

- 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 1

6/1/04

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

INTELLECTUAL PROPERTY DISCLAIMER

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

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

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

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

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

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

Contents

Introduction.....	4
Errata Numbers.....	4
6/1/04 Addenda, Errata, and Clarifications	5
E329 Errata - Table 13-8, RMCP/RMCP+ Packet Format for IPMI via Ethernet	5
E330 Clarification - Table 42-3, Sensor Type Codes	5
E331 Clarification/Typo - Table 22 18, Cipher Suite Record Format.....	5
E332 Errata - Table 44-11, Command Number Assignments and Privilege Levels.....	6
E333 Addendum - Table 43 15, Sensor Unit Type Codes.....	6
E334 Typo - Table 44-11, Command Number Assignments and Privilege Levels	7
E335 Errata and Clarification - Table 3-1, Required BMC Functions	7
E336 Errata - Table 44-11, Command Number Assignments and Privilege Levels.....	7
E337 Clarification - Section 32.3, OEM SEL Record - Type E0h-FFh.....	8
E338 Clarification - Section 22.10, Get BT Interface Capabilities Command - applies to IPMI v1.5 and v2.0	8
E339 Addendum - Table 25-4, Serial/Modem Configuration Parameters	9
E340 Clarification - Figure 9-8, Aborting KCS Transactions in-progress and/or Retrieving KCS Error Status.....	9
E341 Addendum - Table 30-9, Alert Immediate Command, and Table H-1, Sub-function Number Assignments	11
E342 Addendum - Table 42-3, Sensor Type Codes	12
E344 Errata - Table 36-3, Sensor Type Codes, Missing Errata E256	12
E345 Errata - Table 17-1, PEF Action Priorities.....	13
E346 Clarification - Section 13.31.4, Integrity Algorithms, Table 22-30, Set Channel Security Keys....	13
E347 Addendum and Clarification - Table 22-17, Get Channel Cipher Suites Command	14
E348 Typos.....	15
E349 Errata - Table 22-12, Get System Interface Capabilities Command	15
E350 Errata and Clarification - Table 13-21, xRC4-Encrypted Payload Fields.....	15
E351 Errata - Table C3-1, Service Processor Management Interface Description Table Format (applies to IPMI v1.5 & v2.0)	16
E352 Addendum - Table 43-13, Entity ID Codes	16
E353 Addendum - (applies to IPMI v1.5 only).....	17
E354 Clarification - Table 43-13, Entity ID Codes	17
E355 Clarification - Table 22-30, Set Channel Security Keys Command.....	17
E356 Errata - Section 6.12.8, Session Sequence Numbers	18
E357 Addendum and Clarification - Table 36-3, Sensor Type Codes	20
E358 Addendum and Clarification - Table 30-9, Alert Immediate Command.....	21
E359 Addendum - Section 17.7, Event Filter Table and Table 30-2, Get PEF Capabilities Command ..	23
E360 Addendum - Section 21, Firmware Firewall & Command Discovery Commands.....	25
E361 Typos and Clarifications - Section 13.28, Authentication, Integrity, and Confidentiality Algorithm Numbers	30
E362 Typos and Clarifications - Section 13.31, RMCP+ Authenticated Key-Exchange Protocol (RAKP)30	
E366 Errata - Table 22-35, Set User Password Command.....	33

Introduction

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

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

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

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

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

Errata Numbers

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

6/1/04 Addenda, Errata, and Clarifications

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

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

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

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

E330 Clarification - Table 42-3, Sensor Type Codes

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

Table 42-3, Sensor Type Codes

Sensor Type	Sensor Type Code	Sensor-specific Offset	Event
Physical Security (Chassis Intrusion)	05h	00h	General Chassis Intrusion
		01h	Drive Bay intrusion
		02h	I/O Card area intrusion
		03h	Processor area intrusion
		04h	LAN Leash Lost (system is unplugged from LAN) <i>The Event Data 2 field can be used to identify which network controller the leash was lost on where 00h corresponds to the first (or only) network controller.</i>
		05h	Unauthorized dock 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	44b	<p>Start Of RecordThis field starts off with either a C0h or C1h "Start of Record" byte, depending on whether the Cipher Suite is a standard Cipher Suite ID or an OEM Cipher Suite, respectively</p> <p><u>Byte 1:</u> [57:0] = 1100_0000b. Start of Record, Start of record followed by Cypher Suite ID tagStandard Cipher Suite. Data following C0h (1100_0000b) tagstart of record byte: Byte 4-2 - 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.</p> <p>[5:0] = 1100_0001b. Start of Record, Start of record followed by Cipher Suite ID + OEM IANA tagOEM Cipher Suite. Data following C1h (1100_0001) tagstart of record byte: Byte 4-2 - OEM Cipher Suite ID See Table 22-19, Cipher Suite IDs.</p> <p><u>Byte 23:4-5 - OEM IANA</u> Least significant byte first. 3-byte IANA for the OEM or body that defined the Cipher Suite.</p>

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

Incorrect privilege level definition for Activate/Deactivate Payload commands.

Table 44-11, Command Number Assignments and Privilege Levels

	section	NetFn	CMD	C	U	O	A
	...						
Activate Payload	24.1	App	48h	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	<p>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. <u>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.</u></p>
---------------	---	--

...

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							
...							
<u>Get NetFn Support</u>	<u>21.2</u>	<u>App</u>	<u>09h</u>		<u>X</u>		
<u>Get Command Support</u>	<u>21.3</u>	<u>App</u>	<u>0Ah</u>		<u>X</u>		
<u>Get Command Sub-function Support</u>	<u>21.4</u>	<u>App</u>	<u>0Bh</u>		<u>X</u>		
<u>Get Configurable Commands</u>	<u>21.5</u>	<u>App</u>	<u>0Ch</u>		<u>X</u>		
<u>Get Configurable Command Sub-functions</u>	<u>21.6</u>	<u>App</u>	<u>0Dh</u>		<u>X</u>		
<u>unassigned</u>	<u>:</u>	<u>App</u>	<u>0Eh-0Fh</u>	<u>:</u>	<u>:</u>	<u>:</u>	<u>:</u>
<u>Set Command Enables</u>	<u>21.7</u>	<u>App</u>	<u>60h</u>				<u>X</u>
<u>Get Command Enables</u>	<u>21.8</u>	<u>App</u>	<u>61h</u>		<u>X</u>		
<u>Set Command Sub-function Enables</u>	<u>21.9</u>	<u>App</u>	<u>62h</u>				<u>X</u>
<u>Get Command Sub-function Enables</u>	<u>21.10</u>	<u>App</u>	<u>63h</u>		<u>X</u>		

...

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 <u>message_size</u> in bytes. (1 based.) ^[1]
	4	Output (response) buffer <u>message_size</u> 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

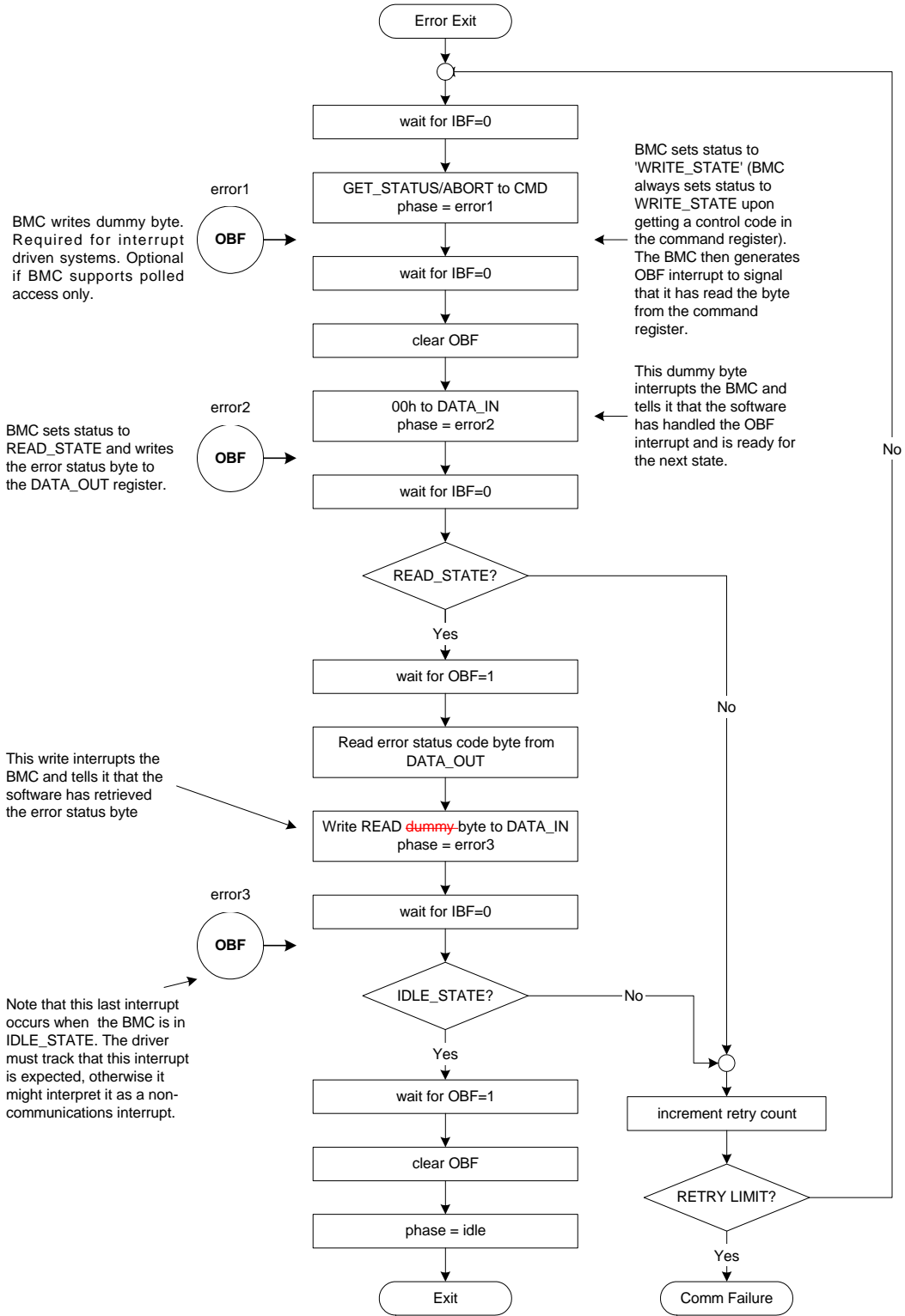
Parameter	#	Parameter Data (non-volatile unless otherwise noted) ⁽¹⁾
...		
<u>Bit Rate Support (READ ONLY, optional)</u>	<u>50</u>	<u>This parameter returns a read-only bitfield indicating which bit rates are supported for this serial channel.</u> <u>data 1 - Bit Rate Support</u> <u>[7:6] - reserved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

...

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

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

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



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

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

Table 30-9, Alert Immediate Command

Byte data field

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

Table H-1, Sub-function Number Assignments

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

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

...

E342 Addendum - Table 42-3, Sensor Type Codes

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

Table 42-3, Sensor Type Codes

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

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

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

Table 36-3, Sensor Type Codes

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

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.
<u>ICMB Group Control</u>	<u>5</u>	<u>(optional) Performs ICMB group control operation according to settings from the Group Control Table parameter in the PEF Configuration Parameters.</u>
Send Alert	<u>56</u>	(mandatory if alerting is supported) Send alerts in order based on the selected Alert Policy. Alert actions will be deferred until after the power down has completed. There is an additional prioritization within alerts being sent: based on the Alert Policy Table entries for the alert. This is described further in <i>Section 17.11, Alert Policy Table</i> .
OEM	OEM	(optional) Priority determined by OEM.

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

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

13.31.4 Integrity Algorithms

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

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

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

Table 22-30, Set Channel Security Keys Command

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

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

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

Table 22-17, Get Channel Cipher Suites Command

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

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

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

E348 Typos

- Cross references to “Table 37-12, Entity ID Codes” should be “Table 43-14, Entity ID Codes”
- Cross references to “Table 37-11, IPMB/I 2C Device Type Codes” should be “Table 43- 12, IPMB/I 2C Device Type Codes”
- Formatting error: A bad cross-reference made it appear as if the last sentence of the first paragraph of section 12.12, Discovering SSIF, was truncated after “(see”. The corrected formatting is:
...the existence and slave address of the SSIF (see [Appendix C1 - Locating IPMI System Interfaces via SM BIOS Tables](#)).
- Bad cross-reference formatting in section 13.32.2, Encryption with AES
- Table 13-8, RMCP/RMCP+ Packet Format for IPMI via Ethernet didn't consistently list field sizes in all columns for parts of Ethernet packet and IP Header that are common across IPMI v1.5/RMCP, ASF/RMCP, and IPMI v2.0/RMCP+
- Formatting: IPMI Session Header in tables 13-9 and 13-10 were not shaded.
- Table 13-11, ‘Maximum Requested Privilege Level’ ala IPMI v1.5. → ‘Maximum Requested Privilege Level’ ~~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 <u>or</u> SMIC	
3	[7:3] - reserved [2:0] - System Interface Version 000b = version 1 (conformant with KCS <u>or</u> SMIC interface <u>as</u> 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 (<u>not encrypted</u>)	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 <u>the offset for the data and decrypt from that point.</u>
	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	<u>Payload data. Encrypted per xRC4 algorithm.</u>
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	4	36	This field must always be 01h to be compatible with any software that implements previous versions of this spec.
Interface Type	1	37 <u>36</u>	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 <u>System Firmware (e.g. BIOS / EFI)</u>

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

Request Data	byte	data field
	1	Channel Number [7:4] - reserved [3:0] - Channel Number <u>(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.)</u>

	3	Key ID [7:0] - key ID. 00h = RMCP+ "KR" key <u>(20 bytes)</u> . 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 <u>for each channel</u> to see if it matches and is unlocked <u>locked</u> . 01h = RMCP+ "KG" key <u>(20 bytes)</u> . "KG" key acts as a value that is used for key exchange for the overall channel. This key is not lockable <u>cannot be locked. This is to ensure - in order to enable</u> a password/key configuration utility to can <u>can</u> 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. <u>KG must be individually settable on each channel that supports RMCP+.</u> <u>all other = reserved</u>

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
			...
<u>Management Subsystem Health</u>	<u>28h</u>	<u>00h</u>	<u>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).)</u>

		<p>01h</p> <p>02h</p> <p>03h</p> <p>04h</p> <p>05h</p>	<p>controller access degraded or unavailable <u>(The ability to access the controller has been degraded, or access is unavailable, but the party that is doing the monitoring cannot determine which.)</u></p> <p>management controller off-line <u>(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.)</u></p> <p>management controller unavailable <u>(controller cannot be accessed because of an error condition)</u></p> <p>Sensor failure <u>(the sensor is known to be in error. It may still be accessible by software)</u></p> <p><u>Event Data 2</u> <u>The Event Data 2 field for this offset can be used to provide additional information on the type of failure with the following definition:</u> <u>[7:0] - Sensor Number. Number of the failed sensor corresponding to event offset 04h or 00h.</u></p> <p>FRU failure</p> <p><u>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:</u></p> <p><u>Event Data 2</u> <u>[7] - logical/physical FRU device</u> <u> 0b = device is not a logical FRU Device</u> <u> 1b = device is logical FRU Device (accessed via FRU commands to mgmt. controller)</u> <u>[6:5] - reserved.</u> <u>[4:3] - LUN for Master Write-Read command or FRU Command. 00b if device is non-intelligent device directly on IPMB.</u> <u>[2:0] - Private bus ID if bus = Private. 000b if device directly on IPMB, or device is a logical FRU Device.</u></p> <p><u>Event Data 3</u> <u>For LOGICAL FRU DEVICE (accessed via FRU commands to mgmt. controller):</u> <u>[7:0] - FRU Device ID within controller that generated the event.FFh = reserved.</u> <u>For non-intelligent FRU device:</u> <u>[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.</u> <u>[0] - reserved.</u></p>
--	--	--	---

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

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

Table 30-9, Alert Immediate Command

	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.
	...	
<i>The following "Platform Event Parameters" (bytes 4:11) can be used to fill in the corresponding event data fields of a Platform Event Trap. When supported, all bytes (4:11) must be supplied. Implementation of this capability is OPTIONAL but highly recommended for IPMI v2.0 implementations. See Table 29-5, Event Request Message Fields, for specification of the individual fields.</i>		
	<u>4</u>	<u>Generator ID</u>
	<u>5</u>	<u>EvMRev</u>
	<u>6</u>	<u>Sensor Type</u>
	<u>7</u>	<u>Sensor #</u>
	<u>8</u>	<u>Event Dir Event Type</u>
	<u>9</u>	<u>Event Data 1</u>
	<u>10</u>	<u>Event Data 2</u>
	<u>11</u>	<u>Event Data 3</u>
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. <u>83h = Platform Event Parameters (4:11) not supported.</u>

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		
<u>Platform Event Parameters</u>	<u>8</u>		

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] - <u>1b = OEM Event Record Filtering supported</u> [6] - reserved Action Support [5] - 1b = diagnostic interrupt [4] - 1b = OEM action [3] - 1b = power cycle [2] - 1b = reset [1] - 1b = power down [0] - 1b = Alert
	4	Number of event filter table entries (1 based)

17.7 Event Filter Table

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

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

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

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

Table 17-2, Event Filter Table Entry

Byte	Field	Description
1	Filter Configuration	<p>[7] - 1b = enable filter 0b = disable filter</p> <p>[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.</p> <p>[4:0] - reserved<u>Record type</u> <u>0h = Record 02h</u> <u>1h = OEM Record C0h-DFh (timestamped. Includes Mfr ID as first three non-timestamp data bytes in record.)</u> <u>2h = OEM Record E0h-FFh (non-timestamped. Mfr ID from Get Device ID command for BMC.)</u></p>
...		
5	Generator ID Byte 1 / <u>OEM Record Type</u>	Slave Address or Software ID from Event Message, <u>or OEM Event Record Type⁽¹⁾</u> . FFh = match any
6	Generator ID Byte 2 / <u>OEM data 1</u>	Channel Number / LUN to match, <u>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)</u> FFh = match any see section 32, <i>SEL Record Formats</i> .
7	Sensor Type / <u>OEM data 2</u>	Type of sensor, <u>or 2nd OEM record data byte following the timestamp.</u> FFh = match any
8	Sensor # / <u>OEM data 3</u>	FFh = match any
9	Event Trigger (Event/Reading Type) / <u>OEM data 4</u>	FFh = match any
10, 11	Event Data 1 Event Offset Mask / <u>OEM data 5:6</u>	<p>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.</p> <p>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.</p> <p><u>data 1</u> 7:0 - mask bit positions 7 to 0, respectively. <u>data 2</u> 15:8 - mask bit positions 15 to 8, respectively.)</p> <p><u>For OEM record matching:</u> <u>data 1 = OEM data 5 (FFh = match any)</u> <u>data2 = OEM data 6 (FFh = match any)</u></p>

12	Event Data 1 AND Mask / OEM Data 7 AND Mask	<p>This value is applied to the entire Event Data 1 byte. The field is Used to indicate 'wildcarded' or 'compared' bits. This field must be used in conjunction with Compare 2. To match any Event Data field value, just set the corresponding AND Mask, Compare 1, and Compare 2 fields to 00h. (See <i>Section 17.8, Event Data 1 Event Offset Mask</i> for more information). Note that the Event Data 1 AND mask, Compare 1 mask, and Compare 2 masks will typically be set to wild-card the least significant of Event Data 1 in order to allow the Event Data 1 Event Mask field to determine matches to the event offset.</p> <p>Bits 7:0 all have the following definition:</p> <ul style="list-style-type: none"> 0 = Wildcard bit. (drops this bit position in the Event Data byte out of the comparison process) Corresponding bit position must be a 1 in Compare 1, and a 0 in Compare 2. (Note - setting a 0 in this bit, a 1 and Compare 1 and a 1 in Compare 2 guarantees that you'll <i>never</i> have a match.) 1 = use bit for further 'exact' or 'non-exact' comparisons based on Compare 1 and Compare 2 values.
13	Event Data 1 Compare 1 / OEM Event Data 7 Compare 1	<p>Used to indicate whether each bit position's comparison is an exact comparison or not. (See <i>Section 17.8, Event Data 1 Event Offset Mask</i> for more information). Here, 'test value' refers to the Event Data value <i>after</i> the AND Mask has been applied.</p> <p>Bits 7:0 all have the following definition:</p> <ul style="list-style-type: none"> 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	<p>(See <i>Section 17.8, Event Data 1 Event Offset Mask</i> for more information). Here, 'test value' refers to the Event Data value <i>after</i> the AND Mask has been applied.</p> <p>Bits 7:0 all have the following definition:</p> <ul style="list-style-type: none"> 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

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

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
	(4)	<i>For Network Function = 2Ch:</i> <u>Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)</u>
(4:6)	<i>For Network Function = 2Eh:</i> <u>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)</u>	
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.
	...	
	(5)	<i>For Network Function = 2Ch:</i> <u>Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1, Network Function Codes)</u>
(6:8)	<i>For Network Function = 2Eh:</i> <u>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)</u>	
IPMI Response Data	1	Completion Code
...		

Table 21-5, Get Configurable Commands Command

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

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

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

Table 21-7, Set Command Enables Command

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

Table 21-7, Set Command Enables Command

IPMI Request Data	1	Channel Number [7:4] - reserved [3:0] - channel number. 0h-7h, Fh = channel numbers Eh = retrieve information for channel this request was issued on.
	...	
<i>For Network Function = 2Ch:</i>		
(19)	<u>Defining body code (See description for Network Function 2Ch/2Dh in Table 5-1. Network Function Codes)</u>	
<i>For Network Function = 2Eh:</i>		
(19:21)	<u>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)</u>	
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]
<u>Get OEM NetFn IANA Support</u>	<u>21.11</u>	<u>O^[1,3,4]</u>

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
...			
<u>Set Command Enables</u>		<u>App</u>	<u>60h</u>
<u>Get Command Enables</u>		<u>App</u>	<u>61h</u>
<u>Set Command Sub-function Enables</u>		<u>App</u>	<u>62h</u>
<u>Get Command Sub-function Enables</u>		<u>App</u>	<u>63h</u>
<u>Get OEM NetFn IANA Support</u>		<u>App</u>	<u>64h</u>
...			

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 use the HMAC-SHA1 integrity algorithm defined in [RFC2104] in the RAKP authentication process, and the HMAC-SHA1-96 integrity algorithm defined in [RFC2404] for data integrity.

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

...

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

Message 1: Remote Console -► Managed System

SID_C, R_M, Role_M, ULength_M, < UName_M >

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

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

Message 2: Managed System -► Remote Console

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

Where:

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

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

After receiving RAKP Message 2, the remote console verifies that the value SID_M is active and that GUID_C matches the managed system that the remote console is expecting to communicate with. The remote console then validates the 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 $\text{R}_M, \text{R}_C, \text{Role}_M, \text{ULength}_M$, and (optional) UName_M using 160-bit key K_G (note - no truncation).

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

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

Then the remote console sends to the managed system as Message 3 the value SID_C and (for the RAKP-HMAC-SHA1 algorithm) the HMAC per [RFC2404/RFC2104] of the values ($\text{R}_C, \text{SID}_M, \text{Role}_M, \text{ULength}_M, \langle \text{UName}_M \rangle$) generated using key $\text{K}[\text{UID}]$ selected by the username, UName_M , and role Role_M .

Message 3: Remote Console -► Managed System

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

Where $\text{HMAC}_{\text{K}[\text{UID}]}$ ($\text{R}_C, \text{SID}_M, \text{Role}_M, \text{ULength}_M, \langle \text{UName}_M \rangle$) represents the value for the Key Exchange Authentication Code for RAKP Message 3. After receiving Message 3, the managed system verifies that the value SID_C is active and then validates the 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 $\text{R}_M, \text{R}_C, \text{Role}_M, \text{ULength}_M$, and (optional) UName_M using 160-bit key K_G (note - no truncation, and that $\text{K}[\text{UID}]$ is used in place of K_g if 'one-key' logins are being used. See 13.28.4, Integrity Algorithms).

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

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

Message 4: Managed System -► Mgmt Console

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

Where $\text{HMAC}_{\text{K}[\text{UID}]}$ ($\text{R}_C, \text{SID}_M, \text{Role}_M, \text{ULength}_M, \langle \text{UName}_M \rangle$) represents the value in the Integrity Check Value field for RAKP Message 4. After receiving Message 4, the management console verifies that the value SID_M is active and then validates the 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

Request Data	byte	data field
	1	<p>User ID.</p> <p>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.</p> <p>...</p> <p>The 'test password' operation can be used to determine whether a password has been stored as 16-bytes or 20-bytes.</p> <p>[7:6] - reserved.</p> <p><u>[7] - password size</u></p> <p><u>1b = set 20-byte user password/key.</u></p> <p><u>0b = set 16-byte user password/key (IPMI v1.5 backward compatible)</u></p> <p>[6] - reserved.</p> <p>[5:0] - User ID. 000000b = reserved. (User ID 1 is permanently associated with User 1, the null user name).</p>

...

Last Page