What is IPMI

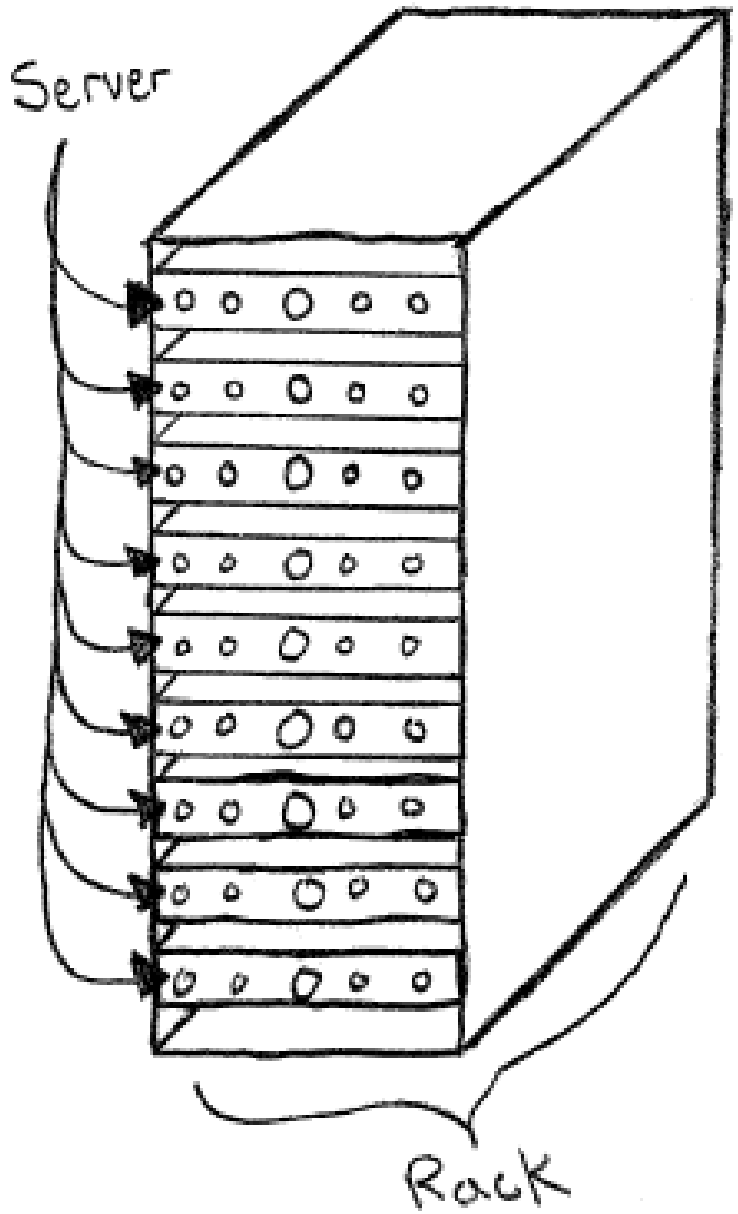
10/13/2022

Basics

First, let us go over the parts from the outside in that make up the whole to help in understanding what is being talked about.

The parts starting from the outside in are:

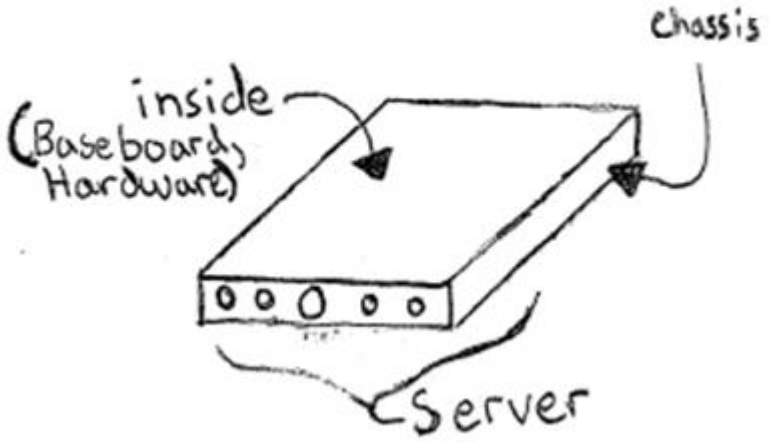
Rack	Server	Baseboard and Hardware



Rack

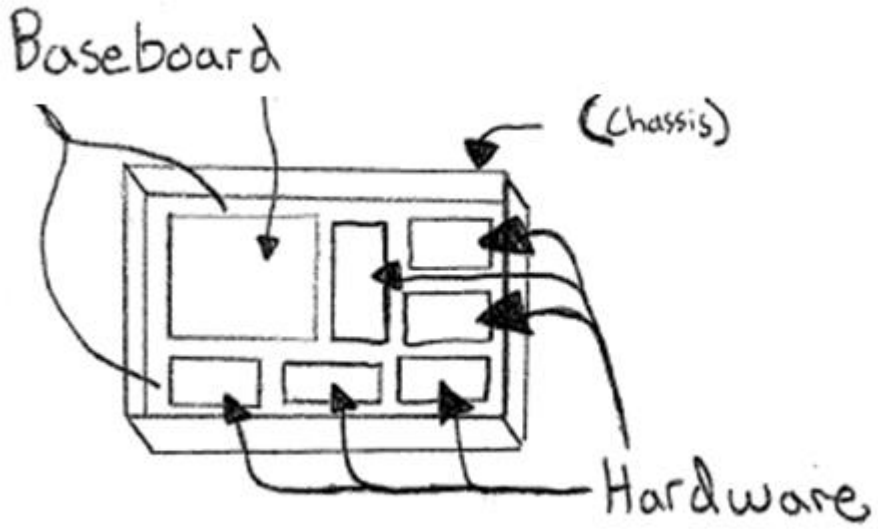
- The rack consists of servers connected together and placed into a cabinet.

Server



- The server consists of the Baseboard and Hardware inside of it, and the Chassis

Baseboard and Hardware



- The Baseboard and Hardware are inside of the chassis of the server. The Baseboard will hold the Baseboard Management Controller (BMC) and the Hardware will consist of Hard Disk Drives, Fans, Sensors, etc.

IPMI

- Now that we know the parts from the top to bottom and an understanding of the parts, we can go over IPMI.
- IPMI, Intelligent Platform Management Interface, is for extending Platform Management between boards in the main chassis and to other chassis.
- Platform Management refers to the monitoring and control functions that are built into the platform with the purpose of monitoring the health of the system.

IPMI Platforms

- There are different platforms with Platform Management that consist of:
  - Monitoring
  - Recovery
  - Logging and Alerting
  - Inventory

IPMI Platforms (Monitoring)

- The Monitoring platform “includes monitoring elements such as system temperatures, voltages, fans, power supplies, bus errors, system physical security, etc.”.

IPMI Platforms (Recovery)

- The Recovery platform “includes automatic and manually driven recovery capabilities such as local or remote system resets and power on/off operations”.

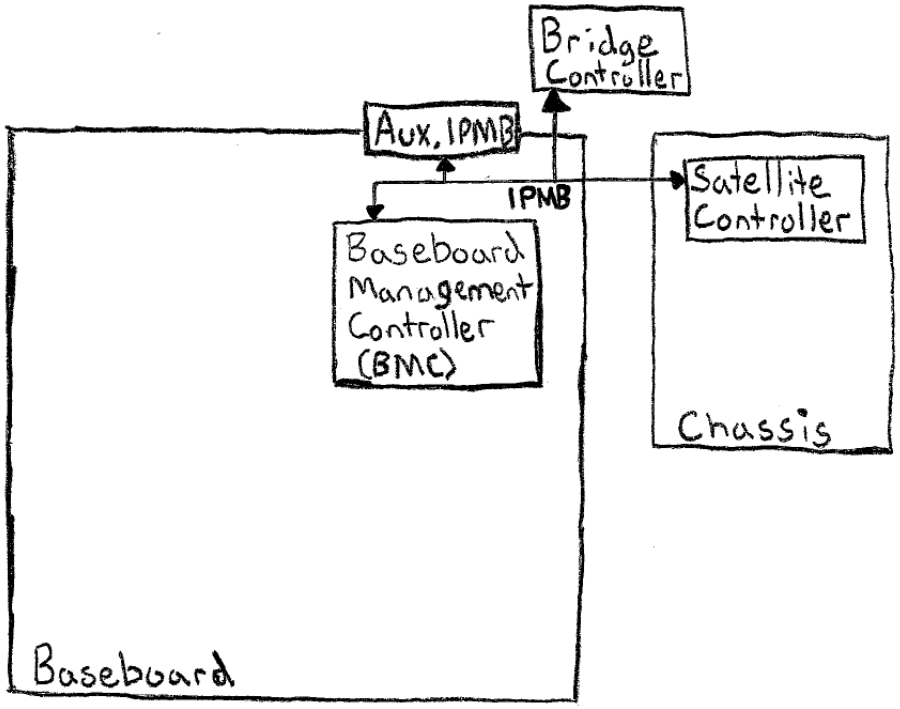
IPMI Platforms (Logging and Alerting)

- The Logging and Alerting platform “includes the logging of abnormal or ‘out-of-range’ conditions for later examination and alerting where the platform issues the alert without aid of run-time software”.

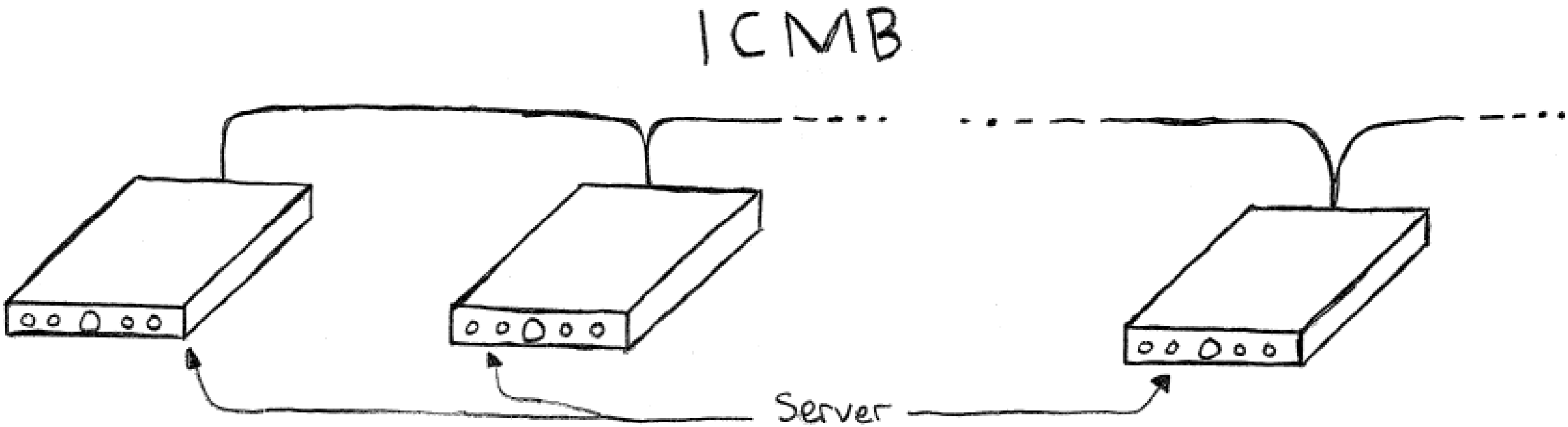
IPMI Platforms (Inventory)

- The Inventory platform “includes inventory information that can help identify a failed hardware unit”.

IPMB



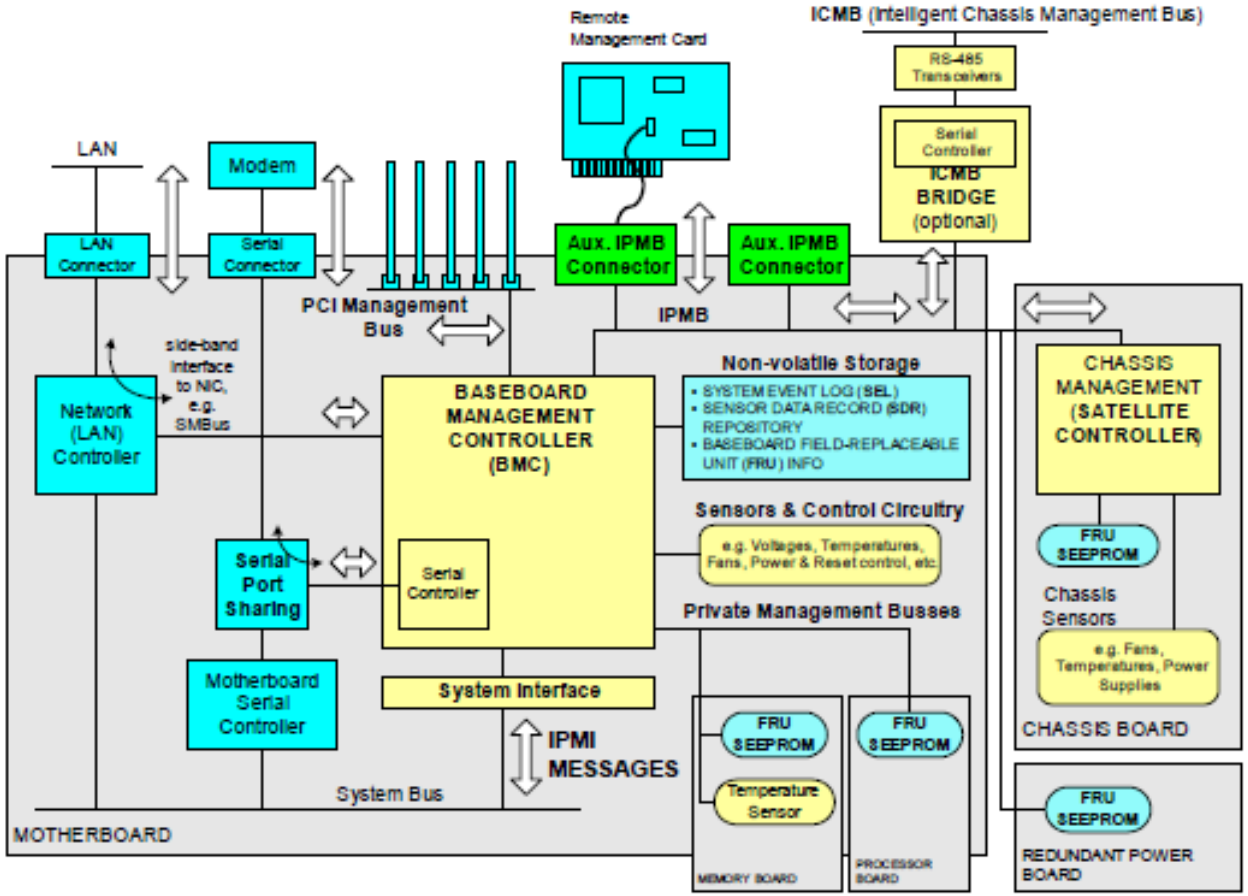
- IPMB, Intelligent Platform Management Bus, is connected between major management controllers that provides communication to and from the baseboard that holds the BMC.
- Overall, the IPMB connects the BMC baseboard together.



- The ICMB, Intelligent Chassis Management Bus, connects chassis together, and "Extends IPMI Intelligent Platform Management to multiple host and peripheral chassis"
- "The ICMB has been targeted to support network/buses with up to 42 chassis [64 chassis maximum]"
- Overall, the ICMB connects the Chassis together.

IPMI Diagram

- BMC -> Brain of IPM and the Language Interpreter between software and hardware.
- IPMB -> Network connection communication within the Chassis.
- ICMB -> Network connection communication outside of the Chassis.



IPMI Messaging

- IPMI Messaging is messaging through platform managements like IPMB and ICMB to the BMC (The Brain).
- IPMI Messaging uses request/response protocol where request messages are ‘commands’.
- Commands are grouped into sets that have Network Function Codes, Command, and Optional Data Fields.
- Response Messages have Network Function Codes, Command, Optional Data Fields, and Completion Codes Field.

IPMI Messaging Components

- **Networking Function (NetFn):** Field that identifies the functional class of the message[1].
- **Request/Response Identifier:** Field that differentiates messages from a Request/Response[1].
- **Requestors ID:** Identifies the source of the Request[1].
- **Responder's ID:** Field that identifies the Responder to the Request[1].
- **Command:** Message in a one-byte command[1].
- **Data:** This field contains additional parameter for a Request/Response[1].

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.
02*, 03*	Bridge	Bridge Requests and Responses	02h (request) or 03h (response) identifies the message as containing data for bridging to the next bus. This data is typically another message, which may also be a bridging message. This function is present only on bridge nodes.
04, 05	Sensor /Event	Sensor and Event Requests and Responses	This functionality can be present on any node. 04h identifies the message as a command/request and 05h as a response, relating to the configuration and transmission of Event Messages and system Sensors.
06, 07	App	Application Requests and Responses	06h identifies the message as an application command/request and 07h a response. The exact format of application messages is implementation-specific for a particular device, with the exception of App messages that are defined by the IPMI specifications. Note that it is possible that new versions of this specification will identify new App commands. To avoid possible conflicts with future versions of this specification, it is highly recommended that the OEM/Group network functions be used for providing 'value added' functions rather than the App network function code.
08, 09	Firmware	Firmware Transfer Requests and Responses	The format of firmware transfer requests and responses matches the format of Application messages. The type and content of firmware transfer messages is defined by the particular device.
0A, 0B	Storage	Non-volatile storage Requests and Responses	This functionality can be present on any node that provides non-volatile storage and retrieval services.
0C, 0D	Transport	Media-specific configuration & control	Requests (0Ch) and responses (0Dh) for IPMI-specified messages that are media-specific configuration and operation, such as configuration of serial and LAN interfaces.
0Eh-2Bh	Reserved	-	reserved (30 Network Functions [15 pairs])
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.
2Eh, 2Fh	OEM/Group	OEM/Non-IPMI group Requests and Response	The first three data bytes of requests and responses under this network function explicitly identify the OEM or non-IPMI group that specifies the command functionality. While the OEM or non-IPMI group defines the functional semantics for the cmd and remaining data fields, the cmd field is required to hold the same value in requests and responses for a given operation in order to be supported under the IPMI message handling and transport mechanisms. When this network function is used, the IANA Enterprise Number for the defining body occupies the first three data bytes in a request, and the first three data bytes following the completion code position in a response.
30h-3Fh	Controller-specific OEM/Group	-	Vendor specific (16 Network Functions [8 pairs]). The Manufacturer ID associated with the controller implementing the command identifies the vendor or group that specifies the command functionality. While the vendor defines the functional semantics for the cmd and data fields, the cmd field is required to hold the same value in requests and responses for a given operation in order for the messages to be supported under the IPMI message handling and transport mechanisms.

* Network Functions that are only utilized in systems that incorporate Bridge nodes.
** This organization value was named 'Compact PCI' in revision 1.0 of this document.

IPMI Completion Codes

- Response Messages will have a completion code attached to them.

Code	Definition
	GENERIC COMPLETION CODES 00h, C0h-FFh
00h	Command Completed Normally.
C0h	Node Busy. Command could not be processed because command processing resources are temporarily unavailable.
C1h	Invalid Command. Used to indicate an unrecognized or unsupported command.
C2h	Command invalid for given LUN.
C3h	Timeout while processing command. Response unavailable.
C4h	Out of space. Command could not be completed because of a lack of storage space required to execute the given command operation.
C5h	Reservation Canceled or Invalid Reservation ID.
C6h	Request data truncated.
C7h	Request data length invalid.
C8h	Request data field length limit exceeded.
C9h	Parameter out of range. One or more parameters in the data field of the Request are out of range. This is different from 'Invalid data field' (CCh) code in that it indicates that the erroneous field(s) has a contiguous range of possible values.
CAh	Cannot return number of requested data bytes.
CBh	Requested Sensor, data, or record not present.
CCh	Invalid data field in Request
CDh	Command illegal for specified sensor or record type.
CEh	Command response could not be provided.
CFh	Cannot execute duplicated request. This completion code is for devices which cannot return the response that was returned for the original instance of the request. Such devices should provide separate commands that allow the completion status of the original request to be determined. An Event Receiver does not use this completion code, but returns the 00h completion code in the response to (valid) duplicated requests.
D0h	Command response could not be provided. SDR Repository in update mode.
D1h	Command response could not be provided. Device in firmware update mode.
D2h	Command response could not be provided. BMC initialization or initialization agent in progress.
D3h	Destination unavailable. Cannot deliver request to selected destination. E.g. this code can be returned if a request message is targeted to SMS, but receive message queue reception is disabled for the particular channel.
D4h	Cannot execute command due to insufficient privilege level or other security-based restriction (e.g. disabled for 'firmware firewall').
D5h	Cannot execute command. Command, or request parameter(s), not supported in present state.
D6h	Cannot execute command. Parameter is illegal because command sub-function has been disabled or is unavailable (e.g. disabled for 'firmware firewall').
FFh	Unspecified error.
	DEVICE-SPECIFIC (OEM) CODES 01h-7Eh
01h-7Eh	Device specific (OEM) completion codes. This range is used for command-specific codes that are also specific for a particular device and version. A-priori knowledge of the device command set is required for interpretation of these codes.
	COMMAND-SPECIFIC CODES 80h-BEh
80h-BEh	Standard command-specific codes. This range is reserved for command-specific completion codes for commands specified in this document.
all other	reserved

- The purpose of the completion code is to be able to tell if the command was received or not and if the command is normal.

IPMB Interface

- Using the IPMB, the BMC will act as a controller. This gives system software access to the IPMB. This is done using the ‘Master Write-Read’ command.
- On the IPMB, management controllers and devices are connected to the IPMB and the ‘Master Write-Read’ command gives access to the management controllers and the devices.
- We are then able to send commands from the BMC and receive responses from management controllers and devices on the IPMB.

IPMB Interface (Sending Messages to IPMB from System Software)

IPMB Request Sent using Send Message Command

NetFn (06h = App request)	LUN (00b)	Command (Send Message)	Channel (00h)
Slave address for write (52h = rsSA)	NetFn/rsLUN (04h / 00b = Sensor/Event, LUN 00b)		check 1 (9Eh)
rqSA (20h = BMC)	rqSeq/rqLUN (000001b / 10b, 10b = SMS LUN)	Cmd (00h = Set Event Receiver)	
event receiver slave address (20h = BMC)	event receiver LUN (00h)	check 2 (BAh)	

- Once the message is sent, there will be a response to the message to know the status of it.

- "System Management Software (SMS) can use the BMC for sending and receiving IPMB Messages" (Intel, Hewlett-Packard, NEC, Dell, 2014, p. 70).
- This is an example of using a BMC as an IPMB controller that sends a message out through the IPMB using the 'Send Message' command.
- In the 'Send Message' command, it is also sending an 'Set Event Receiver' command.

Send Message Command Response

NetFn (07h = App response)	LUN (00b)	Command (Send Message)	Completion Code (00h)

IPMB Interface (Sending IPMB Messages to System Software)

- "The IPMB response to a 'Set Event Receiver' command consists of just a Completion Code byte in the data portion of the IPMB message" (Intel, Hewlett-Packard, NEC, Dell, 2014, p. 71).
- In the prior example, the 'Set Event Receiver' is activated to the IPMB device at the address of 52h (this also means 0x52), LUN 00b and to have the receiver address at 20h (the BMC).
- Having the rqLUN set to 10b (BMC SMS LUN) will have the message response sent to the BMC's Receive Message Queue.

Response for Set Event Receiver in Receive Message Queue

NetFn (05h = Sensor/Event Response)		rqLUN (10b = SMS LUN)		Check 1 (CAh)
rsSA (52h)	rqSeq/rsLUN (000001b / 00b)	Cmd (00h = Set Event Receiver)	Completion Code (00h = OK)	Check 2 (AAh)

Get Message Command Response

NetFn (07h = App response)	LUN (00b)	Command (Get Message)	Completion Code (00h)	Channel Number (00h)
NetFn (05h = Sensor/Event Response)		rqLUN (10b = SMS LUN)	Check 1 (CAh)	
rsSA (52h)	rqSeq/rsLUN (000001b / 00b)	Cmd (00h = Set Event Receiver)	Completion Code (00h = OK)	Check 2 (AAh)

- Using the 'Get Message' Command, we are able to read and receive the message that is stored in the BMC's Receive Message Queue.

ICMB Interface

- The ICMB is implemented access by an ICMB Bridge Controller that is connected to the IPMB and the BMC can act as the ICMB Bridge Controller.
- This is done by using a ‘virtual IPMB’ or by having the BMC provide ICMB access from the IPMB.

ICMB Interface ‘Virtual IPMB’

This happens by having the BMC report that the Chassis Bridge Device is not part of the BMC that then tells the software to access the Bridge Device function by using the ‘Send Message’ commands to the Bridge Device via the primary IPMB.[1]

Chassis Bridge Device to ‘Send Message’ command

The BMC then monitors the ‘Send Message’ command to the Bridge Device address, handles them internally instead of routing them to the physical IPMB.[1]

Monitor ‘Send Message’ command

The responses are placed into the ‘Receive Message Queue’.[1]

‘Receive Message Queue’

ICMB Interface ‘Through the IPMB’

Two Slave Addresses

- This can happen by having the BMC to respond to two slave addresses (The BMC address and the Bridge Device address) .

Directly

- This can happen by having the BMC report the BMC address is the Bridge Device address when the ‘Get Chassis Capabilities’ command comes from the IPMB and then have the bridge commands sent directly through the IPMB.

IPMI Tool Example

After flash FRU2 --Pass:

```
ipmitool> raw 6 4
 55 00
```

- This code was ran and returned a passing response.
- The second byte is 0x55 showing that the test passed and there is No error and All Self Test Passed.

BMC Self Test

Command > Ipmitool raw 0x06 0x04

Before flash FRU2 --Fail:

```
ipmitool> raw 6 4
 57 04
```

- This code was ran and returned a failing response.
- The third byte 0x04 is the same as '0000 0100' in binary
- The third least significant bit is a '1' showing that the failure is Internal Use Area of BMC FRU is corrupted.

Get Self Test Results Command

	byte	data field
Request Data	-	-
Response Data	1	Completion Code
	2	55h = No error. All Self Tests Passed. 56h = Self Test function not implemented in this controller. 57h = Corrupted or inaccessible data or devices 58h = Fatal hardware error (system should consider BMC inoperative). This will indicate that the controller hardware (including associated devices such as sensor hardware or RAM) may need to be repaired or replaced. FFh = reserved. all other: Device-specific 'internal' failure. Refer to the particular device's specification for definition.
	3	For byte 2 = 55h, 56h, FFh: 00h For byte 2 = 58h, all other: Device-specific For byte 2 = 57h: self-test error bitfield. Note: returning 57h does not imply that all tests were run, just that a given test has failed. I.e. 1b means 'failed', 0b means 'unknown'. [7] 1b = Cannot access SEL device [6] 1b = Cannot access SDR Repository [5] 1b = Cannot access BMC FRU device [4] 1b = IPMB signal lines do not respond [3] 1b = SDR Repository empty [2] 1b = Internal Use Area of BMC FRU corrupted [1] 1b = controller update 'boot block' firmware corrupted [0] 1b = controller operational firmware corrupted

References

Intel, Hewlett-Packard, NEC, & Dell. (2014, February 11). *IPMI Intelligent Platform Management Interface Specification, Second Generation, v2.0.*

Slaight, T. (1999, February 24). *Intelligent Chassis Management Bus (ICMB) v1.0 Overview.* Retrieved July 15, 2019, from https://www.slideserve.com/bernad/icmb-v1-0-intro-022499.

Davis, L. (2012, February 14). *Intelligent Chassis Management Bus Description and Pinout.* Retrieved July 15, 2019, from http://www.interfacebus.com/ICMB_Intelligent_Chassis_Management_Bus.html.

BMC_IPMI_Self test PDF