' mode.
4. Highly recommended. This supports utilities that can update the SDRs during run-time. Without this, a system reset will need to be performed to cause the initialization agent to run.
5. Mandatory if writable Sensor Data Record Repository is implemented. A reservation field of 0000h is passed to these commands when in SDR Repository Update Mode.

E401 Addendum and Clarification - Table 22-25, Get AuthCode Command

The description of bytes 2:21 in response for IPMI v2.0 Integrity Algorithm Numbers was ambiguous. The "2:21" field size notation, which by convention would indicate a fixed-length field, was in conflict with the statement that the returned data could be "up to 20 bytes". The text 'up to 20 bytes' should override the "2:21" designation, since a statement of 'up to' would not normally be included for a fixed-length field. Furthermore, if it was intended that integrity hash data was to be returned as a 20-byte field, a description of how shorter integrity hashes would be padded to 20-bytes would be needed.

Thus, the specification is being clarified to allow for the variable length response by changing the field byte designation to "(2:21)" to indicate a possible variable-length return. In addition, in order to attempt to minimize any potential issues with existing implementations that may be returning a fixed length field, the specification is being amended to indicate that the field returned by the implementation can either be variable length or padded with 0's to 20 bytes (since '0' padding is the typical padding used in IPMI).

Table 22-25, Get AuthCode Command

	byte	data field
IPMI Request Data	1	[7:6] - Authentication Type / Integrity Algorithm Number …
IPMI Response Data	1	Completion Code
		For IPMI v1.5 AuthCode Number:
	2:17	AuthCode = See *22.17.1, AuthCode Algorithms*.
		For IPMI v2.0 Integrity Algorithm Number
	(2:21)	Resultant hash, per selected Integrity algorithm. Up to 20 bytes. An implementation can elect to return a variable length field based on the size of the hash for the given integrity algorithm, or can return a fixed field where the hash data is followed by 00h bytes as needed to pad the data to 20 bytes.

E402 Addendum - Table 42-3, Sensor Type Codes

Newer memory modules can have provisions for built-in temperature monitoring that reports a critical over-temperature state for the device. This addendum extends the sensor-specific offsets for the Memory sensor type to allow this state to be reported without having to implement a separate temperature sensor associated with the memory device.

Table 42-3, Sensor Type Codes

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
reserved	00h	-	reserved
		…	
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
		…	…
		09h	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.)
		0Ah	Critical Overtemperature. Memory device has entered a critical overtemperature state, exceeding specified operating conditions. Memory devices in this state may produce errors or become inaccessible.

E403 Addendum - Table 23-4, LAN Configuration Parameters

Errata/clarification to allow the MAC address to not be writable via the *Set LAN Configuration Parameters* command. [I.e. Allow the MAC address to be read only].

Table 23-4, LAN Configuration Parameters

…

MAC Address (can be Read Only)	5	data 1:6 - MAC Address for messages transmitted from BMC. MS-byte first. An implementation can either allow this parameter to be settable, or it can be implemented as Read Only.

…

E404 Addendum and Clarification - Section 1.2, Reference Documents, 20.8, Get Device GUID Command, and 22.14, Get System GUID Command

The definition of GUID for IPMI referenced the Wired-for-management specifications. These specifications are no longer readily available. This addendum changes that reference from the Wired-for-management specifications to the matching GUID definition which is documented in RFC4122. The Reference Documents section and the command descriptions are updated as follows:

1.2 Reference Documents

The following documents are companion and supporting specifications for IPMI and associated interfaces:

…

[RFC 4122] *RFC 4122, A Universally Unique IDentifier (UUID) URN Namespace, P Leach, Microsoft; M. Mealling, Refactored Networks, LL; and R. Salz, DataPower Technology, Inc.; July 2005*

…

~~[WFM] *Wired for Management Baseline Version 2.0 Release*, ©1998, Intel Corporation. Attachment A, UUIDs and GUIDs, provides information specifying the formatting of the IPMI Device GUID and FRU GUID and the System Management BIOS (SM BIOS) UUID unique IDs.~~

20.8 Get Device GUID Command

This command returns a ~~Globally Unique ID~~ GUID (Globally Unique ID), ~~(GUID)~~also referred to as a UUID (Universally Unique IDentifier), for the management controller. The format of the ID follows ~~that~~ the octet format specified in ~~the Attachment A of the *Wired for Management Baseline, Version 2.0* specification~~[RFC 4122]. [RFC4122] specifies four different versions of UUID formats and generation algorithms suitable for use for a Device GUID in IPMI. These are version 1 (0001b) "time based", and three 'name-based' versions: version 3 (0011b) "MD5 hash", version 4 (0100b) "Pseudo-random", and version 5 "SHA1 hash". The version 1 format is recommended. However, versions 3, 4, or 5 formats are also allowed. A Device GUID should never change over the lifetime of the device.

…

Note that the individual fields within the GUID are stored **least-significant byte first**, and in the order illustrated in the following table. This is the reverse of convention described in ~~the Wired for Management Specification~~[RFC4122] where GUID bytes are transmitted in 'network order' (most-significant byte first) starting with the time low field.

…

22.14 Get System GUID Command

This optional, though highly recommended, command can be used to return a GUID (Globally Unique ID), ~~(GUID)~~also referred to as a UUID (Universally Unique IDentifier), for the managed system to support the remote discovery process and other operations. The format of the ID follows the octet format specified in [RFC 4122]. [RFC4122] specifies four different versions of UUID formats and generation algorithms suitable for use for a GUID in IPMI. These are version 1 (0001b) "time based", and three 'name-based' versions: version 3 (0011b) "MD5 hash", version 4 (0100b) "Pseudo-random", and version 5 "SHA1 hash". At present [SMBIOS] does not specify a particular specification or version for UUID generation. In general, if this remains unspecified the version 1 format is recommended by the IPMI Specification for new system implementations. However, versions 3, 4, or 5 formats are also allowed. A System GUID should not change over the lifetime of the system.

If the BMC is on a removable card that can be moved to another system, the vendor of the card or system vendor should provide a mechanism for generating a new System GUID or retrieving the SMBIOS UUID from the given system.

Since the GUID is typically 'permanently' assigned to a system, an interface that would allow the GUID to be configured or changed is not specified. For systems that support [SMBIOS] the System GUID that is returned by the BMC should match the UUID field value in the SMBIOS System Information (Type 1) record.

The session header (Session Request data and Session Response Data) values shown in the following table illustrate the values that would be used to execute a *Get System GUID* Command outside of an active session. The Get System GUID will always be accepted outside of an active session. The *Get System GUID* command can also be executed ~~in~~ within the context of an active session (providing the user is operating at higher than 'Callback' privilege). When the *Get System GUID* command is executed in the context of an active session, the session header fields must have correct values according to the authentication, Session ID, and session sequence number information that was negotiated for the session.

Table 22-16, Get System GUID Command

Session Request Data		authentication type = NONE
		session seq# = null (0's)
		Session ID = null (0's)
		AuthCode = NOT PRESENT
Request Data	-	-
Session Response Data		authentication type = NONE
		session seq# = null (0's)
		Session ID = null (0's)
		AuthCode = NOT PRESENT
Response Data	1	Completion Code
	2:17	GUID bytes 1 through 16. See *Table 20-10, GUID Format*.

E405 Addendum - Section 1.2, Reference Documents

The reference for SMBIOS is out-of-date. The reference is updated with the latest, published, DMTF SMBIOS specification, as follows:

[SMBIOS] System Management BIOS (SMBIOS) Reference Specification, Version 2.4, July 21, 2004. Copyright © "2000, 2002, 2004" Distributed Management Task Force, Inc. (DMTF). All rights reserved.

~~[SMBIOS] *System Management BIOS Specification, Version 2.3.1*, © 1997, 1999 American Megatrends Inc., Award Software International, Compaq Computer Corporation, Dell Computer Corporation, Hewlett Packard Company, Intel Corporation, International Business Machines Corporation, Phoenix Technologies Limited, and SystemSoft Corporation.~~

E406 Clarification - Table 22-15, Get Channel Authentication Capabilities Command

If IPMI v2.0+ extended data is requested for an authenticated channel, and the channel does not support RMCP+ (e.g. a serial channel), then the Response data should return with bit [5] of byte 4 = 0b and bytes 5:9 = 00h.

Table 22-15, Get Channel Authentication Capabilities Command

Session Request Data		authentication type = NONE / payload type = IPMI Message
		session seq# = null (0's)
		Session ID = null (0's)
		AuthCode = NOT PRESENT
IPMI Request Data	1	Channel Number [7] - 1b = get IPMI v2.0+ extended data. If the given channel supports authentication but does not support RMCP+ (e.g. a serial channel), then the Response data should return with bit [5] of byte 4 = 0b, byte 5 should return 01h, 0b = Backward compatible with IPMI v1.5. ~~Result~~ Response data only returns bytes 1:9, bit [7] of byte 3 (Authentication Type Support) and bit [5] of byte 4 returns as 0b, and byte 5 returns 00h. [6:4] - reserved [3:0] - channel number. 0h-7h, Fh = channel numbers Eh = retrieve information for channel this request was issued on.
		...
	4	[7:6] - reserved [5] - KG status (two-key login status). Applies to v2.0/RMCP+ RAKP Authentication only. Otherwise, ignore as 'reserved'. 0b = KG is set to default (all 0's). User key KUID will be used in place of KG in RAKP. (Knowledge of KG not required for activating session.) 1b = KG is set to non-zero value. (Knowledge of both KG and user password (if not anonymous login) required for activating session.) Following bits apply to IPMI v1.5 and v2.0: …
	5	For IPMI v1.5: - reserved For IPMI v2.0+: - Extended Capabilities [7:2] -reserved [1] - 1b = channel supports IPMI v2.0 connections. [0] - 1b = channel supports IPMI v1.5 connections.
	6:8	OEM ID IANA Enterprise Number for OEM/Organization that specified the particular OEM Authentication Type for RMCP. Least significant byte first. Return 00h, 00h, 00h if no OEM authentication type available.
	9	OEM auxiliary data. Additional OEM-specific information for the OEM Authentication Type for RMCP. Return 00h if no OEM authentication type available.

E407 Addendum - Table 13-15, RMCP+ and RAKP Message Status Codes

This addendum is to allow implementations to return an 'insufficient resources to create a session' status to be returned for other types of RAKP messages in addition to the RMCP+ Open Session Response

message. This is because the BMC could run out of resources during other phases of session establishment. For example, in some implementations may have limited resources for tracking sessions that are in the process of being established. Thus, it could be possible to run out of resources if multiple endpoints are in the middle of using RAKP to establish sessions, particularly if errors cause a party to restart session establishment without being able to complete and close the previous attempt. This updates Table 13-5 as follows:

Table 13-15, RMCP+ and RAKP Message Status Codes

Status Code	Description	Message			
		RMCP+ Open Session Response	RAKP Msg 2	RAKP Msg 3	RAKP Msg 4
00h	No errors	X	X	X	X
01h	Insufficient resources to create a session	X	X	X	X
02h	Invalid Session ID	X	X	X	X

…

E408 Addendum and Clarification - Table 13-10, RMCP+ Open Session Response, RAKP Message 3 error handling

It is possible that an invalid role could be passed in the RMCP+ Open Session Request. Therefore, the 'Invalid role' status code is being added to the list of possible status codes returned for the RMCP+ Open Session Response message. Table 13-5 is updated as follows.

In addition, it was noted that in some cases references were made to 'closing the session' in the *Open Session* and RAKP messages

Table 13-10, RMCP+ Open Session Response

	byte	data field
IPMI Session Header		Payload Type = RMCP+ Open Session Response Session ID = 00_00_00_00h Session Sequence Number = 00_00_00_00h
IPMI Payload	1	Message Tag - The BMC returns the Message Tag value that was passed by the remote console in the *Open Session Request* message.
	2	RMCP+ Status Code - Identifies the status of the previous message. If the previous message generated an error, then only the Status Code, Reserved, and Remote Console Session ID fields are returned. See *Table 13-15, RMCP+ and RAKP Message Status Codes*. The ~~session is then immediately closed~~ session establishment in progress is discarded at the BMC, and the remote console will need to start over with a new *Open Session Request* message. (Since the BMC has not yet delivered a Managed System Session ID to the remote console, it shouldn't be carrying any state information from the prior *Open Session Request*, but if it has, that state should be discarded.)

Table 13-15, RMCP+ and RAKP Message Status Codes

Status Code	Description	Message			
		RMCP+ Open Session Response	RAKP Msg 2	RAKP Msg 3	RAKP Msg 4
	...				
09h	Invalid role	X	X		

Table 13-13, RAKP Message 3

	byte	data field
IPMI Session Header		Payload Type = RAKP Message 3 Session ID = 00_00_00_00h Session Sequence Number = 00_00_00_00h
IPMI Payload	1	Message Tag - Selected by remote console. Used by remote console to help match responses up with requests. In this case, the corresponding *RAKP Message 4* that is returned by the BMC. The BMC can use this value to help differentiate retried messages from new messages from the remote console.
	2	RMCP+ Status Code Identifies the status of the previous message. If the previous message generated an error, then only the Completion Code, Reserved, and Managed System Session ID fields are returned. If the BMC receives an error from the remote console, it will immediately ~~close the session~~terminate the RAKP exchange in progress, and will not respond with an RAKP Message 4, even if the remaining parameters and Key Exchange Authentication code (below) ~~is~~ are valid. (Terminating the RAKP exchange in progress means that the BMC will require the remote console to restart the RAKP authentication process starting with RAKP Message 1.) See *Table 13-15, RMCP+ and RAKP Message Status Codes* for the status codes defined for this message.
	...	
	9:N	Key Exchange Authentication Code An integrity check value over the relevant items specified by the RAKP algorithm for Message 3. The size of this field depends on the specific Authentication Algorithm that was selected when the session was created. This field may be 0 bytes (absent) for some algorithms (e.g. RAKP-none). Note that if the authentication algorithm for the given Requested Maximum Privilege Level/Role specifies (e.g. RAKP-none) specifies 'no Authentication Code' then this field must be absent to be considered a match for the algorithm.

E409 Addendum - Additional IPMI Channel Number support

This addendum extends the channels number definition to support four additional channels with the BMC.

Changing the spec to support more channel numbers mainly involves changing a portion of the 'reserved' channel numbers to 'implementation specific' in Table 6-1, changing comments in some of the commands that use channel numbers, and extending the Firmware Firewall sub-function enables to include additional channels. The addition also extends the *Enable Forwarded Commands* command to support more channels, and extends the possible number of command sub-functions in the Firmware Firewall commands that deal with sub-function reporting and configuration.

Table 6-1, Channel Number Assignments

Channel Number	Type/Protocol	Description
0	Primary IPMB	Channel 0 is assigned for communication with the primary IPMB. IPMB protocols are used for IPMI messages. Refer to [IPMB] for more information.
1-~~7~~Bh	Implementation-specific	Channels 1-7 can be assigned to different types types of communication media and protocols for IPMI messages (e.g. IPMB, LAN, ICMB, etc.), based on the system implementation. For IPMI 1.5, 'Channel Protocol Type' and 'Channel Medium Type' numbers identify the channel's protocol and medium, respectively. Software can use the *Get Channel Info* command to retrieve this information.
~~8h~~ Ch-Dh	-	Reserved
Eh	Present I/F	The value Eh is used as a way to identify the current channel that the command is being received from. For example, if software wants to know what channel number it's presently communicating over, it can find out by issuing a *Get Channel Info* command for channel E.
Fh	System Interface	Channel 'F' is assigned for routing messages to the system interface.

Table 21-2, Get NetFn Support Command

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-~~7h~~Bh, Fh = channel numbers Eh = retrieve information for channel this request was issued on.

Table 21-3, Get Command Support Command

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-~~7h~~Bh, Fh = channel numbers Eh = retrieve information for channel this request was issued on.

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

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-~~7h~~Bh, Fh = channel numbers Eh = retrieve information for channel this request was issued on.

Table 21-5, Get Configurable Commands Command

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-~~7h~~Bh, Fh = channel numbers Eh = retrieve information for channel this request was issued on.

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

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-~~7h~~Bh, Fh = channel numbers Eh = retrieve information for channel this request was issued on.
		…
IPMI Response Data	1	Completion Code
	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. The bit offset corresponds to the number of the sub-function. See *Table H-1, Sub-function Number Assignments*. 1b indicates that the sub-function can be enabled/disabled. 0b indicates that the sub-function is not configurable, or is unavailable. 0b is also used for unspecified/reserved sub-function numbers. [31] - bit for sub-function 31. [30] - bit for sub-function 30. … [1] - bit for sub-function 1. [0] - bit for sub-function 0.
	(6:9)	These additional 32-bits, if present, form a bitfield where each bit indicates support for a particular sub-function for the given command, starting from sub-function 32. The bit offset corresponds to the number of the sub-function. See *Table H-1, Sub-function Number Assignments*. These bytes are not required to be returned unless the particular command has sub-functions number definitions >31. Software should assume that an implementation may return these bytes for any command, if the particular command does not have any sub-function numbers >31 specified. 1b indicates that the sub-function can be enabled/disabled. 0b indicates that the sub-function is not configurable, or is unavailable. 0b is also used for unspecified/reserved sub-function numbers. [31] - bit for sub-function 63. [30] - bit for sub-function 62. … [1] - bit for sub-function 33. [0] - bit for sub-function 32.

Table 21-7, Set Command Enables Command

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-~~7h~~Bh, Fh = channel numbers Eh = retrieve information for channel this request was issued on.

Table 21-8, Get Command Enables Command

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-~~7h~~Bh, Fh = channel numbers Eh = retrieve information for channel this request was issued on.

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

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-~~7h~~Bh, Fh = channel numbers Eh = retrieve information for channel this request was issued on.
		…
	5:8	Sub-Function Enables (ls-byte first). The enable/disable settings are *non-volatile* 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 ~~sixteen~~ 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. 1b enables the sub-function 0b disables the sub-function. 0b is also used for un-configurable/reserved sub-function numbers. See *Table H-1, Sub-function Number Assignments*. [31] - bit for sub-function 31. [30] - bit for sub-function 30. … [1] - bit for sub-function 1. [0] - bit for sub-function 0.
	(9:12)	These additional 32-bits, if present, form a bitfield where each bit indicates support for a particular sub-function for the given command, starting from sub-function 32. The bit offset corresponds to the number of the sub-function. Software only needs to send these bytes in the request if it is setting the configuration for sub-functions 32 or higher. Note Software should be prepared that that earlier implementations (pre- errata 3) may return an error completion code if these additional bytes are sent. In general, software should avoid sending these additional bytes unless it knows (e.g. via the *Get Configurable Command Sub-Functions* command) that the given command supports sub-functions >31. 1b enables the sub-function 0b disables the sub-function. 0b is also used for un-configurable/reserved sub-function numbers. See *Table H-1, Sub-function Number Assignments*. [31] - bit for sub-function 63. [30] - bit for sub-function 62. … [1] - bit for sub-function 33. [0] - bit for sub-function 32.
IPMI Response Data	1	Completion Code Generic, plus following command-specific completion codes: 80h = attempt to enable an unsupported or un-configurable sub-function.

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

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-~~7h~~Bh, Fh = channel numbers Eh = retrieve information for channel this request was issued on.
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 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 *Table H-1, Sub-function Number Assignments*. 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.
	(6:9)	These additional 32-bits, if present, form a bitfield where each bit indicates support for a particular sub-function for the given command, starting from sub-function 32. The bit offset corresponds to the number of the sub-function. See *Table H-1, Sub-function Number Assignments*. These bytes are not required to be returned unless the particular command has sub-functions number definitions >31. Software should assume that an implementation may return these bytes for any command, if the particular command does not have any sub-function numbers >31 specified. 1b sub-function is enabled 0b sub-function is disabled or is un-configurable/reserved. [31] - bit for sub-function 63. [30] - bit for sub-function 62. … [1] - bit for sub-function 33. [0] - bit for sub-function 32.

Table 21-11, Get OEM NetFn IANA Support Command

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-~~7h~~Bh, Fh = channel numbers Eh = retrieve information for channel this request was issued on.

Table 22-15, Get Channel Authentication Capabilities Command

Session Request Data		authentication type = NONE / payload type = IPMI Message
		session seq# = null (0's)
		Session ID = null (0's)
		AuthCode = NOT PRESENT
IPMI Request Data	1	Channel Number [7] - 1b = get IPMI v2.0+ extended data. 0b = Backward compatible with IPMI v1.5. Result data only returns bytes 1:9, bit [7] of byte 3 (Authentication Type Support) and bit [5] of byte 4 return as 0b, bit [5] or byte byte 5 returns 00h. [6:4] - reserved [3:0] - channel number. 0h-~~7h~~Bh, Fh = channel numbers Eh = retrieve information for channel this request was issued on.

Table 22-17, Get Channel Cipher Suites Command

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-~~7h~~Bh, Fh = channel numbers Eh = retrieve information for channel this request was issued on.

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. 0h-~~7h~~ Bh = channel numbers ~~08h~~0Ch-0Fh = reserved
	...	
	7:8	Source Channel Support bitfield indicating which channel numbers are available for use for Command Forwarding sources. *The implementation must allow all supported source channels for command forwarding to be enabled and used for command forwarding simultaneously.* byte 1: [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 byte 2: [7] - 1b = channel Fh (system interface) supported for

	Command Forwarding. [6:~~0~~4] - reserved [3] - 1b = channel Bh supported for Command Forwarding [2] - 1b = channel Ah supported for Command Forwarding [1] - 1b = channel 9 supported for Command Forwarding [0] - 1b = channel 8 supported for Command Forwarding
9:10	Target Channel Support bitfield indicating which channel numbers are available for selection as the target channel for forwarded commands. *Note:* • *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.* byte 1: [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 byte 2: ~~reserved~~ [7:4] - reserved [3] - 1b = channel Bh supported for Command Forwarding [2] - 1b = channel Ah supported for Command Forwarding [1] - 1b = channel 9 supported for Command Forwarding [0] - 1b = channel 8 supported for Command Forwarding

Table H-1, Sub-function Number Assignments

Enable Message Channel Receive		App	32h
reserved / unspecified	0		
Channel 1 enable/disable	1		
Channel 2 enable/disable	2		
Channel 3 enable/disable	3		
Channel 4 enable/disable	4		
Channel 5 enable/disable	5		
Channel 6 enable/disable	6		
Channel 7 enable/disable	7		
Channel 8 enable/disable	8		
Channel 9 enable/disable	9		
Channel Ah enable/disable	10		
Channel Bh enable/disable	11		

...

Send Message		App	34h
Send to channel 0	0		
Send to channel 1	1		
Send to channel 2	2		
Send to channel 3	3		
Send to channel 4	4		
Send to channel 5	5		

Send to channel 6	6		
Send to channel 7	7		
Send to channel 8	8		
Send to channel 9	9		
Send to channel Ah	10		
Send to channel Bh	11		

...

Close Session		App	3Ch
reserved / unspecified	0		
Close Channel 1	1		
Close Channel 2	2		
Close Channel 3	3		
Close Channel 4	4		
Close Channel 5	5		
Close Channel 6	6		
Close Channel 7	7		
Close Channel 8	8		
Close Channel 9	9		
Close Channel Ah	10		
Close Channel Bh	11		

...

Set Channel Access		App	40h
Change configuration for channel 0	0		
Change configuration for channel 1	1		
Change configuration for channel 2	2		
Change configuration for channel 3	3		
Change configuration for channel 4	4		
Change configuration for channel 5	5		
Change configuration for channel 6	6		
Change configuration for channel 7	7		
Change configuration for channel 8	8		
Change configuration for channel 9	9		
Change configuration for channel Ah	10		
Change configuration for channel Bh	11		

...

Master Write-Read		App	52h
reserved / unspecified	0		
Access to public bus, channel 1	1		
Access to public bus, channel 2	2		
Access to public bus, channel 3	3		
Access to public bus, channel 4	4		
Access to public bus, channel 5	5		
Access to public bus, channel 6	6		
Access to public bus, channel 7	7		
Access to private bus 0	8		
Access to private bus 1	9		
Access to private bus 2	10		
Access to private bus 3	11		
Access to private bus 4	12		
Access to private bus 5	13		
Access to private bus 6	14		
Access to private bus 7	15		
Access to public bus, channel 8	16		
Access to public bus, channel 9	17		
Access to public bus, channel Ah	18		
Access to public bus, channel Bh	19		

...

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		
Alert to Channel 8	9		
Alert to Channel 9	10		
Alert to Channel Ah	11		
Alert to Channel Bh	12		

...

LAN Device Commands			
Set LAN Configuration Parameters		Transport	01h
reserved / unspecified	0		
Set for channel 1	1		
Set for channel 2	2		
Set for channel 3	3		
Set for channel 4	4		
Set for channel 5	5		
Set for channel 6	6		
Set for channel 7	7		
The following sub-function enables apply across each channel for which 'Set' has been enabled:			
Write parameters 3, 4, 6, 7, 12-15 (IP Address, IP Address Source, Subnet Mask, IPv4 Header Parameters, Default Gateway Address, Default Gateway MAC Address, Backup Gateway Address, Backup Gateway MAC Address)	8		
Write parameter 5 (MAC Address)	9		
Write parameters 8, 9 (Primary & Secondary RMCP Port)	10		
Write parameter 10, 11 (Gratuitous ARP control, Gratuitous ARP interval)	11		
Write Parameter 16 (Community String)	12		
Write Parameter 18 (Destination Type) - volatile	13		
Write Parameter 18 (Destination Type) - non-volatile	14		
Write Parameter 19 (Destination Addresses) - volatile	15		
Write Parameter 19 (Destination Addresses) - non-volatile	16		
Write Parameter 20 (802.1q VLAN ID)	17		
Write Parameter 21 (802.1q Priority)	18		
Write Parameter 24 (RMCP+ Messaging Cipher Suite Privilege Levels)	19		
Write OEM Parameters	20		
Set for channel 8	21		
Set for channel 9	22		
Set for channel Ah	23		
Set for channel Bh	24		

...

Suspend BMC ARPs		Transport	03h
reserved / unspecified	0		
ARP Response for channel 1	1		
ARP Response for channel 2	2		
ARP Response for channel 3	3		
ARP Response for channel 4	4		
ARP Response for channel 5	5		
ARP Response for channel 6	6		

ARP Response for channel 7	7		
reserved / unspecified	8		
Gratuitous ARP for channel 1	9		
Gratuitous ARP for channel 2	10		
Gratuitous ARP for channel 3	11		
Gratuitous ARP for channel 4	12		
Gratuitous ARP for channel 5	13		
Gratuitous ARP for channel 6	14		
Gratuitous ARP for channel 7	15		
ARP Response for channel 8	16		
ARP Response for channel 9	17		
ARP Response for channel Ah	18		
ARP Response for channel Bh	19		
Gratuitous ARP for channel 8h	20		
Gratuitous ARP for channel 9h	21		
Gratuitous ARP for channel Ah	22		
Gratuitous ARP for channel Bh	23		

...

Set Serial/Modem Configuration		Transport	10h
reserved / unspecified	0		
Set for channel 1	1		
Set for channel 2	2		
Set for channel 3	3		
Set for channel 4	4		
Set for channel 5	5		
Set for channel 6	6		
Set for channel 7	7		
The following sub-function enables apply across each channel for which 'Set' has been enabled:			
Write Parameter 2 (Authentication Type Enables)	8		
Write Parameter 3 (Connection Mode)	9		
Write Parameters 4 & 6 (Session Inactivity Timeout, Session Termination)	10		
Write Parameter 5 (Channel Callback Control)	11		
Write Parameter 7 (IPMI Messaging Comm Settings)	12		
Write Parameter 8 (Mux Switch Control)	13		
Write Parameters 9, 10, 11, 12, & 13 (Modem Ring Time, Modem Init String, Modem Escape Sequence, Modem Hang-up Sequence, Modem Dial Command)	14		
Write Parameters 14 & 18 (Page Blackout Interval, Call Retry Interval)	15		
Write Parameter Community String 15	16		
Write Parameters 17, 19, 21, 23 [Destination Info (volatile), Destination Comm Settings (volatile), Destination Dial Strings (volatile), Destination IP Addresses (volatile)]	17		
Write Parameters 17, 19, 21, 23 [Destination Info (non-volatile), Destination Comm Settings (non-volatile), Destination Dial Strings (non-volatile), Destination IP Addresses (non-volatile)]	18		
Write Parameters 25, 26, 27, & 28 (TAP Account, TAP Passwords, TAP Pager ID Strings, TAP Service Settings)	19		
Write Parameter 29 (Terminal Mode Configuration)	20		
Write Parameters 30, 33, 35, 36, & 48 (PPP Protocol Options, PPP Link Authentication, PPP ACCM, PPP Snoop ACCM, PPP Remote Console IP Address)	21		
Write Parameters 31 & 32 (PPP Primary RMCP Port Number, PPP Secondary RMCP Port Number)	22		
Write Parameter 34 (CHAP Name)	23		
Write Parameter 45 (PPP UDP Proxy IP Header)	24		
Write Parameters 38-44 - volatile (Account 0) (PPP Account Dial	25		

String Selector, PPP Account IP Addresses / BMC IP Address, PPP Account User Names, PPP Account User Domains, PPP Account User Passwords, PPP Account Authentication Settings, PPP Account Connection Hold Times)			
Write Parameters 38-44 - non-volatile (Account 1) (PPP Account Dial String Selector, PPP Account IP Addresses / BMC IP Address, PPP Account User Names, PPP Account User Domains, PPP Account User Passwords, PPP Account Authentication Settings, PPP Account Connection Hold Times)	26		
Write Parameters 38-44 - non-volatile (Accounts 2-n) (PPP Account Dial String Selector, PPP Account IP Addresses / BMC IP Address, PPP Account User Names, PPP Account User Domains, PPP Account User Passwords, PPP Account Authentication Settings, PPP Account Connection Hold Times)	27		
Write Parameter 49 (System Phone Number)	~~20~~28		
Write OEM Parameters	~~21~~29		
Set for channel 8	30		
Set for channel 9	31		
Set for channel Ah	32		
Set for channel Bh	33		

...

Set PPP UDP Proxy Transmit Data		Transport	14h
reserved / unspecified	0		
Set for channel 1	1		
Set for channel 2	2		
Set for channel 3	3		
Set for channel 4	4		
Set for channel 5	5		
Set for channel 6	6		
Set for channel 7	7		
Set for channel 8	8		
Set for channel 9	9		
Set for channel Ah	10		
Set for channel Bh	11		

...

Send PPP UDP Proxy Packet		Transport	16h
reserved / unspecified	0		
Send for channel 1	1		
Send for channel 2	2		
Send for channel 3	3		
Send for channel 4	4		
Send for channel 5	5		
Send for channel 6	6		
Send for channel 7	7		
Send for channel 8	8		
Send for channel 9	9		
Send for channel Ah	10		
Send for channel Bh	11		

...

Callback		Transport	19h
reserved / unspecified	0		
Callback using channel 1 parameters	1		
Callback using channel 2 parameters	2		
Callback using channel 3 parameters	3		
Callback using channel 4 parameters	4		
Callback using channel 5 parameters	5		
Callback using channel 6 parameters	6		
Callback using channel 7 parameters	7		

Callback using channel 8 parameters	8		
Callback using channel 9 parameters	9		
Callback using channel Ah parameters	10		
Callback using channel Bh parameters	11		

...

Set User Callback Options		Transport	1Ah
reserved / unspecified	0		
Set for channel 1	1		
Set for channel 2	2		
Set for channel 3	3		
Set for channel 4	4		
Set for channel 5	5		
Set for channel 6	6		
Set for channel 7	7		
Set for channel 8	8		
Set for channel 9	9		
Set for channel Ah	10		
Set for channel Bh	11		

...

Set SOL Configuration Parameters		Transport	21h
reserved / unspecified	0		
Set for channel 1	1		
Set for channel 2	2		
Set for channel 3	3		
Set for channel 4	4		
Set for channel 5	5		
Set for channel 6	6		
Set for channel 7	7		
The following sub-function enables apply across each channel for which 'Set' has been enabled:			
Write Parameter 1 (SOL Enable)	8		
Write Parameter 2 (SOL Authentication)	9		
Write Parameters 3 & 4 (Character Accumulate Interval & Character Send Threshold, SOL Retry)	10		
Write Parameter 5 (SOL non-volatile bit rate -non-volatile)	11		
Write Parameter 6 (SOL volatile bit rate -volatile)	12		
Write Parameter 8 (SOL Payload Port Number)	13		
Set for channel 8	14		
Set for channel 9	15		
Set for channel Ah	16		
Set for channel Bh	17		

E410 Addendum - Table 35b-5, Forwarded Command Command

A completion code for target controller unavailable has been defined so that an implementation can report if the *Forwarded Command* command failed because the target controller was unavailable or absent.

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 Generic plus the following command-specific completion codes: 80h: Target controller unavailable. The forwarded command failed because the target controller could not accept the request. (On SMBus or I^2C/IPMB, this would be the case if the target controller were absent, or if it actively NAK'd the command). BMC shall return FFh or C3h (Timeout while processing command. Response Unavailable) completion code if the target controller does not return a matching Forwarded Command response message within the timeout set by the Enable Forwarded Commands command.
	2:M	Forwarded Command Response Data

E411 Clarification and Errata - 15.11 SOL Packet Acknowledge and Retries

The statement that packets would only be acknowledged if they contain character data was incorrect, since this would mean "control-only" packets from the remote console would not be acknowledged by the BMC. The clarified behavior is that only "ACK-only" packets (packets with a 0h Packet Sequence Number) are not acknowledged.

15.11 SOL Packet Acknowledge and Retries

A packet acknowledge is of one of two types:

- An ACK, indicating that the packet has been received and all its data has been accepted
- A NACK, indicating that the packet was received but some or all of the data could not be accepted

To improve efficiency, the packet acknowledgment information can be carried in a packet that also carries the SOL character data. Conversely, a packet can be an ACK-only packet that carries ACK or NACK information, but no data. ~~However, packets are only acknowledged if they contain character data.~~ Packets with a 0h Packet Sequence Number are not acknowledged. Therefore, ACK-only packets ~~themselves~~, which are specified to have a 0h Packet Sequence Number, are not acknowledged.

Except for ACK-only packets from the BMC, the remote console must acknowledge each SOL packet that it receives. If the BMC does not receive an ACK packet within a timeout interval, the BMC will resend (retry) the packet. The number of retries and the amount of time between retries are configurable through the SOL Configuration Parameters. Once the number of retries has been met the BMC will drop the packet and the data will be lost. Similarly, the BMC will acknowledge all packets it receives from the remote console that have a non-0h Packet Sequence Number. I.e. all packets except ACK-only packets from the remote console.

Revision 4 (6/12/06) Addenda, Errata, and Clarifications

E277 Clarification - Table 28-4, Chassis Control Command

The clarification is made to indicate that Chassis Control command can also be used for 'Blade Control'. The Chassis Control command can be applied to individual compute blades or compute partitions when those entities function as a logical computer 'system'. The power control aspects of the command are not required to be supported in those applications. An implementation can elect to not support the power on/off functions, can use them for power control of the blade/partition independent of the containing chassis, or may map them into a power control scheme for the overall chassis. For example, a scheme where chassis power will go off only after all blades within a chassis have been commanded into the 'power off' state.

Add as a note to the Chassis Control command and put corresponding superscripts referencing the note on the power down, power up, and power cycle bits.

Table 28-4, Chassis Control Command

	byte	data field
Request Data	1	[7:4] - reserved [3:0] - chassis control[2] ...
Response Data	1	Completion Code[1]

1. The implementation is allowed to return the completion code prior to performing the selected control action if necessary.
1. The command can also be used for compute blades or compute partition applications where the blades or partitions entities are emulating independent computer systems that implement IPMI. In these applications, the chassis power control aspects of the command are not required to be supported. Individual blades or computer partitions can elect to either not support the power on/off functions, can use them for power control of the blade/partition independent of the containing chassis, or may map them into a power control scheme for the overall chassis. For example, a scheme where chassis power will go off only after all blades within a chassis have been commanded into the 'power off' state.

E288 Clarification - Section 4. General Mgmt. Controller Required Functions

This erratum adds a clarification on how the mandatory commands listed in the tables apply to satellite controllers, as follows. This clarification also resolves redundancy with the BMC requirements given in Section 3.

4. ~~General Mgmt.~~Satellite Controller Required Functions

All satellite management controllers are required to implement the mandatory IPM Device commands. All other functions are optional. If a function is implemented, such as Event Generation or Sensors, then the mandatory commands for that function must be implemented.

It is highly recommended that ~~all~~ satellite controllers that provide sensors also provide event generation for those sensors. This will eliminate the need for system management software to poll to detect event conditions. It is also highly recommended that all satellite management controllers provide a Primary FRU Inventory device.

E299 Clarification - Table 22-34, Set User Password Command

This clarification provides more information on how the password test option is used with the *Set User Password* command.

Request Data

byte	data field
1	User ID. ...
2	[7:2] - reserved [1:0] - operation 00b = disable user 01b = enable user 10b = set password 11b = test password Compares the password data given in the request with the presently stored password and returns an OK completion code if there is a match. Otherwise, an error completion code is returned. (See the Completion Code description in the response data.)

...

E306 Clarification - Table 22-27, Get Channel Access Command

The Set and Get Channel Access commands do not specify behavior of the Per-message Authentication enable/disable bit for single session channels such as serial/modem channels that lack a per-message authentication capability. Additionally, there is some ambiguity regarding how much the PEF Alerting Enable/Disable bit affects whether PEF alerting is enabled or not. The following text is added for to clarify the use and operation of the PEF Alerting Enable/Disable and Per-message Authentication Enable/Disable settings in the Set and Get Channel Access commands:

Table 22-26, Set Channel Access Command

	byte	data field
Request Data	1	[7:4] - reserved [3:0] - Channel number
	2	[7:6] - 00b = don't set or change Channel Access 01b = set non-volatile Channel Access according to bits [5:0] 10b = set volatile (active) setting of Channel Access according to bits [5:0] 11b = reserved [5] - PEF Alerting Enable/Disable This bit globally gates whether PEF alerts can be issued from the given channel. Setting this to enable PEF alerting is a necessary part of enabling alerts for the channel, but for alerts to be generated the PEF and channel configuration must also be set to enable alerting. The setting this bit to 'enable' does not alter the PEF configuration or the alerting settings in the channel's configuration parameters. For example, if PEF is not configured for generating an alert, enabling PEF alerting with this bit will not change that configuration. Setting this bit to 'disable' will block PEF -generated alerts regardless of the PEF and channel configuration parameters. 0b = enable PEF Alerting 1b = disable PEF Alerting on this channel (the *Alert Immediate* command can still be used to generate alerts) [4] - Per-message Authentication Enable/Disable This bit is ignored for channels (e.g. serial/modem) that do not support Per-message Authentication. 0b = enable Per-message Authentication 1b = disable Per-message Authentication. [Authentication required to activate any session on this channel, but authentication not used on subsequent packets for the session.] [3] - User Level Authentication Enable/Disable. ...

...

Table 22-27, Get Channel Access Command

	byte	data field
Request Data	1	[7:4] - reserved [3:0] - Channel number.
	...	
	2	[7:6] - reserved [5] - 0b = Alerting enabled 1b = Alerting disabled [4] - Per-message Authentication Enable/Disable This bit is unspecified for channels (e.g. serial/modem) that do not support Per-message Authentication. 0b = per message authentication enabled 1b = per message authentication disabled [3] - User Level Authentication Enable 0b = User Level Authentication enabled. 1b = User Level Authentication disabled. ...

...

E412 Errata and Clarification - Typo in activate payload description

Typo in activate payload description. The text should be "Subsequently, the non-volatile…" as follows:

15.8 Volatile and Non-volatile SOL Configuration Parameters

SOL configuration parameters may be volatile, non-volatile, or have both volatile and non-volatile settings.

Unless otherwise specified, the volatile settings are copied from the non-volatile settings when the BMC first initializes. Subsequently, the non-volatile settings are restored whenever the payload is first *activated*.

Unless otherwise specified, changes to volatile parameters take effect immediately (within normal command processing time, typically ~30 milliseconds) and for the duration that the payload is activated. For example, changing the bit rate of SOL using the non-volatile setting will cause the BMC to immediately change its bit rate setting.

…

E413 Clarification - Table 22-8, Get Message Data Fields

20h is the standard slave address for the BMC. The Receive Message queue format omits the slave address from messages that were bridged to the Receive Message Queue from IPMB. Those messages includes checksums. In order to verify those checksums, they must be calculated as if 20h were the value that was dropped for the leading slave address byte. 20h is always used when IPMB is listed as the originating bus, and the checksums must always be calculated as if that slave address value were the one used to deliver the message to the BMC. This was implied (since the BMC slave address on IPMB is 20h per the [ADDR] specification) but not explicitly state. This is clarified by adding a note to Table 22-8, as follows.

Table 22-8, Get Message Data Fields

	Originating Channel Type	Channel Protocol	Message Data for received requests(RQ) and responses (RS)
1	IPMB (I^2C)	IPMB[1]	RQ: netFn/rsLUN, chk1, rqSA, rqSeq/rqLUN, cmd, <data>, chk2
			RS: netFn/rqLUN, chk1, rsSA, rqSeq/rsLUN, cmd, completion code, <data>, chk2
…			
12	System Interface	BT, KCS, SMIC	RQ/RS: See row for Originating Channel Type = 802.3 LAN.

1. This message data matches the IPMB message format with the leading slave address omitted. The format includes checksums. In order to verify those checksums, they must be calculated as if 20h (BMC slave address) was the value that was used as the slave address when the checksums were calculated per [IPMB]. 20h shall always be used for the checksum calculation for the receive message queue data whenever IPMB is listed as the originating bus and with IPMB as the Channel Protocol.

E414 Errata - Table 21-8, Get Command Enables Command

The parameters associated with Network Function 2Ch and 2Eh were in the response data rather than in the request parameters. This is incorrect. They are needed in the request parameters in order to enable selecting the Defining Body or OEM IANA, respectively, for which the Command Enables were configured. This is corrected as follows:

Table 21-8, Get Command Enables Command

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-~~7h~~Bh, Fh = channel numbers Eh = retrieve information for channel this request was issued on.
	2	[7:6] - Operation 00b = Get enable/disables for commands 00h through 7Fh. 01b = Get enables/disables 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
		For Network Function = 2Ch:
	(4)	Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)
		For Network Function = 2Eh:
	(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
	2:~~18~~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. For each bit in the bitfield: 0b = command is disabled or not supported 1b = command is enabled Software can use the *Get Command Support* command to determine which are supported, and the *Get Configurable Commands* command to determine which commands are configurable. Depending on the value of the “Operation” parameter passed in the request: byte 1, bit 0 corresponds to command 00h or command 80h byte 1, bit 7 corresponds to command 07h or command 87h … byte 16, bit 0 correspond to command 78h or command F8h byte 16, bit 7 corresponds to command 7Fh or command FFh
		~~*For Network Function = 2Ch:*~~
	~~(18)~~	~~Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)~~
		~~*For Network Function = 2Eh:*~~
	~~(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)~~

E415 Errata - Table 21-9, Set Configurable Command Sub-function Enables Command and Table 21-10, Get Configurable Command Sub-function Enables Command

The parameters associated with Network Function 2Ch and 2Eh were missing from the request parameters. They are needed in the request parameters in order to enable selecting the Defining Body or OEM IANA, respectively, for which the Command Enables were configured. This is corrected as follows:

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

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-Bh, Fh = channel numbers Eh = retrieve information for channel this request was issued on.
		...
		For Network Function not equal to 2Ch or 2Eh:
	5:8	Sub-Function Enables (ls-byte first). The enable/disable settings are *non-volatile* and take effect on successful completion of the command. ...
	(9:12)	These additional 32-bits, if present, form a bitfield where each bit indicates support for a particular sub-function for the given command, starting from sub-function 32. The bit offset corresponds to the number of the sub-function. ...
		For Network Function = 2Ch:
	5	Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)
	6:9	Sub-Function Enables (see definition for bytes 5:8 for “Network Function not equal to 2Ch or 2Eh” case, above.)
	(10:13)	These additional 32-bits, if present, form a bitfield where each bit indicates support for a particular sub-function for the given command, starting from sub-function 32. The bit offset corresponds to the number of the sub-function. (see definition for bytes 9:12 for “Network Function not equal to 2Ch or 2Eh” case, above.)
		For Network Function = 2Eh:
	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)
	8:11	Sub-Function Enables (see definition for bytes 5:8 for “Network Function not equal to 2Ch or 2Eh” case, above.)
	(12:15)	These additional 32-bits, if present, form a bitfield where each bit indicates support for a particular sub-function for the given command, starting from sub-function 32. The bit offset corresponds to the number of the sub-function. (see definition for bytes 9:12 for “Network Function not equal to 2Ch or 2Eh” case, above.)
IPMI Response Data	1	Completion Code Generic, plus following command-specific completion codes: 80h = attempt to enable an unsupported or un-configurable sub-function.

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

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-**Bh**, Fh = channel numbers Eh = retrieve information for channel this request was issued on.
	2	[7:6] - 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
	4	[7:0] - CMD. Command number to set command sub-function enables for.
		For Network Function = 2Ch:
	(5)	Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)
		For Network Function = 2Eh:
	(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 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 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 *Table H-1, Sub-function Number Assignments*. 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.
	(6:9)	These additional 32-bits, if present, form a bitfield where each bit indicates support for a particular sub-function for the given command, starting from sub-function 32. The bit offset corresponds to the number of the sub-function. See *Table H-1, Sub-function Number Assignments*. These bytes are not required to be returned unless the particular command has sub-functions number definitions >31. Software should assume that an implementation may return these bytes for any command, if the particular command does not have any sub-function numbers >31 specified. 1b sub-function is enabled 0b sub-function is disabled or is un-configurable/reserved. [31] - bit for sub-function 63. [30] - bit for sub-function 62. … [1] - bit for sub-function 33. [0] - bit for sub-function 32.

E416 Clarification - Section 13.27.1, IPMI Message Payloads and IPMI Commands

It was difficult to figure out which IPMI commands should remain available when payloads are used while IPMI Messaging is disabled for the channel. The following table note in section 24.6 should be made more obvious and a better description of what can be done when IPMI messaging is disabled should be provided:

> 1. The following commands remain available for payloads if IPMI Messaging Payload type is disabled: Deactivate Payload, Suspend/Resume Payload Encryption (as defined for given payload), Get Payload Activation Status, Get Channel Payload Version Command, Get Channel OEM Payload Info (if implemented), Set Session Privilege Level, and Close Session.

The text should also indicate that the pre-session commands are available while IPMI Messaging is disabled.

This is clarified by adding text to Section 13.27.1, IPMI Message Payloads and IPMI Commands, as follows:

13.27.1 IPMI Message Payloads and IPMI Commands

IPMI Message Payloads are always accepted over any IPMI session, because they are used for IPMI commands that are used for managing sessions. Thus, the IPMI payload type does not need to be explicitly enabled, and cannot be disabled via the *Activate* and *Deactivate Payload* commands, respectively.

However, while the IPMI Message payload type is accepted, specific IPMI commands may not be accepted. For example, the *Set User Access* command determines whether a given user can execute IPMI commands that are not specific to managing a session or to specific to a particular payload type. For example, if IPMI Messaging is disabled for a user, but the user is enabled for activating the SOL payload type, then IPMI commands associated with SOL and session management, such as *Get SOL Configuration Parameters* and *Close Session* are available, but generic IPMI commands such as *Get SEL Time* are unavailable on the SOL Payload session.

The following commands remain available for payloads if IPMI Messaging Payload type, or IPMI Messaging, is disabled for the channel:

> *Deactivate Payload*, S*uspend/Resume Payload Encryption* (as defined for given payload), *Get Payload Activation Status*, *Get Channel Payload Version Command*, *Get Channel OEM Payload Info* (if implemented), *Set Session Privilege Level*, and *Close Session*.

In addition, the IPMI commands that are available before a session is established, and commands that are required to activate a session, such as *Get System GUID*, and *Activate Session,* also remain available. These commands are identified with the notation “p” in Table G-1, Command Number Assignments and Privilege Levels. Note some of these commands are not supported for IPMI v1.5/RMCP connections, in which case they will be unavailable.

E417 Clarification - Section 12.12, Discovering SSIF and Table C1-1, SM BIOS IPMI Device Information Record

The text for the section on “Discovering SSIF” may be misinterpreted as indicating that the slave address of the SSIF is not the same as the slave address of the BMC for IPMB and other management purposes. This is addressed with clarifications as follows:

12.12 Discovering SSIF

The recommended SMBus slave address for the SSIF to the BMC is address 20h (0010_000x binary). The SSIF Interface can be located at alternative addresses depending on the implementation. Note the slave address of the SSIF is not the same thing as the slave address of the BMC that is used for IPMB and IPMI Message use. For example, the slave address of the BMC on IPMB, and the slave address used with the *Get Message* command is required to be 20h regardless of the address used for the BMC on the SSIF Interface. An Intel architecture compatible system implementation that supports SMBIOS must include an SMBIOS "Type 38" record to support system management software discovery of the existence and slave address of the SSIF.

Also, the word "normally" in the text in the table on "SMBIOS IPMI Device Information Record" infers that the BMC may have other addresses. That text should be deleted as follows:

Table C1-1, SM BIOS IPMI Device Information Record

Offset	Name	Length	Value	Description
00h	Type	BYTE	38	IPMI Device Information structure indicator. (Note this number is given in decimal)
…				
06h	I^2C Slave Address	BYTE	~~Varies~~ 32	The slave address on the I^2C bus of this BMC. (This refers to the address of the BMC on the primary IPMB, if present. This value is ~~normally~~ set to 20h [0010_000x] per the IPMB specification.)
…				

E418 Clarification - Section 35.12, Re-arm Sensor Events Command

The description may be misleading in implying that the behavior is different for auto- re-arm sensors versus manual re-arm sensors. The behavior is the same. In both cases, the command will cause the sensor to regenerate events (event messages and event status) based on the present event condition. This is clarified with changes to the text as follows:

35.12 Re-arm Sensor Events Command

This command is provided to enable software to re-arm thresholds on sensors that require 'manual' re-arming. ~~It can also be used with sensors that automatically re-arm.~~It is also used to enable software to cause sensors (both manual and auto re-arm sensors) ~~to cause them~~ to regenerate events (update their event status and, if enabled, generate event messages) ~~when~~ according to what ~~an~~ event condition(s) currently exist~~s~~ (is presently in effect) when the re-arm command is executed. Thus, t~~T~~he re-arm is actually a request for the event status for a sensor to be rechecked and updated, and if enabled, generate event messages based on that event status.

E419 Addendum - Table 24-2, Activate Payload Command, Table 24-3, Deactivate Payload Command, and Table 24-7, Get Payload Instance Info Command

Since IPMI as a payload type is intrinsically supported for all sessions that support payloads, there is no need to activate it. Furthermore, there are issues with returning an appropriate set of response fields if IPMI were used at the Payload Type in the *Get Payload Instance Info* command. Thus, IPMI should not be passed as a payload type in the *Activate Payload, Deactivate Payload,* or *Get Payload Instance* commands. This addendum adds text indicating that an error completion code should be returned if "IPMI" is passed

as the payload type in the *Activate Payload*, *Deactivate Payload*, or *Get Payload Instance* commands, as follows:

Table 24-2, Activate Payload Command

	byte	data field
Request Data	1	[7:6] - reserved [5:0] - payload type (See *Table 13-16, Payload Type Numbers*) *IPMI Message* payloads do not need to be explicitly activated. A payload that is required to be launched over a different port than that used to establish the initial IPMI session is only required to support the IPMI commands needed by the particular payload type.
	...	
Response Data	1	Completion Code Generic plus the following command-specific completion codes: (An error completion code should be returned if the payload type in the request is set to "IPMI Message" (0h)). 80h: Payload already active on another session (required). This will be returned any time an attempt is made to activate a payload type when that type is already activated for another session, and when the BMC only supports one instance of that payload type running at a time. 81h: Payload type is disabled (optional). Given payload type is not configured to be enabled for activation. 82h: Payload activation limit reached. Cannot activate given payload type because the maximum number of simultaneous instances of that payload type are already running. 83h: Cannot activate payload with encryption. 84h: Cannot activate payload without encryption. BMC requires encryption for all payloads for given privilege level.
	...	

Table 24-3, Deactivate Payload Command

	byte	data field
Request Data	1	[7:6] - reserved [5:0] - payload type (See *Table 13-16, Payload Type Numbers*)
	...	
Response Data	1	Completion Code Generic plus the following command-specific completion codes: (An error completion code should be returned if the payload type in the request is set to "IPMI Message" (0h)). 80h: Payload already deactivated. 81h: Payload type is disabled (optional). Given payload type is not configured to be enabled for activation.
	...	

Table 24-7, Get Payload Instance Info Command

	byte	data field
Request Data	1	Payload Type Number - Type number of the standard payload type or OEM Payload Handle to retrieve status for. See *Table 13-16, Payload Type Numbers*.
	...	
Response Data	1	Completion Code An error completion code should be returned if the payload type in the request is set to "IPMI Message" (0h).
	...	

E420 Clarification - Table 1-1, Glossary

The existing definition of the term "Hard Reset" in the glossary may be confusing when Hard Resets are discussed elsewhere in the document. The present definition states that a Hard Reset is: "A reset event in the system that initializes all components and invalidates caches". But since the BMC may be considered to be part of "all components", this could be misconstrued that the BMC (and any satellite controllers) must also be reset when issuing a hard reset via the *Chassis Control* command. This is addressed with a revised definition of the Hard Reset in the glossary, as follows:

Table 1-1, Glossary

Term	Definition
...	
Hard Reset	A hardware reset event ~~in the system~~ that initializes ~~all~~ components and invalidates caches for a system or subsystem. In the context of this specification, the term Hard Reset is generally used to refer to System Hard Resets, where System Hard Resets are Hard Resets of the computer system that do not reset the BMC, Satellite Controllers, or other elements of the platform management subsystem. Unless explicitly stated, Hard Resets or System Hard Resets do not refer to resets of the BMC or other elements of the platform management subsystem.
...	

E421 Clarification - Section 13.28.4, Integrity Algorithms

The text doesn't fully describe the use of IPMI v2.0/RMCP+ Integrity Algorithms with the *Get AuthCode* command, and could be misconstrued as conflicting with the information on using IPMI v2.0/RMCP+ Integrity Algorithms that is written in the field descriptions of the *Get AuthCode* command. This is clarified as follows:

13.28.4 Integrity Algorithms

…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.

When using the integrity algorithms with the *Get AuthCode* command, the integrity algorithm is applied to the data passed in the *Get AuthCode* command, using key information selected by the given User ID and Channel number. If an SIK needs to be calculated, it is calculated using the user key (password) information as descibed for 'one-key' logins.

E422 Clarification - Table 13-11, RAKP Message 1

Clarification for User Name field in RMCP+ RAKP Message 1. Folks get confused whether this is a variable or fixed-length field. It's variable length. Which is why a length field (byte 28) was provided.

The clarification is to add parentheses around 29:44 for the User Name field to indicate a variable length field per IPMI specification notation convention, as follows:

Table 13-11, RAKP Message 1

	byte	data field
IPMI Session Header		Payload Type = RAKP Message 1 Session ID = 00_00_00_00h Session Sequence Number = 00_00_00_00h
IPMI Payload	1	Message Tag - Selected by remote console. Used by remote console to help match responses up with requests. In this case, the corresponding *RAKP Message 2* that is returned by the BMC. The BMC can use this value to help differentiate retried messages from new messages from the remote console.
	...	
	28	User Name Length 00h No name present 01h-10h Name length 11h-FFh Reserved for future definition by this specification
	(29:44)	User Name ASCII character Name that the user at the Remote Console wishes to assume for this session. No NULL characters (00h) are allowed in the name. Sixteen-bytes, max.

E423 Addendum - Table 42-2, Generic Event/Reading Type Codes

The Session Audit does not include events that would enable logging invalid passwords/usernames that have been received while an IPMI Session was trying to be established. This Addendum add a sensor-specific offset to the Session Audit sensor type to cover this kind of event.

Session Audit	2Ah	00h 01h 02h	Session Activated Session Deactivated Invalid Username or Password An Invalid Username or Password was received during the session establishment process. *The Event Data 2 & 3 fields can be used to provide an event extension code for the preceding offsets, with the following definition:* Event Data 2 7:6 reserved 5:0 User ID for user that activated session. 00_0000b = unspecified. Event Data 3 7:6 reserved 5:4 Deactivation cause 00b = Session deactivatation cause unspecified. This value is also used for Session Activated events. 01b = Session deactivated by C*lose Session* command 10b = Session deactivated by timeout 11b = Session deactivated by configuration change 3:0 Channel number that session was activated/deactivated over. Use channel number that session was activated over if a session was closed for an unspecified reason, a timeout, or a configuration change.

E424 Addendum - Table 17-1, PEF Action Priorities

The reset PEF action is changed from Mandatory to Optional.

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~~optional) 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 *Section 17.11, Alert Policy Table*.
OEM	OEM	(optional) Priority determined by OEM.

E425 Addendum - Table 5-1, Network Function Codes

Type 02h is added to the Group Extension Network Function Code to support the Server System Infrastructure forum, as shown below.

Table 5-1, Network Function Codes

Value(s)	Name	Meaning	Description
			…
2Ch-2Dh	Group Extension	Non-IPMI group Requests and Responses	The first data byte position in requests and responses under this network function identifies the defining body that specifies command functionality. Software assumes that the command and completion code field positions will hold command and completion code values. The following values are used to identify the defining body: 00h** PICMG - PCI Industrial Computer Manufacturer's Group. (www.picmg.com) 01h DMTF Pre-OS Working Group ASF Specification (www.dmtf.org) 02h Server System Infrastructure (SSI) Forum (www.ssiforum.org) all other Reserved When this network function is used, the ID for the defining body occupies the first data byte in a request, and the second data byte (following the completion code) in a response.
			…

E426 Addendum - Table 13-7, RMCP Packet Fields for ASF Presence Pong Message (Ping Response)

The definition of the Supported Interactions field in the ASF Presence Pong is updated to reflect the definition and use of bit 5 for indicating support for the DMTF DASH initiative.

Table 13-7, RMCP Packet Fields for ASF Presence Pong Message (Ping Response)

	Field	size in bytes	Value
UDP Header	Source Port	2	26Fh
	Destination Port	2	from Ping request
	UDP Length	2	per UDP
	UDP Checksum	2	per UDP
RMCP Header	Version	1	6 = RMCP Version 1.0
	Reserved	1	00h
	RMCP Sequence Number	1	FFh for IPMI[2]
	Class of Message	1	06h = ASF
ASF Message	IANA Enterprise Number	4	4542 = ASF IANA
	Message Type	1	40h = Presence Pong
	Message Tag	1	from Ping request
	Reserved	1	00h
	Data Length	1	16 (10h)
	IANA Enterprise Number	4	If no OEM-specific capabilities exist, this field contains the ASF IANA (4542) and the OEM-defined field is set to all zeroes (00000000h). Otherwise, this field contains the OEM's IANA Enterprise Number and the OEM-defined field contains the OEM-specific capabilities.
	OEM-defined	4	Not used for IPMI. This field can contain OEM-defined values; the definition of these values is left to the manufacturer identified by the preceding IANA Enterprise number.
	Supported Entities	1	81h for IPMI [7] 1b = IPMI Supported [6:4] Reserved [3:0] 0001b = ASF Version 1.0
	Supported Interactions	1	[7] Set to 1b if RMCP security extensions are supported[1] [6] Reserved for future definition by ASF specification. Set to 0b. [5] Set to 1b if DMTF DASH is

		supported [4:0] Reserved for future definition by ASF specification, set to 00000b
Reserved	6	Reserved for future definition by ASF specification, set to 00 00 00 00 00 00h

1. IPMI v1.5 and IPMI v2.0/RMCP+ do not use RMCP security extensions specified in [ASF 2.0], thus this bit will typically be 0bs. It's possible a BMC implementation could also support ASF 2.0 messages, in which case this bit could be set to indicate those extensions are supported for ASF messages that would utilize them.

E427 Addendum - Table 42-3, Sensor Type Codes

The following addendum adds support to the Processor sensor for monitoring and generating events for processor Machine Check Exception and Correctable Machine Check Error events.

Table 42-3, Sensor Type Codes

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
			...
Processor	07h	00h 01h ... 0Ah 0Bh 0Ch	IERR Thermal Trip ... Processor Automatically Throttled (processor throttling triggered by a hardware-based mechanism operating independent from system software, such as automatic thermal throttling or throttling to limit power consumption.) Machine Check Exception (Uncorrectable) Correctable Machine Check Error
			...
Event Logging Disabled	10h	00h ... 06h	Correctable Memory Error Logging Disabled Event Data 2 [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). ... Correctable Machine Check Error Logging Disabled If the following field is not provided, then this event indicates that Correctable Machine Check error logging has been disabled for all Processor sensors. Event Data 2 Event Data 2 may be optionally used to return an Entity Instance or a vendor selected processor number that identifies the processor associated with this event. [7:0] - Instance ID number of the (processor) Entity that the sensor is associated with (if SDR provided for this sensor), or a vendor selected logical processor number if no SDR. Event Data 3 If Event Data 2 is provided then Event Data 3 may be optionally used to indicate whether Event Data 2 is being used to hold an Entity Instance number or a vendor-specific processor number. If Event Data 2 is provided by Event Data 3 is not, then Event Data 2 is assumed to hold an Entity Instance number. [7] - 0b = Entity Instance number 1b = Vendor-specific processor number [6:0] - reserved
			...

E428 Clarification - Table 22 -14, Master Write-Read Command

The use of the bus type bit and its relationship and interaction with the channel number and bus ID values was not explained. This is clarified as follows:

Table 22-14, Master Write-Read Command

	byte	data field
Request Data	1	bus ID: [7:4] channel number (Ignored when bus type = 1b) [3:1] bus ID, 0-based (always 000b for public bus [bus type = 0b]) [0] bus type: 0b = public (e.g. IPMB or PCI Management Bus. The channel number value is used to select the target bus.) 1b = private bus (The bus ID value is used to select the target bus.)

...

E429 Addendum - Table 42-3, Sensor Type Codes

This addendum extends the Slot/Connector Types in Event Data 2 of the Slot / Connector sensor with the addition of a USB type, as follows:

Table 42-3, Sensor Type Codes

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
...			
Slot / Connector	21h	00h ... 09h	Fault Status asserted ... Slot holds spare device *The Event Data 2 & 3 fields can be used to provide an event extension code, with the following definition:* Event Data 2 7 reserved 6:0 Slot/Connector Type 0 PCI 1 Drive Array 2 External Peripheral Connector 3 Docking 4 other standard internal expansion slot 5 slot associated with entity specified by Entity ID for sensor 6 AdvancedTCA 7 DIMM/memory device 8 FAN 9 PCI Express™ 10 SCSI (parallel) 11 SATA / SAS 12 USB all other = reserved Event Data 3 7:0 Slot/Connector Number

...

E430 Table 28-14, Boot Option Parameters

This addendum adds remote boot device support to the Boot Option Parameters by extending the Boot device selector as follows:

Table 28-14, Boot Option Parameters

Parameter	#	Parameter Data (non-volatile unless otherwise noted)
…		
boot flags (semi-volatile) [1]	5	data 1 … data 2 [7] - 1b = CMOS clear [6] - 1b = Lock Keyboard [5:2] - Boot device selector ⊙ 0000b = No override 0001b = Force PXE 0010b = Force boot from default Hard-drive[2] 0011b = Force boot from default Hard-drive, request Safe Mode[2] 0100b = Force boot from default Diagnostic Partition[2] 0101b = Force boot from default CD/DVD[2] 0110b = Force boot into BIOS Setup 0111b = Force boot from remotely connected (redirected) Floppy/primary removable media[2] 1001b = Force boot from primary remote media[2] 1000b = Force boot from remotely connected (redirected) CD/DVD[2] 1010b = reserved 1011b = Force boot from remotely connected (redirected) Hard Drive[2] ~~0111b~~-1100-1110b = Reserved 1111b = Force boot from Floppy/primary removable media[2] [1] - 1b = Screen Blank [0] - 1b = Lock out Reset buttons ⊙ data 3 …
…		

2. IPMI allows software to use the boot initiator mailbox as a way for a remote application to pass OEM parameters for additional selection of the boot process and direction of the startup of post-boot software. If additional parameters are not included, the system boots the primary/first-scanned device of the type specified.

…

E431 Add SHA-256 -based Authentication and Integrity Algorithm Support

The following additions are made to enable optional support of SHA256 for use for IPMI RAKP+ Authentication and Integrity.

[FIPS-180-2] NIST, FIPS PUB 180-2: Secure Hash Standard, August 2002. http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

[RFC 4634] *US Secure Hash Algorithms (SHA and HMAC-SHA)*, D. Eastlake 3rd, Motorola Labs , T. Hansen, AT&T Labs, July 2006

[RFC 4868] *Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec*, S. Kelly, Aruba Networks, S. Frankel, NIST, May 2007

Table 13-17, Authentication Algorithm Numbers

number*	type	Mandatory / Optional
00h	RAKP-none	M
…		
03h	RAKP-HMAC-SHA256	O
C0h-FFh	OEM	O
all other	reserved	-

* The number range is limited to six (6) bits (00h-3Fh)

13.28.1b RAKP-HMAC-SHA256 Authentication Algorithm

RAKP-HMAC-SHA256 specifies the use of RAKP messages for the key exchange portion of establishing the session, and that HMAC-SHA256 (per [FIPS 180-2] and [RFC4634] and is used to create a *32-byte* Key Exchange Authentication Code fields in RAKP Message 2 and RAKP Message 3. HMAC-SHA256-128 (per [RFC4868]) is used for generating a *16-byte* Integrity Check Value field for RAKP Message 4.

13.28.4 Integrity Algorithms

…

HMAC-SHA1-96, HMAC-SHA256-128, 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+.

Table 13-18, Integrity Algorithm Numbers

number*	type	Mandatory / Optional[1]
00h	none	M
01h	HMAC-SHA1-96	M
02h	HMAC-MD5-128	O
03h	MD5-128	O
04h	HMAC-SHA256-128	O
C0h-FFh	OEM	O
all other	reserved	-

* The number range is limited to six (6) bits (00h-3Fh)

E432 Typo - Section 13.28, Authentication, Integrity, and Confidentiality Algorithm Numbers

The text referred to RFC2101 instead of RFC2404. 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~~RFC2104]) 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.

…

E433 Addendum - Table 42-3, Sensor Type Codes

Add offset for Thermal Trip to the chip set sensor.

Table 42-3, Sensor Type Codes

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
			…
Chip Set	19h	00h	Soft Power Control Failure (chip set did not respond to BMC request to change system power state). This offset is similar to offset 05h for a power unit, except that the power unit event is only related to a failure to power up, while this event corresponds to any system power state change directly requested via the BMC. Event Data 2 … Event Data 3 … 01h Thermal Trip
			…

E434 Clarification and Errata - Table 22-19, Cipher Suite IDs and 13.28.2, RAKP-none Authentication Algorithm

The characteristics for Cipher Suite 0 was incorrectly called "straight password". Table 22-19 lists "straight password" under the characteristics for Cipher Suite 0, but because of the way RAKP-none is specified, there is no way for passing a password into RAKP. The Authentication Code field that would be used to carry the password is specified as being absent for Cipher Suite 0, and there no specification of how a straight password would be carried in the Authentication Code field when authentication is not being used. Thus, the 'characteristics' is being changed from being called 'straight password' to 'no password'. This makes it consistent with the description and usage of the "RAKP-none" authentication algorithm. The characteristics for Cipher Suite ID 0 are changed as follows:

Table 22-19, Cipher Suite IDs

ID	characteristics	Cipher Suite	Authentication Algorithm	Integrity Algorithm(s)	Confidentiality Algorithm(s)
0	~~"straight password""~~"no password"	00h, 00h, 00h	RAKP-none	None	None

...

The description of the RAKP-none algorithm is also updated to clarify that the RAKP-none does not use passwords, as follows:

13.28.2 RAKP-none Authentication Algorithm

RAKP-none uses the same steps and messages as RAKP-HMAC-SHA1, but the Key Exchange Authentication Code field in RAKP Message 2 and RAKP Message 3 and the Integrity Check Value field in RAKP Message 4 are absent since they are not used. RAKP-none does not provide password authentication or RAKP packet level data integrity checking. The RAKP steps establish Session IDs and privilege level ~~for~~ using only the given username/role. A BMC implementation can be configured with a null username that has a null (all 0's) password. A BMC configured this way, and using the RAKP-none Authentication Algorithm, provides a way to enable access the BMC without requiring a username and password.

E435 Addendum - Table 28-14, Boot Option Parameters

This adds the capability to select a particular instance of a boot device.

Table 28-14, Boot Option Parameters

Parameter	#	Parameter Data (non-volatile unless otherwise noted)
…		
boot flags (semi-volatile) [1]	5	data 1 [7] - 1b = boot flags valid. The bit should be set to indicate that valid flag data is present. This bit may be automatically cleared based on the boot flag valid bit clearing parameter, above. … data 5 - ~~reserved~~ [7:5] – reserved [4:0] – Device Instance Selector If this value is 0, then there is no change to Boot Device Selector behavior. If this value is not zero, then the behavior of Boot Device Selector for the following values 0001b = Force PXE 0010b = Force boot from default Hard-drive, 0011b = Force boot from default Hard-drive, request Safe Mode, 0101b = Force boot from default CD/DVD, 0111b = Force boot from remotely connected (redirected) Floppy/primary removable media 1000b = Force boot from remotely connected (redirected) CD/DVD 1001b = Force boot from primary remote media 1011b = Force boot from remotely connected (redirected) Hard Drive 1111b = Force boot from Floppy/primary removable media is modified to use the value in this parameter to select a particular instance of a device of that type to boot from. For example, if the Boot Device Selector were 0010b (Hard-drive) and the Logical Device selector was 00010b, it would select logical external hard drive 1. Devices instances may be physical or logical depending on the system. The system should attempt to boot from the specified device instance first. If the boot fails the system should attempt booting using the system's boot order configuration. If the requested value is out of the range of logical devices, then the system should treat the value as zero. 00000b = no specific device instance requested 00001b to 01111b = external (direct attached) device instance 1 to 15, respectively 10000b = reserved 10001b to 11111b = internal device instance number 1 to 15, respectively
…		

E436 Typo - Table 22-34, Set User Password Command

For 20 byte passwords, the Password data field is a fixed 20 bytes in size. ASCII strings that are less than 20 bytes must then be padded to 20 bytes. This was incorrectly written as 16 bytes due to a cut-n-paste error. This is corrected as follows:

Table 22-34, Set User Password Command

...

	For password size = 20 bytes:
3:22	20-byte Password data. This is a required, fixed length field when used for the set- and test password operations. If the password is entered as an ASCII string, it must be null (00h) terminated and 00h padded if the string is shorter than ~~16~~ 20 bytes. This field need not be present if the operation is 'disable user' or 'enable user'. If this field is present for those operations, the BMC will ignore the data.

E437 Addendum - Table 5-1, Network Function Codes, add Group Extension

A group extension value is added for the DCMI specifications as follows:

Value(s)	Name	Meaning	Description
00, 01	Chassis	Chassis Device Requests and Responses	00h identifies the message as a command/request and 01h as a response, relating to the common chassis control and status functions.

...

2Ch, 2Dh	Group Extension	Non-IPMI group Requests and Responses	The first data byte position in requests and responses under this network function identifies the defining body that specifies command functionality. Software assumes that the command and completion code field positions will hold command and completion code values. The following values are used to identify the defining body: 00h** PICMG - PCI Industrial Computer Manufacturer's Group. (www.picmg.com) 01h DMTF Pre-OS Working Group ASF Specification (www.dmtf.org) 02h Server System Infrastructure (SSI) Forum (www.ssiforum.org) DCh DCMI Specifications (www.intel.com/go/dcmi) all other Reserved When this network function is used, the ID for the defining body occupies the first data byte in a request, and the second data byte (following the completion code) in a response.

...

E438 Errata and Clarification - Section 13.28.4 - Integrity Algorithms

This errata provides additional explanation about how the Integrity Algorithms are used to calculate the AuthCode field and how the MD5-128 Integrity Algorithm is applied. The errata also clarifies the resulting sizes of the AuthCode field for the different algorithms.

none. If the Integrity Algorithm is **none** the AuthCode value is not calculated and the AuthCode field in the message is not present (zero bytes).

HMAC-SHA1-96, HMAC-SHA256-128, 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 to produce the AuthCode field value ~~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+.

When the HMAC-SHA1-96 Integrity Algorithm is used the resulting AuthCode field is 12 bytes (96 bits).

When the HMAC-SHA256-128 and HMAC-MD5-128 Integrity Algorithms are used the resulting AuthCode field is 16-bytes (128 bits).

MD5-128 uses a straight MD5 signature with the user's key information appended at the beginning and the end of the packet data to calculate the AuthCode field as:

-AuthCode = MD5(password + AuthType/Format + … + Next_Header + password)

The MD5-128 Integrity Algorithm does not use K1 or HMAC. This results in significantly fewer computation steps than the HMAC- algorithms, potentially providing significantly improved throughput performance on certain management controllers. However, this algorithm also delivers less protection against password and replay attacks than the HMAC based options and thus should only be used when operating in a trusted environment where data integrity checking is desired but other attacks are not a concern.

When the MD5-128 Integrity Algorithm is used the resulting AuthCode field is 16 bytes (128 bits).

E439 Addendum - Table 42-3, Sensor Type Codes

Additional Error Type values are added to the Configuration Error offset for the power supply sensor as follows:

Table 42-3, Sensor Type Codes

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
…			
Power Supply (also used for power converters [e.g. DC-to-DC converters] and VRMs [voltage regulator modules]).	08h	00h 01h 02h 03h 04h 05h 06h	Presence detected Power Supply Failure detected Predictive Failure Power Supply input lost (AC/DC)[2] Power Supply input lost or out-of-range Power Supply input out-of-range, but present Configuration error. The Event Data 3 field provides a more detailed definition of the error: 7:4 = Reserved for future definition, set to 0000b 3:0 = *Error Type*, one of 0h = Vendor mismatch, for power supplies that include this status. (Typically, the system OEM defines the vendor compatibility criteria that drives this status). … 3h = Power Supply rating mismatch. The power rating of the supply does not match the system's requirements. 4h = Voltage rating mismatch. The voltage rating of the supply does not match the system's requirements. Others = Reserved for future definition
…			

E440 Addendum - Table 43-13, Entity ID Codes

This addendum is for the addition of a new Entity ID to cover an air inlet. Table 43-13 is updated as shown below. Code 54 (36h) is set to 'reserved' to help avoid any misinterpretation due to a previous cut-and-paste error in the specification markup that had the System Firmware entity type (22h) duplicated in the table as Entity ID 36h. Additional codes were defined to improve interoperability with the DCMI specifications.

Table 43-13, Entity ID Codes

Code		Entity
0	00h	unspecified
...		
54	36h	reserved. This value was previously a duplicate of 22h (System Firmware). This value should remain reserved for any future versions of the specification to avoid conflicts with older applications that may interpret this as System Firmware.
...		
56-63	38h-3Fh	reserved
64	40h	air inlet - This Entity ID enables associating sensors such as temperature to the airflow at an air inlet.
65	41h	processor / CPU - This Entity ID value is equivalent to Entity ID 03h (processor). It is provided for interoperability with the DCMI specifications.
66	42h	baseboard / main system board - This Entity ID value is equivalent to Entity ID 07h (system board). It is provided for interoperability with the DCMI specifications.
...		

E441 Addendum - Table 42-3, Sensor Type Codes

Add text to the Power Unit sensor offset 04h to indicate that the offset can apply to any loss of the power source for the Power Unit, not just AC power.

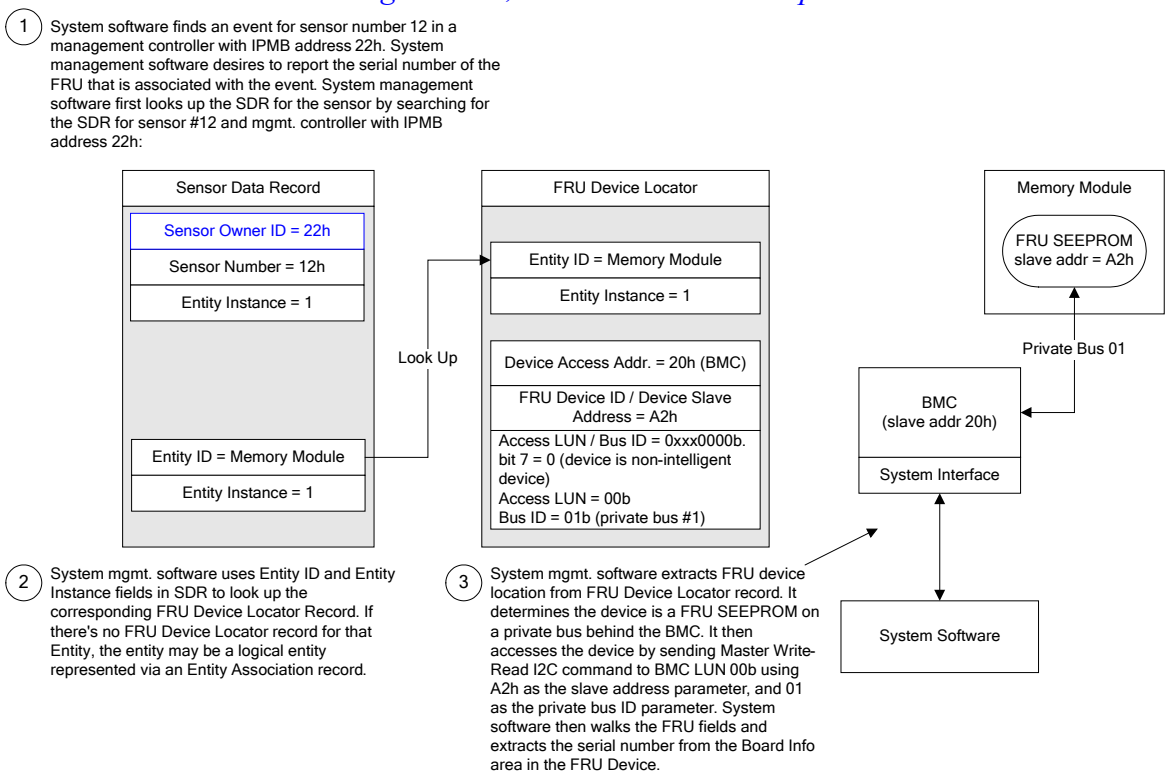
Table 42-3, Sensor Type Codes

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
...			
Power Unit	09h	00h	Power Off / Power Down
		01h	Power Cycle
		02h	240VA Power Down
		03h	Interlock Power Down
		04h	AC lost / Power input lost (The power source for the power unit was lost)
		05h	Soft Power Control Failure (unit did not respond to request to turn on)
		06h	Power Unit Failure detected
		07h	Predictive Failure

E442 Clarification - Figure 39-1, Sensor to FRU Lookup

The figure described looking up the SDR for a management controller with IPMB address 22h but did not include an example field showing 22h. This has led to some confusion about whether the SDR is for a device at 22h or whether it is a typo and the SDR is intended to be for the BMC at address 20h. This is clarified by showing the content of the Sensor Owner ID field in the SDR as 22h, as shown:

Figure 39-1, Sensor to FRU Lookup



E443 Addendum and Errata - Table 23-4, LAN Configuration Parameters, Table 25-4, Serial/Modem Configuration Parameters, Table 42-3, Sensor Type Codes, and Table H-1, Sub-function Number Assignments

A "Bad Password Threshold" parameter is added to the Serial/Modem Configuration Parameters and LAN Configuration Parameters to allow the option of automatically disabling a user's access on a particular channel after a series of login attempts that failed due to bad passwords. This requires additions to Table 23-4, LAN Configuration Parameters, Table 25-4, Serial/Modem Configuration Parameters, Table 42-3, Sensor Type Codes, and Table H-1, Sub-function Number Assignments to support the new parameter and to extend the Session Audit sensor to support an event that indicates when a user was automatically disabled due to bad passwords.

This also corrects an error in Table H-1. Table H-1 was missing a sub-function number to cover the Destination Address VLAN Tags parameter for the Set LAN Configuration Parameters command.

Table 23-4, LAN Configuration Parameters

Parameter	#	Parameter Data (non-volatile unless otherwise noted)[1]
		...
Destination Address VLAN TAGs (can be READ ONLY, see description)	25	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. ...

Bad Password Threshold (optional)	26	Sets/Gets the Bad Password Threshold. If implemented and non-zero, this value determines the number of sequential bad passwords that will be allowed to be entered for the identified user before the user is automatically disabled from access on the channel. For example, a value of 3 indicates that 3 sequential attempts are allowed for the given username on the particular channel. If the password for the third attempt is not correct, the user will be disabled for the channel. If this value is zero (00h) then there is no limit on bad passwords. The effect of the disable is the same as if a Set User Access command were used to remove the user's access from the channel. Bad password attempts are tracked according to individual username on a per channel basis. (Thus, a given username may be disabled on one channel, but still enabled on another) Bad password attempts are not counted if integrity check or other session parameters, such as session ID, sequence number, etc. are invalid. That is, bad password attempts are not counted if there are any other errors that would have caused the login attempt to be rejected even if the password was valid. The count of bad password attempts is retained as long as the BMC remains powered and is not reinitialized. Counting automatically starts over (is reset) under any one of the following conditions: a) a valid password is received on any of the allowed attempts b) the Attempt Count Reset Interval expires c) the user is re-enabled using the Set User Access command d) the user is automatically re-enabled when the User Lockout Interval expires. e) the Bad Threshold number parameter value is re-written or changed The Set User Access command is used to re-enable the user for the Channel. byte 1 [7:1] - reserved [0] - 0b = do not generate an event message when the user is disabled. 1b = generate a Session Audit sensor "Invalid password disable" event message. byte 2 7:0 - Bad Password Threshold number. byte 3:4 15:0 - Attempt Count Reset Interval. The interval, in tens of seconds, for which the accumulated count of bad password attempts is retained before being automatically reset to zero. The interval starts with the most recent bad password attempt for the given username on the channel. This interval is allowed to reset if a BMC power cycles or re-initialization occurs while the interval is being counted. 0000h = Attempt Count Reset Interval is disabled. The count of bad password attempts is retained as long as the BMC remains powered and is not reinitialized. byte 5:6 15:0 - User Lockout Interval. The interval, in tens of seconds, that the user will remain disabled after being disabled because the Bad Password Threshold number was reached. The user is automatically re-enabled when the interval expires. Note that this requires the BMC implementation to track that the user was disabled because of a Bad Password Threshold. This interval is allowed to be restarted if a BMC power cycle or re-initialization occurs while the interval is being counted. Note that this requires an internal non-volatile setting to be maintained that tracks when a particular user has been temporarily disabled due to the Bad Password Threshold. This is required to distinguish a user that was disabled automatically from a user that is intentionally disabled using the Set User Access command. 0000h = User Lockout Interval is disabled. If a user was automatically disabled due to the Bad Password threshold, the user will remain disabled until re-enabled via the Set User Access command.
OEM Parameters	192 : 255	This range is available for special OEM configuration parameters. The OEM is identified according to the Manufacturer ID field returned by the *Get Device ID* command.

Table 25-4, Serial/Modem Configuration Parameters

Parameter	#	Parameter Data (non-volatile unless otherwise noted)[1]
Set In Progress (volatile)	0	data 1 - This parameter is used to indicate when any of the following parameters are being updated, and when the updates are completed. The bit is primarily provided to alert software than some other software or utility is in the process of making changes to the data. …

…

Bad Password Threshold (optional)	54	Sets/Gets the Bad Password Threshold. If implemented and non-zero, this value determines the number of sequential bad passwords that will be allowed to be entered for the identified user before the user is automatically disabled from access on the channel. For example, a value of 3 indicates that 3 sequential attempts are allowed for the given username on the particular channel. If the password for the third attempt is not correct, the user will be disabled for the channel. If this value is zero (00h) then there is no limit on bad passwords. The effect of the disable is the same as if a Set User Access command were used to remove the user's access from the channel. Bad password attempts are tracked according to individual username on a per channel basis. (Thus, a given username may be disabled on one channel, but still enabled on another) Bad password attempts are not counted if integrity check or other session parameters, such as session ID, sequence number, etc. are invalid. That is, bad password attempts are not counted if there are any other errors that would have caused the login attempt to be rejected even if the password was valid. The count of bad password attempts is retained as long as the BMC remains powered and is not reinitialized. Counting automatically starts over (is reset) under any one of the following conditions: a) a valid password is received on any of the allowed attempts b) the Attempt Count Reset Interval expires c) the user is re-enabled using the Set User Access command d) the user is automatically re-enabled when the User Lockout Interval expires. e) the Bad Threshold number parameter value is re-written or changed The Set User Access command is used to re-enable the user for the Channel. byte 1 [7:1] - reserved [0] - 0b = do not generate an event message when the user is disabled. 1b = generate a Session Audit sensor "Invalid password disable" event message. byte 2 7:0 - Bad Password Threshold number. byte 3:4 15:0 - Attempt Count Reset Interval. The interval, in tens of seconds, for which the accumulated count of bad password attempts is retained before being automatically reset to zero. The interval starts with the most recent bad password attempt for the given username on the channel. This interval is allowed to reset if a BMC power cycles or re-initialization occurs while the interval is being counted. 0000h = Attempt Count Reset Interval is disabled. The count of bad password attempts is retained as long as the BMC remains powered and is not reinitialized. byte 5:6 15:0 - User Lockout Interval. The interval, in tens of seconds, that the user will remain disabled after being disabled because the Bad Password Threshold number was reached. The user is automatically re-enabled when the interval expires. Note that this requires the BMC implementation to track that the user was disabled because of a Bad Password Threshold. This interval is allowed to be restarted if a BMC power cycle or re-initialization occurs while the interval is being counted. Note that this requires an internal non-volatile setting to be maintained that tracks when a particular user has been temporarily disabled due to the Bad Password Threshold. This is required to distinguish a user that was disabled automatically from a user that is intentionally disabled using the Set User Access command. 0000h = User Lockout Interval is disabled. If a user was automatically disabled due to the Bad Password threshold, the user will remain disabled until re-enabled via the Set User Access command.
OEM Parameters	192:255	This range is available for special OEM configuration parameters. The OEM is identified according to the Manufacturer ID field returned by the *Get Device ID* command.

Table 42-3, Sensor Type Codes

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
			...
Session Audit	2Ah	00h	Session Activated
		01h	Session Deactivated
		02h	Invalid Username or Password An Invalid Username or Password was received during the session establishment process.
		03h	Invalid password disable. A user's access has been disabled due to a series of bad password attempts. This offset can be used in conjunction with the Bad Password Threshold option. Refer to the LAN or serial/modem configuration parameter for 'Bad Password Threshold' for more information. *The Event Data 2 & 3 fields can be used to provide an event extension code for the preceding offsets, with the following definition:* Event Data 2 ...

Table H-1, Sub-function Number Assignments

LAN Device Commands			
Set LAN Configuration Parameters		Transport	01h
reserved / unspecified	0		
Set for channel 1	1		
...			
Set for channel Bh	24		
Write Parameter 25 (Destination Address VLAN TAGs)	25		
Write Parameter 26 (Bad Password Threshold)	26		
...			
Serial/Modem Device Commands			
Set Serial/Modem Configuration		Transport	10h
reserved / unspecified	0		
Set for channel 1	1		
...			
Set for channel Bh	33		
Write Parameter 54 (Bad Password Threshold)	34		

E445 Addendum - Table 5-1, Network Function Codes, add Group Extension

A group extension value is added for the VITA Standards Organization as follows:

Value(s)	Name	Meaning	Description
00, 01	Chassis	Chassis Device Requests and Responses	00h identifies the message as a command/request and 01h as a response, relating to the common chassis control and status functions.
...			
2Ch, 2Dh	Group Extension	Non-IPMI group Requests and Responses	The first data byte position in requests and responses under this network function identifies the defining body that specifies command functionality. Software assumes that the command and completion code field positions will hold command and completion code values. The following values are used to identify the defining body: 00h** PICMG - PCI Industrial Computer Manufacturer's Group. (www.picmg.com) 01h DMTF Pre-OS Working Group ASF Specification (www.dmtf.org) 02h Server System Infrastructure (SSI) Forum (www.ssiforum.org) 03h VITA Standards Organization (VSO) (www.vita.com) DCh DCMI Specifications (www.intel.com/go/dcmi) all other Reserved When this network function is used, the ID for the defining body occupies the first data byte in a request, and the second data byte (following the completion code) in a response.

E445 Errata - Table 23-4, LAN Configuration Parameters

Bits 4:3 for the 802.1q parameter in the LAN Configuration Parameters were not described. The bits are reserved. This is corrected as follows:

Table 23-4, LAN Configuration Parameters

Parameter	#	Parameter Data (non-volatile unless otherwise noted)[1]
...		
802.1q VLAN Priority	21	data 1 [7:53] - reserved [2:0] - Value for Priority field of 802.1q fields. Ignored when VLAN ID enable is 0b (disabled) - See *802.1q VLAN ID* parameter, above. Setting is network dependent. By default, this should be set to 000b.

Last Page