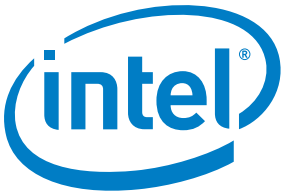


Intel® Rack Scale Design

Firmware Extension Specification
Software v2.2

December 19, 2017

Revision 001

Document Number: 336860-001

Contents

Figures

Tables

Revision History

Revision	Description	Date
001	Initial release.	December 19, 2017

§

1.0 Introduction

This document describes the extended BIOS/ Baseboard Management Controller (BMC) functionalities to support Intel® Rack Scale Design (Intel® RSD) Firmware Extension Specification v2.2 on the Intel® Xeon® Processor Scalable Family platform and other Intel® RSD v2.2 supported platforms.

Section 2.0 of this document describes the Intelligent Platform Management Interface (IPMI) extensions required to support Intel® RSD v2.2.

Section 3.0 of this document outlines specific data structures used in the IPMI extensions where such data structures are not part of a standard specifications already.

1.1 Terminology

Table 1. Terminology

Term	Type	Description	Examples
BMC		Baseboard Management Controller. An independent micro controller that provides platform level manageability functions in a server platform.	
Boolbyte	byte	A byte that can only have a value of 1 or 0.	
Byte	byte	8 bits	
CMI		Cable Management Interface	
CPM		Cable Management Interface	
FPGA		Field Programmable Gate Array	
GUID	intx4	Globally Unique (Arbitrary) ID (128 bits) xxxxx-xxxx-xxxxxxxxxx	`21EC2020-3AEA-4069-A2DD-08002B30309D`
Index	int	Unique int of Order or Location (Preferred to *always* start with 1, leave 0 reserved)	
int	int	32-bit int	
IPMB		Intelligent Platform Management Bus. Bus protocol used for communication between various IPMI devices.	
IPMI		Intelligent Platform Management Interface. Out of Band management interface used by server platforms to communicate via IPMB protocol.	
KCS		Keyboard Controller Style – An IPMI system interface to communicate with BMC	
LLDP		Link Layer Discovery Protocol	
LPC		Low Pin Count interface for accessing BMC	
MBP		Management Backplane – a rack-mount board that the trays dock with in the back of the rack. Each segment of the BDC MPB is 8U high with two vertically daisy-chained in a rack.	

Term	Type	Description	Examples
ME		Intel® Manageability Engine.	
Module		A hot-swappable sub-unit of a tray that is often a server. This is also known as a “Sled”. This could also be a multi-unit baseboard that contains uServers.	
Node		A Xeon or uServer that is a sub-unit of a module	
Pod		A Collection of racks, managed by a Pod Manager	
PCIe*		PCI Express*	
PSME		Intelligent Platform Management Bus	
Rack		A physical rack and a collection of trays managed by a RMM	
SPS		Intel Server Platform Services. Firmware within Intel ME engine that supports various server manageability features.	
String	string	113 byte (max) string (to allow for headers to make 128)	
Tray		A unit that slides directly in a Rack, it may contain modules. This term is synonymous with “Chassis” and “Drawer”.	
TPM		Trusted Platform Module	
Intel TXT		Intel® Trusted Execution Technology	

1.2 Reference Documents

Table 2. Reference Documents

Doc ID	Title	Location
336811	Intel® Rack Scale Design (RSD) Conformance and Software Reference Kit Getting Started Guide v2.2, Revision 001	http://www.intel.com/intelRSD
336814	Intel® Rack Scale Design Pod Manager (PDOM) Release Notes, Software v2.2, Revision 001	
336815	Intel® Rack Scale Design Pod Manager (PDOM) User Guide, Software v2.2, Revision 001	
336816	Intel® Rack Scale Design PSME Release Notes, Software v2.2, Revision 001	
336810	Intel® Rack Scale Design PSME User Guide, Software v2.2, Revision 001	
336855	Intel® Rack Scale Design PSME REST API Specification, Software v2.2, Revision 001	
336856	Intel® Rack Scale Design Storage Services API Specification, Software v2.2, Revision 001	
336857	Intel® Rack Scale Design Pod Manager REST API Specification, Software v2.2, Revision 001	
336858	Intel® Rack Scale Design Rack Management Module (RMM) API Specification, Software v2.2, Revision 001	
336859	Intel® Rack Scale Design Generic Assets Management Interface API Specification, Software v2.2, Revision 001	

Doc ID	Title	Location
336861	Intel® Rack Scale Design Architecture Specification, Software v2.2, Revision 001	
336862	Intel® RSD v2.2 Solid State Drive (SSD) Technical Advisory	
RFC2119	Key words for use in RFCs to Indicate Requirement Levels, March 1997	https://www.ietf.org/rfc/rfc2119.txt
SDP0266	Scalable Platforms Management API Specification v1.1.0	https://www.dmtf.org/sites/default/files/standards/documents/DSP0266_1.1.0.pdf
DSP8010	Redfish Schema v2016.3	https://www.dmtf.org/sites/default/files/standards/documents/DSP8010_2016.3.zip
332200	Intel® Intelligent Power Node Manager 4.0 External Interface Specification Using IPMI	http://www.intel.com/content/dam/www/public/us/en/documents/technical-specifications/intel-power-node-manager-v3-spec.pdf
332936	Intel® Rack Scale Design BIOS & BMC Technical Guide	https://www.intel.com/content/dam/www/public/us/en/documents/guides/bios-bmc-technical-guide.pdf
N/A	Intel® Intelligent Platform Management Interface 2.0 Specification	https://www.intel.com/content/www/us/en/servers/ipmi/ipmi-second-gen-interface-spec-v2-rev1-1.html
N/A	Intel® 64 and IA-32 Architectures Software Developer Manuals	https://software.intel.com/en-us/articles/intel-sdm
N/A	System Management BIOS (SMBIOS) Reference Specification	https://www.dmtf.org/standards/smbios
N/A	PCI Express External Cabling Specification, Revision 3.0	https://pcisig.com/specifications

§

2.0 Intel® Rack Scale Design IPMI Commands

This section describes the various Intelligent Platform Management Interface (IPMI) commands that are required to be implemented in various FW entities (UEFI FW, ME/SPS, and BMC) to support Intel® RSD 2.2

2.1 ME/SPS IPMI Commands

If the platform supports Intel® Manageability Engine (ME)/ Intel Server Platform Services (SPS), Intel® RSD 2.2 requires ME/SPS to implement Intel® Intelligent Power Node Manager functionality and expose telemetry data available through Node Manager.

The Baseboard Management Controller (BMC) is required to implement support for Intelligent Platform Management Bus (IPMB) proxy functionality to allow the Intelligent Platform Management Bus (PSME) to communicate with ME/SPS. Intel® Rack Scale Design PSME will send IPMI commands to ME/SPS using this Intelligent Platform Management Bus (IPMB) proxy capability. Refer to Intel® Intelligent Power Node Manager External Interface Specification for details on the ME/SPS supported Intelligent Platform Management Interface (IPMI) commands.

2.2 Rack Scale BMC IPMI Extension Commands

IPMI Rack Scale Extensions Commands, Refer to Table 3:

Table 4 defines the requests that the BMC accepts and the corresponding functionality and request/response data for these commands. These commands may be sent to the BMC from the IPMB, Low Pin Count (LPC)/Keyboard Controller Style (KCS), or Management LAN interfaces (e.g. NC-SI over dedicated or shared Ethernet interface). The source entities that will send these commands can be BIOS and PSME.

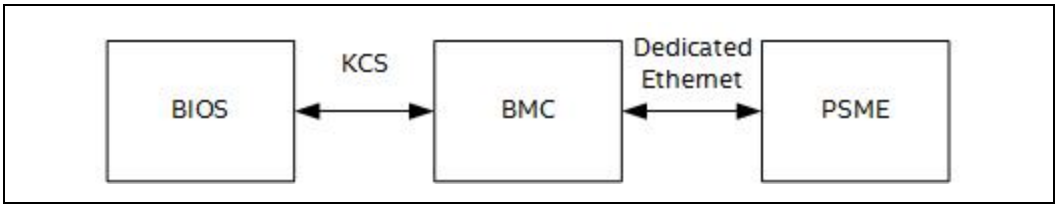
A Platform implementation must ensure that interfaces used by BIOS and PSME are mutually exclusive so the BMC can isolate and fail commands that are coming from the non-allowed command source. BMC must also isolate OS level SW interface into the BMC IPMI interface so the OS level software does not have access to Rack Scale Design specific IPMI commands. An example connection diagram is shown in Figure 1.

For example if a command is allowed only from PSME, then BMC must fail the same command if it is received from any other interface that PSME is not using. If the BMC exposes more than one management Ethernet ports, BMC must isolate Rack Scale Design specific IPMI commands to only one such management port where PSME is connected to. Specification of such dedicated ports externally visible for PSME to use can be through documentation for the specific system and BMC implementation.

Determination of the specific Ethernet port used for PSME communication is outside the scope of this document. The Description column in IPMI Rack Scale Extensions Commands:

Table 4 indicates the allowed source for each command.

Figure 1. Example Connection Diagram



Beyond the commands described in this section, the BMC will also support IPMB Bridging and expose ME/SPS IPMI commands to PSME. Refer to Table 2, *Intelligent Platform Management Interface 2.0 Specification* and *Intel® Intelligent Power Node Manager External Interface Specification* for details on IPMB Bridging support and the specific IPMI commands that are applicable from Intel® Rack Scale Design perspective.

The Intel® RSD FW Extension IPMI commands use the IPMI specification defined "Group Extension" Network Function Code (2Ch/2Dh). The specific defining body ID that is used is 04h (Intel® Rack Scale Design Specification). As such, the first data byte position in requests and second data byte in responses is to be set to 04h to identify Rack Scale Design FW Extensions. Refer to Table 2, *Intelligent Platform Management Interface 2.0 Specification,* and Section 5.1, Management Network Exposure to Host OS for details about the Group Extension Network Function code usage.

Note: It is possible that using Group Extension Network Function Code and an as yet undefined/unused defining body ID of 04h may cause conflicts with any other BMC IPMI implementation that is also using the same mechanism. Care must be taken to avoid such conflicts and it is recommended that the defining body ID of 04h be used for Intel® RSD purposes.

Note: If the BMC implements the PSME functionality directly, then some of the IPMI commands may become redundant and the BMC's PSME functionality can subsume those IPMI command functionality internally. In such cases, the BMC implementation may review the specific IPMI commands that are marked with PSME as the only source and are typically the ones that could be subsumed.

Note: Platform BMC may support both Intel® RSD 2.1 BMC IPMI commands (as described in the specification in Intel® Rack Scale Deign Bios and BMC Technical Guide, refer to Table 2.) and Intel® RSD v2.2 BMC IPMI commands as described in this document simultaneously if the platform needs to support Intel® RSD 2.1 and Intel® RSD v2.2 Software stack. Depending on the platform, the BIOS may also support either Intel® RSD 2.1 and Intel® RSD v2.2 IPMI commands or just Intel® RSD v2.2 IPMI commands and BMC may use the data obtained from BIOS through the Intel® RSD v2.2 interfaces and expose via the Intel® RSD 2.1 IPMI commands to the Intel® RSD 2.1 PSME Software.

Table 3 below summarizes the list of Intel® RSD FW Extension commands that are applicable for Intel® Xeon® Processor Scalable Family Platform. It also identifies dependencies within each firmware component and whether Intel provides UEFI FW Reference Code for each specific command.

Table 3. IPMI Rack Scale Design Extensions Command Summary

Command Code	Command	BIOS	BMC	ME/SPS	Intel® RSD 2.1 vs. Intel® RSD v2.2
1Fh	Get CPU PECI Package Config Data	Yes	Yes	Yes	New to Intel® RSD v2.2
20h	Get MDR Data Region Status	Yes	Yes	No	New Command Format and additional regions in Intel® RSD v2.2
21h	Get MDR Region Update Complete	Yes	Yes	No	New Command Format and additional regions in Intel® RSD v2.2
22h	MDR Region Read	Yes	Yes	No	New Command Format and additional regions in Intel® RSD v2.2
23h	MDR Region Write	Yes	Yes	No	New Command Format and additional regions in Intel® RSD v2.2
24h	Get MDR Region Lock	Yes	Yes	No	New Command Format and additional regions in Intel® RSD v2.2
25h	Get System Mode	Yes	Yes	No	New to Intel® RSD v2.2
26h	Set System Mode	No	Yes	No	New to Intel® RSD v2.2
29h	Get Trusted Platform Module (TPM) Configuration	Yes	Yes	No	New to Intel® RSD v2.2
30h	Set TPM Configuration	Yes	Yes	No	New to Intel® RSD v2.2
31h	TPM Configuration Update Complete	Yes	Yes	No	New to Intel® RSD v2.2
37h	Read PCIe* Cable EEPROM Data	Yes	Yes	No	New Command Format in Intel® RSD v2.2

For the base specification and descriptions of the BMC commands other than those specified in this chapter, Refer to Table 2, *Intelligent Platform Management Interface 2.0 Specification.*

All Get commands require Privileged User access.

All Set commands require Privileged Admin access.

2.2.1 IPMI Rack Scale Extensions Commands:

Table 4. IPMI Rack Scale Extensions Command Table

Net Function = Group Extension (2Ch/2Dh), LUN = 00			
Code	**Command**	**Request, Response Data**	**Description**
1Fh	Get CPU PECI Package Config Data	Request: byte 1 – 04h (Intel® RSD Specification) byte 2 – 0 based Node index byte 3 – 0-based CPU index byte 4 – PCS command index byte 5 – Reserved byte 6:9 – PCS command parameter Response: byte 1 – Completion code 00h = Normal C7h = Invalid length C9h = Invalid parameter D3h = Node/CPU not present byte 2 – 04h (Intel® RSD Specification) byte 3 – 0-based Node index byte 4 – 0-based CPU index byte 5 – Reserved byte 6:9 – PCS PECI Data	Command Source: PSME This command is used to access telemetry and other information available via the CPU PECI bus. BMC/ME must provide access only to Read capability and not write capability. Actual implementation could be either by the BMC accessing the PECI bus directly or via ME PECI-Proxy method depending on the hardware implementation. Refer to *Intel® Intelligent Power Node Manager 4p0* External Interface Specification for PECI-Proxy interface details. PSME may attempt to use PECI Proxy command first and if it's not supported or nonfunctional, then it may use this command to access PECI.
20h	Get MDR Data Region Status	Request: byte 1 – 04h (Intel® RSD Specification) byte 2 – Data Region 01h = SMBIOS region 02h = Reserved region 03h = Reserved region 04h = iSCSI Boot Options region All other values are reserved. Response: byte 1 – Completion code 00h = Normal C9h = Invalid region specified byte 2 – 04h (Intel® RSD Specification) byte 3 – MDR version byte 4 – Data region identifier 01h = SMBIOS table 02h = Reserved region 03h = Reserved region 04h = iSCSI Boot Options region All other values are reserved. byte 5 – Data validation 00h = Invalid data	Command Source: BIOS, PSME A multi-purpose Managed Data Region (MDR) in BMC is used to maintain certain data that can be stored and accessed by other components in Intel® RSD. This command is utilized by either the BIOS, PSME or an external application to retrieve the status of the specified data region to see if it has valid data and, if it does, whether the length and checksum of the data matches the local copy of the data. If determined that the data needs to be refreshed, use the following read / write / lock commands to perform the update.

Net Function = Group Extension (2Ch/2Dh), LUN = 00			
Code	**Command**	**Request, Response Data**	**Description**
		01h = Valid data All other values are reserved. byte 6 – Unique data contents update count byte 7 – Lock Status 00h = Unlocked 01h = Strict lock 02h = Pre-emptible Lock All other values are reserved. Byte 8:9 – Maximum size of region in bytes byte 10:11 – Used size of region in bytes byte 12 – Region checksum	
21h	MDR Region Update Complete	Request: byte 1 – 04h (Intel® RSD Specification) byte 2 – Session Lock Handle byte 3 – Data Region. 01h = SMBIOS region 02h = Reserved region 03h = Reserved region 04h = iSCSI Boot Options region All other values are reserved. Response: byte 1 – Completion code 00h = Normal 81h = Region is in use by another user. C9h = Invalid Region Specified. D5h = Region is not locked. byte 2 – 04h (Intel® RSD Specification)	Command Source: BIOS, PSME This command is utilized by the BIOS and PSME to indicate that its update of the specified data region is complete. Session Lock Handle is returned by the "Get MDR Region Lock" command. **NOTE**: Call to MDR Region Update Complete after all the writes have happened will result in the region having valid data and the lock is released. If MDR Region Update Complete is not called, then the data region remains invalid even if writes have happened.
22h	MDR Region Read	Request: byte 1 – 04h (Intel® RSD Specification) byte 2 – Data Region. 01h = SMBIOS region 02h = Reserved region 03h = Reserved region 04h = iSCSI Boot Options region All other values are reserved. Byte 3 – Data Length to Read. Byte 4:5 – Offset to read. Response: byte 1 – Completion code 00h = Normal 81h = Region is in use by another user or region data is invalid.	Command Source: PSME, BIOS This command is utilized by BIOS or external applications to retrieve the specified data region. There is no need to acquire a lock using Get MDR Region Lock prior to issuing MDR Region Read command. If a Get MDR Region Lock happens while a read is happening, then a subsequent MDR Region Read will return a response of 81h.

Net Function = Group Extension (2Ch/2Dh), LUN = 00			
Code	**Command**	**Request, Response Data**	**Description**
		C4h = Request is larger than maximum allowed payload size. C7h = Requested data extends beyond length of region C9h = Invalid Region Specified. byte 2 – 04h (Intel® RSD Specification) byte 3 – Read Length. byte 4 – Update Count byte 5:N – Data	
23h	MDR Region Write	Request: byte 1 – 04h (Intel® RSD Specification) byte 2 – Session Lock Handle byte 3 – Data Region. 01h = SMBIOS region 02h = Reserved region 03h = Reserved region 04h = iSCSI Boot Options region All other values are reserved. byte 4 – Data Length byte 5:6 – Data Offset. Byte 7:N – Data to be written. Response: byte 1 – Completion code 00h = Normal 81h = Region is in use by another user. C7h = Requested data extends beyond length of region C9h = Invalid Region Specified. D5h = Data region not locked. byte 2 – 04h (Intel® RSD Specification) byte 3 – Data Region. 01h = SMBIOS region 02h = Reserved region 03h = Reserved region 04h = iSCSI Boot Options region All other values are reserved. byte 4 – Valid Data 00h = Invalid 01h = Valid All other values are reserved. byte 5 – Lock Status 00h = Unlocked 01h = Strict Lock 02h = Pre-emptible Lock All other values are reserved. byte 6:7 – Max Region Length	Command Source: BIOS, PSME This command is utilized by BIOS or PSME to write the specified data region. BIOS is responsible for writing SMBIOS region. PSME is responsible for writing iSCSI Boot Options region. Section 4.0 for detailed structure for iSCSI Boot Options region. For each of the supported data region, BIOS or PSME (as appropriate) builds a memory buffer with the specific data for that region and passes it to BMC via this command in 255 byte chunks. The maximum capacity of each region cannot exceed 64Kbytes (as stipulated by the 2byte data offset field) or the maximum region size as identified by the “Get MDR Data Region Status” IPMI command, whichever is lower. If the MDR Data region size is determined to be less than the size required for the actual data, then the BIOS or PSME will write no data for that region. A valid checksum, but empty region, will indicate the data exceeded size limits. However, a platform design will make all efforts to accommodate the size requirements for this data for the worst case configuration possible for the platform. Session Lock Handle is returned by the “Get MDR Region Lock” command.

Net Function = Group Extension (2Ch/2Dh), LUN = 00			
Code	**Command**	**Request, Response Data**	**Description**
		byte 8:9 – Size of region used (in bytes)	
24h	Get MDR Region Lock	Request: byte 1 – 04h (Intel® RSD Specification) byte 2 – Session Lock Handle 00h = Request lock/session handle Any other value is an existing session handle to be extended or aborted/invalidated. byte 3 – Data Region 01h = SMBIOS region 02h = Reserved region 03h = Reserved region 04h = iSCSI Boot Options region All other values are reserved. byte 4 – Lock Type 00h = Abort; unlock without completing operation. 01h = Strict Lock; only writes by user who locked the region are allowed. 02h = Pre-emptible lock; strict locks override this type of lock. All other values are reserved. byte 5:6 – Timeout. Number of milliseconds allowed before lock is released. Response: byte 1 – Completion code 00h = Normal 81h = Region is in use by another user or session handle is invalid. C9h = Invalid Region Specified. D5h = Data region locked. byte 2 – 04h (Intel® RSD Specification) byte 3 – Session Lock Handle; only valid if byte 1 is a successful code (00h).	Command Source: BIOS, PSME This command locks the specified data region. Strict or Pre-emptible lock can be used when an MDR region must be written. If Get MDR Data Region Status indicates it's locked (any type), it is recommended that the caller waits for the lock to be released before reading or writing the data, unless absolutely necessary to update the data immediately. **NOTE**: Call to Get MDR Region Lock to acquire lock for a specified region and a subsequent call to Get MDR Region Lock to Abort the lock or no subsequent call to MDR Region Update Complete renders the data in the region invalid. Get MDR Data Region Status in that case returns Invalid Data for the selected region.
25h	Get System Mode	Request: byte 1 – 04h (Intel® RSD Specification) Response: byte 1 – Completion code 00h = Normal byte 2 – 04h (Intel® RSD Specification) byte 3 – Admin Mode	Command Source: BIOS, PSME

Net Function = Group Extension (2Ch/2Dh), LUN = 00			
Code	**Command**	**Request, Response Data**	**Description**
		00h = Admin Mode (Default) 01h = User Mode byte 4 – Rack Scale Design Mode 00h = Rack Scale Mode Off (Default) 01h = Rack Scale Mode On	Command that can be used by the trusted component that is performing the actual FW update (BIOS SMM or other SW agents on the system) to determine if the current boot of the node is in admin mode and hence firmware updates and other system configuration actions can be allowed. The actual implementation may use one trusted component (BIOS SMM for example) to lock the SPI based on the selected mode and keep unlocked in other cases. It is required that the in-band BMC/ME/SPS/NIC/3D Xpoint and BIOS FW updates for the composed node are allowed only in Admin mode. Prevent Updates only when (Rack Scale Mode = On and User Mode = Yes) Allow updates in any other combination. A call to Get System Mode immediately after Set System Mode will return the configuration set with the Set System Mode command. In other words, PSME must not rely on the Get System Mode IPMI command response to confirm if FW updates will be prevented.
26h	Set System Mode	Request: byte 1 – 04h (Intel® RSD Specification) byte 2 – Admin Mode 00h = Admin Mode 01h = User Mode byte 3 – Rack Scale Design Mode 00h = Rack Scale Mode Off 01h = Rack Scale Mode On All other values are reserved. Response: byte 1 – Completion code 00h = Normal D4h = Insufficient privilege level byte 2 – 04h (Intel® RSD Specification) byte 3 – Admin Mode 00h = Admin Mode 01h = User Mode byte 4 – Rack Scale Design Mode	Command Source: PSME Command used by Rack Scale Design PSME to indicate that the node must boot in admin mode for the next boot. PSME must ensure there's a reboot initiated for this Set System Mode IPMI command changes are activated by the BIOS. This command must be allowed only within an authenticated IPMI session.

Net Function = Group Extension (2Ch/2Dh), LUN = 00			
Code	**Command**	**Request, Response Data**	**Description**
		00h = Rack Scale Mode Off 01h = Rack Scale Mode On	
29h	Get TPM Configuration	Request: byte 1 – 04h (Intel® RSD Specification) Response: byte 1 – Completion code 00h = Normal / Command Successful byte 2 – 04h (Intel® RSD Specification) byte 3 – TPM Command Status 00h = Command Not Valid. No action needed by BIOS (Default) 01h = Command Valid. BIOS must act on this byte 4 – TPM Version Selection 00h = Disable TPM (Default) 01h and above = Desired TPM version value as identified in the "TPM Configuration Index" field from the SMBIOS TPM OEM Record data (Refer to Section 3.6) byte 5 – Clear TPM Action 00h = Preserve User TPM Ownership (Default) 01h = Clear User TPM Ownership All other values are reserved.	Command Source: BIOS, PSME This command is used to get desired TPM configuration on the platform. The BIOS will issue this IPMI command to get the new TPM configuration information that PSME has setup for BIOS to use. If the response indicates any TPM action should be taken, the BIOS performs the necessary changes. The BIOS will issue "TPM Configuration Update Complete" IPMI Command after it makes the necessary changes and then reboot the platform.
30h	Set TPM Configuration	Request: byte 1 – 04h (Intel® RSD Specification) byte 2 – TPM Command Status 00h = Command Not Valid or BIOS completed setting up TPM 01h = Command Valid. BIOS must act on this data byte 3 – TPM Version Selection 00h = Disable TPM 01h and above = Desired TPM version value as identified in the "TPM Configuration Index" field from the SMBIOS TPM OEM Record data (Refer to Section 3.6). byte 4 – Clear TPM Action 00h = Preserve User TPM Ownership 01h = Clear User TPM Ownership All other values are reserved.	Command Source: PSME This command is used to clear the user TPM information on the platform. The PSME issues this command to instruct the BIOS to change the TPM configuration information upon the next boot. When the PSME issues the command, it sets byte 3 to 01h to let the BIOS know it has to act on the data during next boot. The BMC saves this information and provides it via the Get TPM configuration command.

Net Function = Group Extension (2Ch/2Dh), LUN = 00			
Code	**Command**	**Request, Response Data**	**Description**
		Response: byte 1 – Completion code 00h = Normal / Command Successful D4h = Insufficient privilege level byte 2 – 04h (Intel® RSD Specification) byte 3 – TPM Command Status 00h = Command Not Valid. No action needed by BIOS or BIOS completed requested action and byte 3 reflects current configuration (Default) 01h = Command Valid. BIOS must act on this byte 4 – TPM Version Selection 00h = Disable TPM (Default) 01h and above = Desired TPM version value as identified in the "TPM Configuration Index" field from the SMBIOS TPM OEM Record data (Refer to Section 3.6). byte 5 – Clear TPM Action 00h = Preserve User TPM Ownership (Default) 01h = Clear User TPM Ownership	It is possible that a Disable TPM command may not be successful due to other dependencies (for example, the platform has Intel Trusted Execution Technology (Intel TXT) enabled which requires the TPM to be enabled). In such cases, the BIOS is expected to issue this command with the current configuration identified appropriately in byte 3.
31h	TPM Configuration Update Complete	Request: byte 1 – 04h (Intel® RSD Specification) byte 2 – TPM Command Status 00h = BIOS completed setting up TPM byte 3 – TPM Version Selection – current configuration selected/enabled by BIOS. 00h = Disable TPM 01h and above = Desired TPM version value as identified in the "TPM Configuration Index" field from the SMBIOS TPM OEM Record data (Refer to Section 3.6). byte 4 – Clear TPM Action performed 00h = Preserve User TPM Ownership 01h = Clear User TPM Ownership All other values are reserved. Response: byte 1 – Completion code 00h = Normal / Command Successful	Command Source: BIOS This command is used by BIOS to indicate the TPM configuration has been updated successfully. In response to this command, the BMC save bytes 2 – 4 and returns the same via the Get TPM Configuration command in the corresponding bytes. The BIOS must not issue this command unless the "Get TPM Configuration" returns a response with byte 3 as 01h which indicates the BIOS must act on the command.

Net Function = Group Extension (2Ch/2Dh), LUN = 00			
Code	**Command**	**Request, Response Data**	**Description**
		D4h = Insufficient privilege level byte 2 – 04h (Intel® RSD Specification)	
37h	Read PCIe Cable EEPROM Data	Request: byte 1 – 04h (Intel® RSD Specification) byte 2-3 – PCIe Slot ID where the Cable is connected Byte 4 – Cable Port Index within the PCIe Slot where the cable is connected. Byte 5 – Upper Memory Page Select. Must be 0. This allows for future additional pages that could get defined for the upper memory region (bytes 128 to 255) of the Cable Memory. byte 6 – starting offset of the data to be read (0 to 255) byte 7 – length of data to be read in bytes Response: byte 1 – Completion code 00h = Normal / Command Successful 80h = PCIe Slot ID specified is invalid 81h = Cable port index specified is invalid 82h = No Cable present for the specified Cable Port Index C2h = No Cabled PCIe ports available in the platform C7h = Invalid Length D4h = Insufficient privilege level byte 2 – 04h (Intel® RSD Specification) byte 3 – length of data read byte 4 – N – Cable memory data	Command Source: PSME This command is used by PSME to fetch specific EEPROM data from the PCIe cable (conforming to SFF-8644 connector standard) that is connected to a specific Cable Port. The PCIe Slot ID specified is the same Slot ID value specified in the SMBIOS table type 9 structure as well as the Cabled PCIe Port Information SMBIOS data structure as described Section 3.10.

§

3.0 SMBIOS Data Structures

This section defines the data structures used for various platform inventory data in SMBIOS for use by Intel® RSD v2.2 above and beyond what is available as part of the SMBIOS specification. It is expected that BIOS will update the SMBIOS table and use the MDR related IPMI commands described in the previous section to pass the updated table to the BMC for use by PSME.

Intel® RSD v2.2 defines type 190 to 209 for Rack Scale Design SMBIOS structure definitions. Care must be taken by the OEM BIOS to avoid using the type 190 to 209 for other OEM-specific information in the SMBIOS tables.

Where appropriate, this section refers to the SMBIOS specification structures that are used by the Rack Scale Design and is fully supported on a Rack Scale Design compliant system.

3.1 NIC Information

SMBIOS OEM type 190 is used to declare details about available/configured NIC on the platform.

Table 5. NIC Information SMBIOS Structure

Offset	Name	Length	Value	Description
00h	Type	BYTE	190	NIC Information Identifier
01h	Length	BYTE	Varies	-
02h	Handle	WORD	Varies	-
04h	PCI Class Code	1 BYTE	Varies	PCI Class Code is used to identify the generic function of the device
05h	Slot Number	WORD	Varies	Physical Slot Number of the Slot Connect to the PCIe* port
07h	Vendor ID	WORD	Varies	Vendor Identification Number
09h	Device ID	WORD	Varies	Device Identification Number
0Bh	SubVendor ID	WORD	Varies	Sub system ID and Sub Vendor ID Differentiate specific model
0Dh	SubDevice ID	WORD	Varies	Sub system ID and Sub Vendor ID Differentiate specific model
0Fh	Maximum Speed	DWORD	Varies	Maximum speed in Mbps (e.g. 10000 for 10 Gbps; 40000 for 40 Gbps)
13h	Current Speed	DWORD	Varies	Current speed in Mbps (e.g. 10000 for 10 Gbps; 40000 for 40 Gbps)
17h	Port Index	WORD	Varies	Port Index in the system. This value matches the value exposed by the BIOS in the IPMI specification's Device Instance selector field definition for Get/Set System Boot Options IPMI command (Boot Options Parameter structure, Boot Flags parameter 5 data 5).
19h	MAC Address	32 BYTES	Varies	Network MAC address
39h	Firmware Version	BYTE	STRING	String number of the string that identifies the version of the firmware installed on the NIC.

3.2 PCIe* Information

SMBIOS OEM type 192 is used to declare details about the available and enabled PCI Express* (PCIe*) devices on the platform.

Table 6. PCI Express* (PCIe*) Information SMBIOS Structure

Offset	Name	Length	Value	Description
00h	Type	BYTE	192	PCIe
01h	Length	BYTE	Varies	Size of the Table
02h	Handle	WORD	Varies	A unique 16-bit number of structure's handle
04h	PCI Class Code	BYTE	Varies	PCI Class Code is used to identify the generic function of the device
05h	Slot Number	WORD	Varies	The physical Slot Number of the Slot Connect to the PCIe port. The slot number shall match the Slot ID field within the SMBIOS System Slots (Type 9) attribute structure for the slot this PCIe device is located in.
07h	Vendor ID	WORD	Varies	Vendor Identification Number
09h	Device ID	WORD	Varies	Device Identification Number
0Bh	SubVendor ID	WORD	Varies	Sub system ID and Sub Vendor ID Differentiate specific model
0Dh	SubDevice ID	WORD	Varies	Sub system ID and Sub Vendor ID Differentiate specific model
0Fh	Link Speed	DWORD	Varies	Current Link Speed
13h	Link Width	DWORD	Varies	Current Link Width (1, 2, 4, 8, 16 etc.)

3.3 Processor CPUID Information

SMBIOS OEM type 193 is used to declare CPUID leaf information of the processors on the system. The structure is as defined in Table 7 and Table 8. Fields EAX_00h etc. below from offset 05h onwards uses a structure as shown in Table 9 to pack the four DWORDS returned by the CPUID instruction. Refer to *Intel® 64 and IA-32 Architectures Software Developer Manual* for details of the actual data returned by the CPUID instruction.

Note: The BIOS shall construct the CPUID SMBIOS table as late as possible on every boot to allow for any dynamic changes per configuration of the system to reflect in the CPUID.

Table 7. Processor CPUID SMBIOS Structure Subtype 1

Offset	Name	Length	Value	Description
00h	Type	BYTE	193	Processor CPUID Information indicator
01h	Length	BYTE	Varies	-
02h	Handle	WORD	Varies	-
04h	Socket Designation	BYTE	STRING	The string number of the string that identifies the physically labelled socket or board position where the processor is located. EXAMPLE: "Socket 0"

Offset	Name	Length	Value	Description
05h	SubType	BYTE	01h	SubType associated with processor CPUID SMBIOS Structure. This subtype holds CPUID data for EAX values 00h to 10h
06h	EAX_00h	16 BYTES	Varies	CPUID data when EAX = 00h.
16h	EAX_01h	16 BYTES	Varies	CPUID data when EAX = 01h.
26h	EAX_02h	16 BYTES	Varies	CPUID data when EAX = 02h.
36h	EAX_03h	16 BYTES	Varies	CPUID data when EAX = 03h.
46h	EAX_04h	16 BYTES	Varies	CPUID data when EAX = 04h.
56h	EAX_05h	16 BYTES	Varies	CPUID data when EAX = 05h.
66h	EAX_06h	16 BYTES	Varies	CPUID data when EAX = 06h.
76h	EAX_07h	16 BYTES	Varies	CPUID data when EAX = 07h.
86h	EAX_09h	16 BYTES	Varies	CPUID data when EAX = 09h.
96h	EAX_0Ah	16 BYTES	Varies	CPUID data when EAX = 0Ah.
A6h	EAX_0Bh	16 BYTES	Varies	CPUID data when EAX = 0Bh.
B6h	EAX_0Dh	16 BYTES	Varies	CPUID data when EAX = 0Dh, ECX = 00h.
C6h	EAX_0Fh	16 BYTES	Varies	CPUID data when EAX = 0Fh, ECX = 00h.
D6h	EAX_10h	16 BYTES	Varies	CPUID data when EAX = 10h, ECX = 00h

Table 8. Processor CPUID SMBIOS Structure Subtype 2

Offset	Name	Length	Value	Description
00h	Type	BYTE	193	Processor CPUID Information indicator
01h	Length	BYTE	Varies	-
02h	Handle	WORD	Varies	-
04h	Socket Designation	BYTE	STRING	String number of the string that identifies the physically labelled socket or board position where the processor is located. EXAMPLE: “Socket 0”
05h	SubType	BYTE	02h	SubType associated with processor CPUID SMBIOS Structure. This subtype holds CPUID data for EAX values 14h to 80000008h.
06h	EAX_14h	16 BYTES	Varies	CPUID data when EAX = 14h, ECX = 00h.
16h	EAX_15h	16 BYTES	Varies	CPUID data when EAX = 15h.
26h	EAX_16h	16 BYTES	Varies	CPUID data when EAX = 16h.
36h	EAX_17_00h	16 BYTES	Varies	CPUID data when EAX = 17h, ECX = 00h.
46h	EAX_17_01h	16 BYTES	Varies	CPUID data when EAX = 17h, ECX = 01h.
56h	EAX_17_02h	16 BYTES	Varies	CPUID data when EAX = 17h, ECX = 02h.
66h	EAX_17_03h	16 BYTES	Varies	CPUID data when EAX = 17h, ECX = 03h.
76h	EAX_80000000h	16 BYTES	Varies	CPUID data when EAX = 80000000h.
86h	EAX_80000001h	16 BYTES	Varies	CPUID data when EAX = 80000001h.
96h	EAX_80000002h	16 BYTES	Varies	CPUID data when EAX = 80000002h.
A6h	EAX_80000003h	16 BYTES	Varies	CPUID data when EAX = 80000003h.
B6h	EAX_80000004h	16 BYTES	Varies	CPUID data when EAX = 80000004h.
C6h	EAX_80000006h	16 BYTES	Varies	CPUID data when EAX = 80000006h.
D6h	EAX_80000007h	16 BYTES	Varies	CPUID data when EAX = 80000007h.

Offset	Name	Length	Value	Description
E6h	EAX_80000008h	16 BYTES	Varies	CPUID data when EAX = 80000008h.

Table 9. CPUID Data Byte Order

Offset	Name	Length	Value	Description
00h	EAX	DWORD	Varies	EAX register value as returned by CPUID instruction.
04h	EBX	DWORD	Varies	EBX register value as returned by CPUID instruction.
08h	ECX	DWORD	Varies	ECX register value as returned by CPUID instruction.
0Ch	EDX	DWORD	Varies	EDX register value as returned by CPUID instruction.

3.4 Memory Device Information

Intel® RSD 2.x relies on the SMBIOS specification defined Memory related data structures (Types 17, 18, 19 and 20). The BIOS implements support for these memory related SMBIOS data structures in a Rack Scale Design compliant system. Refer to *System Management BIOS (SMBIOS) Reference Specification* for more details on these memory data structures.

3.5 Storage Device informationg36

SMBIOS OEM type 194 is used to declare Storage Device information on the system. The structure is as defined in the Table 10.

Table 10. Storage Device SMBIOS Structure

Offset	Name	Length	Value	Description
00h	Type	BYTE	194	Storage Device Information indicator
01h	Length	BYTE	Varies	-
02h	Handle	WORD	Varies	-
04h	Port Designation	BYTE	STRING	String number of the string that identifies the physically labelled port or board position where the storage device is located. EXAMPLE: “SATA Port 0” or “PCIe Port 1”
05h	Device Index	BYTE	Varies	Index of this device in the IPMI Get/Set Boot Options command as provided by BIOS. This is used by PSME to identify the boot device index for use in Set Boot Options IPMI command when a change of boot device is required.
06h	Connector Type	BYTE	Varies	Connector Type indicator. 00h = Unknown 01h = SATA 02h = SAS 03h = PCIe 04h = m.2 05h = USB

Offset	Name	Length	Value	Description
				06h = U.2
07h	Device Protocol	BYTE	Varies	Device Protocol indicator 00h = Unknown 01h = IDE 02h = AHCI 03h = NVMe 04h = USB
08h	Device Type	BYTE	Varies	Device Type indicator. 01h = HDD 02h = SSD 03h = Optical – DVD 04h = Optical – Blue Ray 05h = USB
09h	Device Capacity	DWORD	Varies	Device Capacity in GB
0Dh	Device RPM	WORD	Varies	Device RPM value (applicable for HDDs. 0 for Non HDD)
0Fh	Device Model	BYTE	STRING	String number of the string that identifies the device's model string.
10h	Device Serial	BYTE	STRING	String number of the string that identifies the device's serial number.
11h	PCI Class Code	BYTE	Varies	PCI Class code in case of PCIe storage device, 0 otherwise
12h	Vendor ID	WORD	Varies	PCI Vendor ID in case of PCIe storage device, 0 otherwise
14h	Device ID	WORD	Varies	PCI Device ID in case of PCIe storage device, 0 otherwise
16h	Sub Vendor ID	WORD	Varies	PCI Sub Vendor ID in case of PCIe storage device, 0 otherwise
18h	Sub Device ID	WORD	Varies	PCI Sub Device ID in case of PCIe storage device, 0 otherwise
1Ah	Firmware Version	BYTE	STRING	String number of the string that identifies the device's firmware version.

3.6 Trusted Platform Module Information

SMBIOS OEM type 195 is used to declare TPM information on the system. The structure is as defined in Table 11. UEFI FW shall implement this structure only if the TPM is present on the platform, even if the TPM is currently disabled on the platform. Multiple structures can be exposed by UEFI FW if multiple TPM versions are possible on the platform. If no TPM is present, then this structure shall not be implemented in the SMBIOS.

Table 11. TPM Information SMBIOS Structure

Offset	Name	Length	Value	Description
00h	Type	BYTE	195	TPM Information indicator
01h	Length	BYTE	Varies	-
02h	Handle	WORD	Varies	-

Offset	Name	Length	Value	Description
04h	TPM Configuration Index	BYTE	Varies	1 based index to be used by PSME to select the desired configuration in the Set TPM Configuration IPMI command.
05h	TPM Version	BYTE	STRING	String number of the string that identifies the version of the TPM present on the system. Example: “TPM 2.0”
06h	TPM Status	BYTE	Varies	Current Status of this specific TPM in the platform. 00h = Disabled 01h = Enabled

3.7 Intel® Trusted Execution Technology Information

SMBIOS OEM type 196 is used to declare Intel TXT information on the system. The structure is as defined in Table 12. UEFI FW shall implement this structure only if Intel TXT capability is present on the platform, even if Intel XT is currently disabled on the platform.

Table 12. TXT Information SMBIOS Structure

Offset	Name	Length	Value	Description
00h	Type	BYTE	196	Intel TXT Information indicator
01h	Length	BYTE	Varies	-
02h	Handle	WORD	Varies	-
04h	TXT Status	BYTE	Varies	Current Status of Intel TXT in the platform. 00h = Disabled 01h = Enabled

3.8 Memory Device Extended Information

SMBIOS OEM type 197 is used to declare extended information of the memory devices present on the system. Each of memory device extended information OEM record structure must match the SMBIOS spec defined, Type 17 memory device structure.

Table 13. Memory Device Extended Information SMBIOS Structure

Offset	Name	Length	Value	Description
00h	Type	BYTE	197	Memory Device Extended Information indicator
01h	Length	BYTE	Varies	-
02h	Handle	WORD	Varies	-
04h	Memory Device Handle	WORD	Varies	Handle, or instance number of the Memory Device SMBIOS OEM Record (Type 17) that this extended information represents.
06h	Memory Type	BYTE	Varies	Type of Memory 00h = DIMM 01h = NVDIMM_N (Byte Accessible Persistent Memory)

Offset	Name	Length	Value	Description
				02h = NVDIMM_F (Block Accessible Persistent Memory) 03h = NVDIMM_P
07h	Memory Media	BYTE	Varies	Type of Memory Media 00h = DRAM 01h = NAND 03h = Proprietary
08h	Memory Firmware Revision	BYTE	STRING	String number of the string that identifies the revision information of the firmware in the memory device.
09h	Memory Firmware API Version	BYTE	STRING	String number of the string that identifies the API version information that the firmware exposes.
0Ah	Max TDP	DWORD	Varies	Maximum Thermal Design Power (TDP) in mWs that this memory device will consume.
0Eh	Memory SMBus Address	BYTE	Varies	The SMBus address that is associated with this specific memory device. The specific SMBus address that must be reported must allow access to the SPD and/or 3DXPoint Firmware Interface. PSME will use this information to pass to Intel® Intelligent Power Node Manager as necessary to gather information from the Memory subsystem.

3.9 FPGA Information

SMBIOS OEM type 198 is used to declare FPGA information on the system. The structure is as defined in Table 14. UEFI FW shall implement this structure only if FPGA capability is present on the platform.

Table 14. FPGA Information SMBIOS Structure

Offset	Name	Length	Value	Description
00h	Type	BYTE	198	FPGA Information indicator
01h	Length	BYTE	Varies	-
02h	Handle	WORD	Varies	-
04h	FPGA Index	BYTE	Varies	1 based index to identify this specific FPGA instance on the platform
05h	FPGA Type	BYTE	Varies	Indicates the type of this FPGA: 00h = Integrated 01h = Discrete 02h = Discrete with SoC / HPS (Hard Processor Subsystem)
06h	FPGA Status	BYTE	Varies	Current Status of this FPGA. 00h = Disabled 01h = Enabled

Offset	Name	Length	Value	Description
07h	Socket Identifier	BYTE	Varies	Identifies the specific socket the FPGA is integrated into. Applicable only for Integrated FPGA type. Otherwise, this field is not applicable.
08h	FPGA Vendor	BYTE	STRING	String number of the string that identifies the FPGA vendor (either a descriptive string or Vendor ID identified in the PCI configuration space for the discrete FPGA)
09h	FPGA Family	BYTE	STRING	String number of the string that identifies the FPGA family.
0Ah	FPGA Model	BYTE	STRING	String number of the string that identifies the FPGA model.
0Bh	FPGA Bit Stream Version	BYTE	STRING	String number of the string that identifies the Bit Stream version.
0Ch	FPGA HPS Core Count	BYTE	Varies	Identifies the number of cores supported in the HPS Cores if available.
0Eh	FPGA HPS ISA	BYTE	Varies	Identifies the instruction set architecture for the HPS cores on this FPGA. 00h - x86 01h - x86-64 02h - IA-64 03h - ARM-A32 04h - ARM-A64 05h - MIPS32 06h - MIPS64 07h - OEM
0Fh	FPGA HSSI Configuration	BYTE	Varies	Identifies the supported HSSI configuration in this FPGA. 00h = Networking 01h = PCIe FFh = Information not available. Other HSSI related fields must be ignored.
10h	FPGA HSSI Port Count	BYTE	Varies	Number of Ports supported for the available HSSI configuration E.g. 2 for “2X10G”, 4 for “4x10G”, 2 for “2X40G”, 2 for “2 x16 PCIe” etc.
11h	FPGA HSSI Port Speed	BYTE	Varies	Speed in Gbps or number of Lanes supported by the HSSI configuration E.g. 10 for “2x10G”, 40 for “2x40G”, 16 for “x16 PCIe” etc.
12h	FPGA HSSI Side Band Configuration	BYTE	STRING	String number of the string that identifies the HSSI side band configuration mechanism supported by this FPGA. E.g.: “SPI”, “I2C-0”, “SMBus” etc.
13h	FPGA Reconfiguration Slots	BYTE	Varies	Number of Partial Reconfiguration (PR) Slots available in this FPGA
14h	PCIe Slot Number	WORD	Varies	Physical Slot Number of the Slot connected to the PCIE port where this FPGA can be accessed.
16h	PCIe Bus Identifier	BYTE	Varies	PCIe bus number where this FPGA can be accessed

Offset	Name	Length	Value	Description
17h	PCIe Device Identifier	BYTE	Varies	PCIe device number where this FPGA can be accessed
18h	PCIe Function Identifier	BYTE	Varies	PCIe function number where this FPGA can be accessed
19h	Thermal Design Power	DWORD	Varies	Thermal Design Power of the FPGA device in mW
1Dh	On Package Memory Technology	BYTE	Varies	Identifies the technology of the attached memory if any. 00h = None 01h = EDRAM 02h = HBM 03h = HBM2
1Eh	On Package Memory Capacity	DWORD	Varies	Capacity of On Package Memory in MB
22h	On Package Memory Speed	WORD	Varies	Speed of Memory in MHz

3.10 Cabled PCIe* Port Information

SMBIOS OEM type 199 is used to declare the Fixed Side Cabled PCIe Port information on the system. The structure is defined in Table 15. UEFI FW shall implement this structure only if the Cabled PCIe ports are present on the platform. Typically, Cabled PCIe ports are implemented in a system using the PCIe Cable Adapters that are either present onboard or attached to an existing PCIe slot via an adapter card (often with a re-timer). This structure describes each Cable Management Interface (CMI) controller within a Cabled PCIe port.

Platform BMC and BIOS must use the same value for the Slot ID for a given physical slot represented in the structure below. Determination and communication between Platform BMC and BIOS of which specific PCIe slot in the system has Cabled PCIe ports and determination of link widths, specific Slot ID etc. are out of the scope of this specification.

Intel® RSD requires that cabled PCIe ports are designed following the *PCI Express External Cabling Specification*, Revision 3.0 and the CMI as specified in the specification. Intel Rack Scale Design also requires systems implementing cabled PCIe ports to support SMBus accessibility to the slots by the system BMC to allow for cable identification and management.

Table 15. Cabled PCIe* Port Information SMBIOS Structure

Offset	Name	Length	Value	Description
00h	Type	BYTE	199	Cabled PCIe Port Information indicator
01h	Length	BYTE	Varies	
02h	Handle	WORD	Varies	
04h	Slot ID	WORD	Varies	PCIe Slot ID to which this specific Cabled PCIe Slot is connected. This information is used to identify and address specific Cabled PCIe slot / CMI controller by the BMC. This matches the slot ID field defined in the System Slots SMBIOS table (Type 9) as defined by the SMBIOS specification.

Offset	Name	Length	Value	Description
				It is required this field also matches with the slot ID that the BMC uses for the same slot. This field will be passed into the Get Cable EEPROM Data IPMI command to identify the specific cable port to fetch data.
06h	Cable Port Link Width	BYTE	Varies	Indicates the link width of this specific Cable Port. Typically 4 to indicate x4 cable port, 8 to indicate a x8 cable port etc.
07h	Cable Index Count	BYTE	Varies	Number of cable indices and corresponding PCIe lane ranges available within this Cable Port
08h	Cable Port Index 0	BYTE	Varies	0 based index that identifies a specific cable port index within the PCIe Slot identified by the Slot ID field above. It is required that field match the port index the BMC uses for this port. This field will be passed into the Get Cable EEPROM Data IPMI command to identify the specific cable port to fetch data.
09h	Cable Port Index 0 Start Lane	BYTE	Varies	One of 0, 4, 8 or 12 to indicate the starting lane of the x4 lane to which this specific cable port index applies to.
0Ah	Cable Port Index 1	BYTE	Varies	Optional/As needed to match the Cable Index Count field above.
0Bh	Cable Port Index 1 Start Lane	BYTE	Varies	Optional/As needed to match the Cable Index Count field above.
0Ch	Cable Port Index 2	BYTE	Varies	Optional/As needed to match the Cable Index Count field above.
0Dh	Cable Port Index 2 Start Lane	BYTE	Varies	Optional/As needed to match the Cable Index Count field above.
0Eh	Cable Port Index 3	BYTE	Varies	Optional/As needed to match the Cable Index Count field above.
0Fh	Cable Port index 3 Start Lane	BYTE	Varies	Optional/As needed to match the Cable Index Count field above.

3.11 SMBIOS Physical Device Mapping Information

SMBIOS OEM type 200 is used to declare the relationship between Physical devices to the SMBIOS representation. This allows BMC/PSME to identify and link the details as described in the SMBIOS tables to the actual physical location of the components/devices in the system.

The BIOS implements as many structures as needed to describe the relationship of existing components in the platform. Each SMBIOS physical device mapping table shall describe one type of physical device only. There can be multiple tables describing the same type of physical device if the number of physical devices of the same type exceeds the size of a single SMBIOS structure.

Table 16. SMBIOS Physical Device Mapping Table

Offset	Name	Length	Value	Description
00h	Type	BYTE	200	SMBIOS Physical Device Mapping Table indicator
01h	Length	BYTE	Varies	
02h	Handle	WORD	Varies	
04h	Physical Device Type	BYTE	Varies	Identifies the type of component this mapping table applies to. 00h – Undefined/Invalid 01h – Processor (Type 4 SMBIOS table) 02h – PCIe System Slots (Type 9 SMBIOS table describing PCIe) 03h – Memory Device (Type 17 SMBIOS table) 04h-FFh – Undefined/Invalid
05h	Reserved	BYTE	00h	Reserved
06h	Physical Device Mapping Structure 1	4 BYTES	Varies	Data describing the physical location of the device and the SMBIOS table handle that identifies the device in SMBIOS.
0Ah	Physical Device Mapping Structure 2	4 BYTES	Varies	Data describing the physical location of the device and the SMBIOS table handle that identifies the device in SMBIOS.
0Eh	Physical Device Mapping Structure …	4 BYTES	Varies	Data describing the physical location of the device and the SMBIOS table handle that identifies the device in SMBIOS.
…	Physical Device Mapping Structure n	4 BYTES	Varies	Data describing the physical location of the device and the SMBIOS table handle that identifies the device in SMBIOS.

Table 17. SMBIOS Processor Physical Device Mapping Structure

Offset	Name	Length	Value	Description
00h	SMBIOS Handle	WORD	Varies	Identifies the SMBIOS handle of the structure that defines the specific Processor physical device this structure refers to.
02h	Socket Number	BYTE	Varies	Identifies the zero based physical number of the socket the specific SMBIOS table describes.
03h	Reserved	BYTE	00h	Reserved and must be 0.

Table 18 shall be used to describe only the PCIe System slots present in the system and described by SMBIOS type 9 tables.

Table 18. SMBIOS PCIe System Slot Physical Device Mapping Structure

Offset	Name	Length	Value	Description
00h	SMBIOS Handle	WORD	Varies	Identifies the SMBIOS handle of the structure that defines the specific PCIe type system slot this structure refers to.

Offset	Name	Length	Value	Description
02h	Riser Number	BYTE	Varies	Identifies the zero based physical number of the riser the specific PCIe system slot is present.
03h	Slot Number	BYTE	Varies	Identifies the zero based physical number of the slot within the riser identifying the specific PCIe system slot is present.

Table 19. SMBIOS Memory Physical Device Mapping Structure

Offset	Name	Length	Value	Description
00h	SMBIOS Handle	WORD	Varies	Identifies the SMBIOS handle of the structure that defines the specific Memory device this structure refers to.
02h	CPU Socket Number	BYTE	Varies	Identifies the zero based physical number of the CPU this memory device belongs to.
03h	DIMM Slot Number	BYTE	Varies	Identifies the zero based physical number of the slot within the identified CPU the specific DIMM slot is present.

§

4.0 iSCSI Boot Options Support

4.1 Overview

The MDR mechanism is used to transfer OOB iSCSI configuration data from PSME into the BMC data repository. The BIOS retrieves this information from the BMC via the MDR interface during the boot sequence and applies the necessary iSCSI boot option changes and use them as appropriate. If a valid iSCSI boot options region is detected, then the BIOS parses the region and saves off the data for future use. The BIOS uses this stored information to attempt iSCSI boot when HDD is configured as the boot target using IPMI Set System Boot Options command.

The following sections define the format of the MDR data region holding the iSCSI boot options configuration data.

The iSCSI boot options MDR region is organized into a flat structure with various sub structures concatenated together similar to SMBIOS tables. The PSME writes the whole MDR region at any given time to provide a new set of iSCSI Boot option configurations. Presence of all structures (specifically iSCSI Boot Attempt structure) indicates to the BIOS that it has valid iSCSI configuration to process.

Note: If no structure is available in MDR region or if Get MDR Data Region Status IPMI command returns “Invalid Data”, then the BIOS must preserve the prior copy of the iSCSI configuration data. If the version structure is available, it indicates to the BIOS no iSCSI Configuration is available and the BIOS should delete its internal copies of prior iSCSI configuration data.

Note: All structures defined in this section follows SMBIOS specification for structure format. Refer to Section 6.1 of the *System Management BIOS (SMBIOS) Reference Specification* for more details.

4.2 STRING Definition

Text strings associated within a given iSCSI parameter structure are appended directly after the formatted portion of the structure. Each string is terminated with a null (00h) BYTE and the set of strings is terminated with an additional null (00h) BYTE. When the formatted portion of an iSCSI parameter structure references a string, it does so by specifying a non-zero string number within the structure’s string-set. For example, if a string field contains 02h, it references the second string following the formatted portion of the iSCSI parameter structure. If a string field references no string, a null (0) is placed in that string field. If the formatted portion of the structure contains string-reference fields and all the string fields are set to 0 (no string references), the formatted section of the structure is followed by two null (00h) BYTES.

4.3 iSCSI Boot Options MDR Region Layout

The MDR region for iSCSI Boot options is composed of several structures identified by a parameter id. Table 20 shows the high level layout of the MDR region.

Table 20. iSCSI Boot Options MDR Region Layout

Block Name	Parameter ID	Description
Version	1	Version parameter structure identifying the version of the iSCSI Boot Options MDR Layout. The first table in the iSCSI Boot Options MDR region must be of this type.
Initiator 1	2	One or more Initiator parameter structures defining the configuration of Initiator. Typically there can be as many Initiator parameter structures as there are NIC cards on the composed node. At least one Initiator structure is required for the system BIOS to attempt an iSCSI boot.
Initiator n	2	
Target 1	3	One or more target parameter structures defining the configuration of Target. At least one target structure is required for the system BIOS to attempt an iSCSI boot.
Target n	3	
iSCSI Attempt 1	4	One or more iSCSI Boot Attempt parameter structure defining the combinations of Initiator and Target to attempt to boot. The order of these structures define the priority used by the system BIOS to attempt an iSCSI boot. System BIOS attempts to boot with the first configuration. If that fails, it attempts the second configuration and so on.
iSCSI Attempt n	4	

4.3.1 Version (Parameter 1)

The Version parameter structure identifies the iSCSI Boot Options MDR region.

Table 21. Version Structure

Offset	Name	Length	Value	Description
00h	Parameter ID	BYTE	1	Version Parameter ID

Offset	Name	Length	Value	Description
01h	Length	WORD	07h	Size of the formatted section of this structure.
03h	Handle	WORD	Varies	Unused. Must be 0
05h	Major Version	BYTE	1	Major Version of the iSCSI Boot Options MDR region
06h	Minor Version	BYTE	0	Minor Version of the iSCSI Boot Options MDR region

4.3.2 Initiator (Parameter 2)

The Initiator parameter structure defines the Initiator configuration. There can be one or more of this structure defined in the iSCSI boot options MDR region.

Table 22. Initiator Structure

Offset	Name	Length	Value	Description
00h	Parameter ID	BYTE	2	Initiator Parameter ID
01h	Length	WORD	5Eh	Size of the formatted section of this structure.
03h	Handle	WORD	Varies	Unique value that identifies this specific structure across all iSCSI boot option structures
05h	Initiator DHCP Enabled	BYTE	Varies	0: Do not obtain IP info via DHCP 1: Obtain IP info via DHCP for the initiator
06h	Initiator Name	STRING	Varies	String number of the string that holds the initiator name
07h	IP Address Type	BYTE	Varies	0: Auto (try IPv4 first, then try IPv6) 1: IPv4 2: IPv6
08h	Initiator IP Address	16 BYTES	Varies	-
18h	Initiator Subnet Mask	16 BYTES	Varies	-
28h	Initiator Gateway	16 BYTES	Varies	-
38h	Primary DNS	16 BYTES	Varies	-
48h	Secondary DNS	16 BYTES	Varies	-
58h	NIC MAC Address	6 BYTES	Varies	MAC Address of the NIC adaptor to boot from for this Initiator

4.3.3 Target (Parameter 3)

The Target parameter structure defines the Target configuration. There can be one or more of this structure defined in the iSCSI boot options MDR region.

Table 23. Target Structure

Offset	Name	Length	Value	Description
00h	Parameter ID	BYTE	3	Target Parameter ID

Offset	Name	Length	Value	Description
01h	Length	WORD	25h	Size of the formatted section of this structure
03h	Handle	WORD	Varies	Unique value that identifies this specific structure across all iSCSI boot option structures
05h	Target DHCP Enabled	BYTE	Varies	0: Do not obtain Target details (boot target name, IP info) via DHCP 1: Obtain target details (boot target name, IP info) via DHCP
06h	Target Name	STRING	Varies	String number of the string that holds the iSCSI Qualified Name (IQN) of target
07h	IP Address Type	BYTE	Varies	0: Auto (try IPv4 first, then try IPv6) 1: IPv4 2: IPv6
08h	Target IP Address	16 BYTES	Varies	-
18h	Target Port	2 BYTES	Varies	-
1Ah	Target LUN	2 BYTES	Varies	-
1Ch	VLAN Enable	1 BYTE	Varies	0: VLAN is disabled for this target 1: VLAN is enabled for this target
1Dh	VLAN ID	2 BYTES	Varies	VLAN ID when VLAN is enabled
1Fh	Router Advertisement	1 BYTE	Varies	0: IPv6 Router Advertisement is disabled for this target 1: IPv6 Router Advertisement is enabled for this target
20h	Authentication Method	1 BYTE	Varies	0: No CHAP 1: CHAP 2: Mutual CHAP
21h	CHAP User Name	STRING	Varies	String number of the string that holds the CHAP user name
22h	CHAP Secret	STRING	Varies	String number of the string that holds the CHAP secret
23h	Mutual CHAP Name	STRING	Varies	String number of the string that holds the user name for 2 way CHAP authentication
24h	Mutual CHAP Secret	STRING	Varies	String number of the string that holds the secret for 2 way CHAP authentication

4.3.4 iSCSI Attempt (Parameter 4)

The iSCSI attempt parameter structure defines the pair of initiator and target configurations to be used for an iSCSI boot attempt. There can be one or more of this structure defined in the iSCSI boot options MDR region. The order of each structure in the MDR table defines the priority that must be used to attempt an iSCSI boot by the BIOS.

Table 24. iSCSI Attempt Structure

Offset	Name	Length	Value	Description
00h	Parameter ID	BYTE	4	iSCSI Attempt Parameter ID
01h	Length	WORD	0Ch	Size of the formatted section of this structure

Offset	Name	Length	Value	Description
03h	Handle	WORD	Varies	Unique value that identifies this specific structure across all iSCSI boot option structures
05h	Initiator Handle	WORD	Varies	Handle of the Initiator parameter structure that must be used for this specific iSCSI boot attempt
07h	Target Handle	WORD	Varies	Handle of the target parameter structure that must be used for this specific iSCSI boot attempt
09h	Connection Wait Time	WORD	Varies	Connection wait time in milliseconds
0Bh	Connection Retry Count	BYTE	Varies	Number of times this iSCSI attempt must be retried before moving to the next iSCSI attempt.

§

5.0 Miscellaneous Requirements

5.1 Management Network Exposure to Host OS

If the BMC Management Network interface is visible to the host OS, BIOS FW will hide the BMC Management Network interface to the Host OS on an Intel® RSD enabled platform. This allows a completely private management network to manage the compute node from Rack Scale Design perspective.

5.2 Link Layer Discovery Protocol Support

The BMC will optionally implement Link Layer Discovery Protocol (LLDP) support on the management network so the management NIC's MAC address, and other pertinent details, may be obtained by the RMM for location identification or other purposes.

§