



Intel[®] Rack Scale Design

BIOS & BMC Technical Guide

February 2017

Revision 003



No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and noninfringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services, and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications, and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Copies of documents that have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting <http://www.intel.com/design/literature.htm>.

Intel and the Intel logo are trademarks of Intel Corporation in the United States and other countries.

*Other names and brands may be claimed as the property of others

Copyright © 2017 Intel Corporation. All rights reserved.



Contents

1	Introduction	5
1.1	Nomenclature used in this document.....	5
1.2	References.....	6
2	Intel® RSD BMC IPMI Commands.....	7
3	Encoding Rules.....	24
3.1	Max length strings.....	24
3.2	Populating location info	24
4	Network Configuration.....	25
4.1	Static mode.....	25
4.2	DHCP mode	25
5	BMC / CPP Interaction	26
5.1	DHCP mode	26
6	BMC / BIOS iSCSI Interaction.....	27
7	Sensors.....	28
7.1	Intel® RSD NetFn.....	28
7.2	CPU health sensor	29
7.3	Memory health sensor	30
8	Node Discovery	32
8.1	Compute node information	32
8.2	BMC resident SMBIOS data storage	32
8.3	SMBIOS OEM records.....	32
8.3.1	NIC information (Type 190) definition.....	33
8.3.2	HD information (Type 191) definition.....	33
8.3.3	PCIe* information (Type 192) definition.....	34
8.4	Node insertion state.....	34

Tables

Table 1	Nomenclature	5
Table 2	Reference documents	6
Table 3	Intel® RSD BMC IPMI command table	7
Table 4	Sensors: Rackscale NetFn	28
Table 5	Sensors: CPU health sensor	29
Table 6	Sensors: memory health sensor.....	30
Table 7	NIC information (Type 190) structure definition	33
Table 8	HD information (Type 191) structure definition	33
Table 9	PCIe information (Type 192) structure definition	34



Revision History

Revision	Description	Date
003	Updated OEM SMBIOS records as version 0.02. Defined SMBIOS data retention policy. Modified Set/Get Serial Debug Selector commands for debug use only	February 2017
002	Compliant with Intel Legal. Add note to iSCSI Field commands to clarify that they only apply to the Legacy BootOpROM parameters. Add note about what should be done if the RackScale NetFn sensor is not found Add Managed Data Region (MDR) v1.0 command.	June 2016
001	Initial public release.	August 2015

§



1 Introduction

This document describes the extended BIOS/BMC functionalities to support the Intel® Rack Scale Design for Cloud.

1.1 Nomenclature used in this document

This document uses specific terms, codes, and abbreviations that are described in the table below.

Table 1 Nomenclature

Term	Type	Description	Examples
int	int	32 bit int	
byte	byte	8 bits	
boolbyte	byte	A byte that can only have a value of 1 or 0.	
string	string	113 byte (max) string (to allow for headers to make 128)	
Index	int	Unique int of Order or Location (Preferred to *always* start with 1, leave 0 reserved)	
GUID	intx4	Globally Unique (Arbitrary) ID (128 bits) xxxxx-xxxx-xxxxxxxxxx	21EC2020-3AEA-4069-A2DD-08002B30309D
Pod		A Collection of racks, managed by a Pod Manager	
Rack		A physical rack and a collection of trays managed by a RMM	
Tray		A unit that slides directly in a Rack, it may contain modules. This term is synonymous with "Chassis" and "Drawer".	
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	
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.	
CM		Control Module – a pluggable subcomponent of a Management Backplane assembly. It contains the active components of the MBP assembly.	
CPP		Control Plane Processor – a pluggable subcomponent of a tray switch assembly. This is also known as a TMC as it manages the tray.	
PUID		Pod Unique ID relative to POD	1-3-1
PRLoc	string	Same as PUID	1-3-1
DCUID		Data Center Unique ID relative to Datacenter	pod1.jf5.intel.com-1-3-1
FQLoc	string	Same as DCUID	pod1.jf5.intel.com-1-3-1
GeoTag	string	Unique User Defined string for a Geo Location	Oregon-JF5
PodDCUID	string	Pod Datacenter Unique ID = Pod Fully Qualified Host Name	pod1.jf5.intel.com
RackPUID	int	Rack Pod Unique ID. BDC uses the ordered Rack Number (Index) within the Pod	3 (means Rack3)
RackDCUID	string	PodDCUID-RackIndex	pod1.jf5.intel.com-1
RackPUName	string	Pod Unique user defined opaque string for the rack	DemoRack23



Term	Type	Description	Examples
RackBPID	int	Backplane ID within the Rack (starting at BPID 1). Backplanes are 8U high, so several (daisy chained) can exist in a rack.	1
TraySlotID	int	Tray Slot ID within Backplane. This is the GPIO* pin strap assignments on the backplane. It is only used internally (along with RackBPID) in the formula below to determine the TrayRUID	1 (int from 1 to 8)
TrayRUID	int	Tray (aka Chassis/Drawer) Slot Location within Rack. Calculated: ((RackBPID-1)*8)+TraySlotID	2 (means Tray 2 within Rack)
TrayPUID	string	RackPUID-TrayRUID	1-3... (means rack1-tray3)
ModuleTUID	int	Module (Sled) Slot within Tray (this could be a server node)	2 (means module/sled 2)
ModulePUID	string	TrayPUID-ModuleTUID	1-3-1 (rack1-tray3-module1)
NodeMUID	int	uServer (micro server) slot index within Module (not all modules have these)	4 (means uServer 4 within sled)
NodePUID	string	ModulePUID-NodeMUID	1-3-1-2 (rack1, tray3, module1, node2)
ComponentType	byte	Component Type Enum of byte type.	1=Pod, 2=Rack, 3=Tray, 4=Module, 5=Node, 6=MultiModule

1.2 References

Table 2 Reference documents

Doc ID	Title
335451	Intel® Rack Scale Design Generic Assets Management Interface API Specification
335452	Intel® Rack Scale Design BIOS & BMC Technical Guide
335501	Intel® Rack Scale Design Architecture Specification
335454	Intel® Rack Scale Design Software Reference Kit Getting Started Guide
335455	Intel® Rack Scale Design Pod Manager API Specification
335456	Intel® Rack Scale Design Pod Manager Release Notes
335457	Intel® Rack Scale Design Pod Manager User Guide
335458	Intel® Rack Scale Design PSME REST API Specification
335459	Intel® Rack Scale Design PSME Release Notes
335460	Intel® Rack Scale Design PSME User Guide
335461	Intel® Rack Scale Design Storage Services API Specification
335462	Intel® Rack Scale Design Rack Management Module (RMM) API Specification
335463	Intel® Rack Scale Design RMM Release Notes
335464	Intel® Rack Scale Design Software RMM User Guide
n/a	Scalable Platforms Management API





2 Intel® RSD BMC IPMI Commands

Table 3 defines the requests that the BMC accepts and the corresponding functionality and request/response data for these commands. These commands are sent to the BMC from the IPMB, LPC/KCS, or LAN interfaces.

For the base specification and descriptions of the BMC commands other than those specified in this chapter, see the Intelligent Platform Management Interface specification.

All Get commands require Privileged User access.

All Set commands require Privileged Admin access.

Table 3 Intel® RSD BMC IPMI command table

Net Function = Intel General Application ("Rackscale NetFn" Sensor Value), LUN = 00																							
Code	Command	Request, Response Data	Description																				
01h	Get Fan PWM	<p>Request: None</p> <p>Response: byte 1 – Completion code 00h = Normal 80h = Invalid context (e.g., sent to uServer Node BMC and not baseboard) byte 2 – Number of supported Fans byte 3 – Maximum of all the Fans' PWM byte 4 – Fan 0's PWM byte 5 – Fan 1's PWM bytes 6:N - Remaining "Fans" PWM</p>	This command returns the current required PWM value based on the thermal condition. It uses the local BMC FSC algorithm along with the FSC Configuration for the PWM calculation. It is primary for reference use only. The "Get Thermal State" command returns the actual baseboard thermal condition for Intel® RSD upper level FSC control.																				
02h	Set iSCSI Field	<p>Note: This command only affects the Legacy OpROM. If the BIOS is booting in Legacy mode it will read this data and set the Legacy Boot OpROM parameters.</p> <p>Request: byte 1 – Type of Field 00h = Default (legacy – ignored) byte 2 – Instance of Field.</p> <table border="1"> <thead> <tr> <th>Fld</th> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Boot Priority</td> <td>boolbyte</td> <td>1 = Primary, 2 = Secondary</td> </tr> <tr> <td>2</td> <td>Initiator name</td> <td>string</td> <td>The iSCSI Qualified Name (IQN) for the initiator</td> </tr> <tr> <td>3</td> <td>Initiator DHCP</td> <td>boolbyte</td> <td>Enable or disable to get initiator configuration from DHCP</td> </tr> <tr> <td>4</td> <td>Expansion Rom Menu</td> <td>boolbyte</td> <td>Enable or disable</td> </tr> </tbody> </table>	Fld	Name	Type	Description	1	Boot Priority	boolbyte	1 = Primary, 2 = Secondary	2	Initiator name	string	The iSCSI Qualified Name (IQN) for the initiator	3	Initiator DHCP	boolbyte	Enable or disable to get initiator configuration from DHCP	4	Expansion Rom Menu	boolbyte	Enable or disable	
Fld	Name	Type	Description																				
1	Boot Priority	boolbyte	1 = Primary, 2 = Secondary																				
2	Initiator name	string	The iSCSI Qualified Name (IQN) for the initiator																				
3	Initiator DHCP	boolbyte	Enable or disable to get initiator configuration from DHCP																				
4	Expansion Rom Menu	boolbyte	Enable or disable																				



Net Function = Intel General Application ("Rackscale NetFn" Sensor Value), LUN = 00																		
Code	Command	Request, Response Data				Description												
		5	Connection Wait Time	int	Connection Wait Time (5-60 in seconds)													
		6	Initiator IP	byte[4]	The initiator IPv4 address													
		7	Initiator Netmask	byte[4]	Initiator subnet mask													
		8	Gateway	byte[4]	Gateway IPv4 Address													
		9	iSCSIVlan	word	iSCSIVLan ID (0-4094)													
		10	Target Name	string	The iSCSI Qualified Name (IQN) for the target													
		11	Target DHCP	boolbyte	Enable or Disable to get the target information from DHCP Root Path													
		12	Target IP	byte[4]	The target IPv4 address													
		13	Target Port	word	The TCP port number on the target to connect to(1024-65535)													
		14	Boot LUN	byte	The LUN number to boot from(0-255)													
		15	Auth Method	byte	0 = None, 1 = CHAP, 2 = MultiCHAP													
		16	CHAP Username	string	Chap Username (max 32 characters)													
		17	CHAP Secret	string	Must be 12 to 16 characters long													
		18	Mutual CHAP Secret	string	Must be 12 to 16 characters long													
					iSCSI Boot Flags													
					<table border="1"> <thead> <tr> <th colspan="2">iSCSI Parameters Status Flag</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>OOB iSCSI Settings Disabled</td> </tr> <tr> <td>1</td> <td>OOB iSCSI Settings (enabled) in Sync with BIOS</td> </tr> <tr> <td>2</td> <td>Disable iSCSI Initiator</td> </tr> <tr> <td>3</td> <td>BMC Settings are Blank (new system)</td> </tr> <tr> <td>4</td> <td>Settings have been Updated</td> </tr> </tbody> </table>		iSCSI Parameters Status Flag		0	OOB iSCSI Settings Disabled	1	OOB iSCSI Settings (enabled) in Sync with BIOS	2	Disable iSCSI Initiator	3	BMC Settings are Blank (new system)	4	Settings have been Updated
iSCSI Parameters Status Flag																		
0	OOB iSCSI Settings Disabled																	
1	OOB iSCSI Settings (enabled) in Sync with BIOS																	
2	Disable iSCSI Initiator																	
3	BMC Settings are Blank (new system)																	
4	Settings have been Updated																	
		19	Boot Flags	byte[4]														
		20	NIC	byte	NIC number to boot from (default 0)													
		<p>byte 3 – Number of bytes to Set (ignored for non-string fields) byte 4:N – data bytes to Set into Field</p> <p>Response: byte 1 – Completion code 00h = Normal 80h = Invalid context (e.g., sent to uServer BMC and not node) C7h = Request data length invalid C8h = Request data file length limit exceeded C9h = Parameter out of range (or reserved) D3h = Node not present</p> <p>byte 2 – Maximum number of data bytes possible for Field.</p>																
03h	Get iSCSI Field	<p>Note: This command only affects the Legacy OpROM. If the BIOS is booting in Legacy mode it will read this data and set the Legacy Boot OpROM parameters.</p> <p>Request:</p>																



Net Function = Intel General Application ("Rackscale NetFn" Sensor Value), LUN = 00																																																																																							
Code	Command	Request, Response Data	Description																																																																																				
		byte 1 – Instance of Field. <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>Fld</th> <th>Name</th> <th>Type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Boot Priority</td> <td>boolbyte</td> <td>1 = Primary, 2 = Secondary</td> </tr> <tr> <td>2</td> <td>Initiator name</td> <td>string</td> <td>The iSCSI Qualified Name (IQN) for the initiator</td> </tr> <tr> <td>3</td> <td>Initiator DHCP</td> <td>boolbyte</td> <td>Enable or disable to get initiator configuration from DHCP</td> </tr> <tr> <td>4</td> <td>Expansion Rom Menu</td> <td>boolbyte</td> <td>Enable or disable</td> </tr> <tr> <td>5</td> <td>Connection Wait Time</td> <td>int</td> <td>Connection Wait Time (5-60 in seconds)</td> </tr> <tr> <td>6</td> <td>Initiator IP</td> <td>byte[4]</td> <td>The initiator IPv4 address</td> </tr> <tr> <td>7</td> <td>Initiator Netmask</td> <td>byte[4]</td> <td>Initiator subnet mask</td> </tr> <tr> <td>8</td> <td>Gateway</td> <td>byte[4]</td> <td>Gateway IPv4 Address</td> </tr> <tr> <td>9</td> <td>iSCSIVlan</td> <td>word</td> <td>iSCSIVlan ID (0-4094)</td> </tr> <tr> <td>10</td> <td>Target Name</td> <td>string</td> <td>The iSCSI Qualified Name (IQN) for the target</td> </tr> <tr> <td>11</td> <td>Target DHCP</td> <td>boolbyte</td> <td>Enable or Disable to get the target information from DHCP Root Path</td> </tr> <tr> <td>12</td> <td>Target IP</td> <td>byte[4]</td> <td>The target IPv4 address</td> </tr> <tr> <td>13</td> <td>Target Port</td> <td>word</td> <td>The TCP port number on the target to connect to(1024-65535)</td> </tr> <tr> <td>14</td> <td>Boot LUN</td> <td>byte</td> <td>The LUN number to boot from</td> </tr> <tr> <td>15</td> <td>Auth Method</td> <td>byte</td> <td>0 = None, 1 = CHAP, 2 = MultiCHAP</td> </tr> <tr> <td>16</td> <td>CHAP Username</td> <td>string[32]</td> <td>Chap Username (max 32 characters)</td> </tr> <tr> <td>17</td> <td>CHAP Secret</td> <td>string[32]</td> <td>Must be 12 to 16 characters long</td> </tr> <tr> <td>18</td> <td>Mutual CHAP Secret</td> <td>string[32]</td> <td>Must be 12 to 16 characters long</td> </tr> <tr> <td>19</td> <td>Boot Flags</td> <td>byte[4]</td> <td>iSCSI Boot Flags</td> </tr> <tr> <td>20</td> <td>NIC</td> <td>byte</td> <td>NIC number to boot from (default 0)</td> </tr> </tbody> </table> <p>Response: byte 1 – Completion code 00h = Normal 80h = Invalid context (e.g., sent to uServer BMC and not node) C7h = Request data length invalid C9h = Parameter out of range (or reserved) D3h = Node not present byte 2 – Maximum number of data bytes possible for Field. byte 3 – Number of data bytes actually gotten from Instance of Field bytes 4:N – Data bytes from the Field</p>		Fld	Name	Type	Description	1	Boot Priority	boolbyte	1 = Primary, 2 = Secondary	2	Initiator name	string	The iSCSI Qualified Name (IQN) for the initiator	3	Initiator DHCP	boolbyte	Enable or disable to get initiator configuration from DHCP	4	Expansion Rom Menu	boolbyte	Enable or disable	5	Connection Wait Time	int	Connection Wait Time (5-60 in seconds)	6	Initiator IP	byte[4]	The initiator IPv4 address	7	Initiator Netmask	byte[4]	Initiator subnet mask	8	Gateway	byte[4]	Gateway IPv4 Address	9	iSCSIVlan	word	iSCSIVlan ID (0-4094)	10	Target Name	string	The iSCSI Qualified Name (IQN) for the target	11	Target DHCP	boolbyte	Enable or Disable to get the target information from DHCP Root Path	12	Target IP	byte[4]	The target IPv4 address	13	Target Port	word	The TCP port number on the target to connect to(1024-65535)	14	Boot LUN	byte	The LUN number to boot from	15	Auth Method	byte	0 = None, 1 = CHAP, 2 = MultiCHAP	16	CHAP Username	string[32]	Chap Username (max 32 characters)	17	CHAP Secret	string[32]	Must be 12 to 16 characters long	18	Mutual CHAP Secret	string[32]	Must be 12 to 16 characters long	19	Boot Flags	byte[4]	iSCSI Boot Flags	20	NIC	byte	NIC number to boot from (default 0)
Fld	Name	Type	Description																																																																																				
1	Boot Priority	boolbyte	1 = Primary, 2 = Secondary																																																																																				
2	Initiator name	string	The iSCSI Qualified Name (IQN) for the initiator																																																																																				
3	Initiator DHCP	boolbyte	Enable or disable to get initiator configuration from DHCP																																																																																				
4	Expansion Rom Menu	boolbyte	Enable or disable																																																																																				
5	Connection Wait Time	int	Connection Wait Time (5-60 in seconds)																																																																																				
6	Initiator IP	byte[4]	The initiator IPv4 address																																																																																				
7	Initiator Netmask	byte[4]	Initiator subnet mask																																																																																				
8	Gateway	byte[4]	Gateway IPv4 Address																																																																																				
9	iSCSIVlan	word	iSCSIVlan ID (0-4094)																																																																																				
10	Target Name	string	The iSCSI Qualified Name (IQN) for the target																																																																																				
11	Target DHCP	boolbyte	Enable or Disable to get the target information from DHCP Root Path																																																																																				
12	Target IP	byte[4]	The target IPv4 address																																																																																				
13	Target Port	word	The TCP port number on the target to connect to(1024-65535)																																																																																				
14	Boot LUN	byte	The LUN number to boot from																																																																																				
15	Auth Method	byte	0 = None, 1 = CHAP, 2 = MultiCHAP																																																																																				
16	CHAP Username	string[32]	Chap Username (max 32 characters)																																																																																				
17	CHAP Secret	string[32]	Must be 12 to 16 characters long																																																																																				
18	Mutual CHAP Secret	string[32]	Must be 12 to 16 characters long																																																																																				
19	Boot Flags	byte[4]	iSCSI Boot Flags																																																																																				
20	NIC	byte	NIC number to boot from (default 0)																																																																																				
04h	Set Structure Block	Command removed from BMC.	Deprecated command—supplanted with Set iSCSI Field Command.																																																																																				
05h	Get Structure Block	Command removed from BMC.	Deprecated command—supplanted with Get iSCSI Field Command.																																																																																				



Net Function = Intel General Application ("Rackscale NetFn" Sensor Value), LUN = 00			
Code	Command	Request, Response Data	Description
06h	Set ID Field	<p>Request:</p> <p>byte 1 – Type of Field 00h = Default</p> <p>byte 2 – Instance of Field. 00h = Reserved 01h = ComponentType (reserved/RO) 02h = GUID (for programming idrom) 03h = GeoTag 04h = PodDCUID 05h = RackPUIID 06h = RackPUName 07h = RackDCUID (derived on Rack) 08h = RackBPID (for Tray Only/RO) 09h = TrayRUID (derived on Tray) 0Ah = TrayPUIID (derived on Tray) 0Bh = ModuleTUID (GPIO slot id, 1-based/RO) 0Ch = ModulePUIID (derived on Module/RO) 0Dh = NodeMUID (uServer only 1-based/RO) 0Eh = NodePUIID (uServer only derived/RO) 0Fh = ClientID (See section 3.2/RO)</p> <p>byte 3 – Number of bytes to Set (ignored for non-string fields) byte 4:N – data bytes to Set into Field</p> <p>Response:</p> <p>byte 1 – Completion code 00h = Normal 80h = Field is unsupported 82h = Field is Read-only C1h = Invalid command (e.g. uServer field request directed to non-uServer) C7h = Request data length invalid C8h = Request data file length limit exceeded C9h = Parameter out of range (or reserved) D3h = Node not present</p> <p>byte 2 – Maximum number of data bytes possible for Field.</p>	<p>Instead of Structure Block commands, the Field command allows an individual field to be set/get via any interfaces. The maximum bytes for the command is 128.</p>
07h	Get ID Field	<p>Request:</p> <p>byte 1 – Instance of Field. 00h = Reserved 01h = ComponentType 1=Pod 2=rack 3=tray 4=module 5=sub-node 6=uServer (Multi-module)</p> <p>02h = GUID 03h = GeoTag 04h = PodDCUID 05h = RackPUIID 06h = RackPUName</p>	



Net Function = Intel General Application ("Rackscale NetFn" Sensor Value), LUN = 00			
Code	Command	Request, Response Data	Description
		07h = RackDCUID (derived on Rack) 08h = RackBPID (only available on Tray) 09h = TrayRUID (derived on Tray) 0Ah = TrayPUID (derived on Tray) 0Bh = ModuleTUID (GPIO slot id, 1-based) 0Ch = ModulePUID (derived on Module) 0Dh = NodeMUID (uServer only 1-based) 0Eh = NodePUID (uServer only derived) 0Fh = ClientID (See section 3.2: derived) Response: byte 1 – Completion code 00h = Normal 80h = Field is unsupported or invalid for context (e.g., node specific field request directed to baseboard) C1h = Invalid command (e.g. uServer field request directed to non-uServer) C7h = Request data length invalid C8h = Request data file length limit exceeded C9h = Parameter out of range (or reserved) D3h = Node not present byte 2 – Maximum number of data bytes possible for Field. byte 3 – Number of data bytes actually gotten from Instance of Field bytes 4:N – Data bytes from the Field	
08h	Get Thermal State	Request: None Response: byte 1 – Completion code 00h = Normal 80h = Invalid context (e.g., sent to uServer Node BMC and not baseboard) byte 2: Thermal state 00h = Payload is powered off 01h = Below normal operating temperture. 02h = At normal operating temperature. 03h = Warm but acceptable range. 04h = Above warm operating rnage. 05h = At the high end of threshold. 06h = Outside of proper operating range. 07h = Criital state.	This command returns the current baseboard thermal state described in "Intel® RSD Thermal Control" document.
09h	Get Slot ID	Request: None Response: byte 1 – Completion code 00h = Normal D3h = Node not present byte 2 – Slot ID byte 3 – Board ID byte 4 – Revision ID byte 5 – Reserved, presently 0x00	Get the Slot ID in the tray; also the Board ID bits and Revision ID bits on server. Bytes 3-5 will be zero on uServer.



Net Function = Intel General Application ("Rackscale NetFn" Sensor Value), LUN = 00			
Code	Command	Request, Response Data	Description
0Ah	Get Node Info	<p>Request: None</p> <p>Response: byte 1 – Completion code 00h = Normal 80h = Invalid context (e.g., sent to uServer Node BMC and not baseboard) C1h = Invalid command (e.g. uServer related request directed to non-uServer) byte 2 – Number of Nodes supported byte 3 – Node 0 Info [1:7] – Reserved [0] – Node Present byte 4 – Node 1 Info [1:7] – Reserved [0] – Node Present byte 5 – Node 2 Info [1:7] – Reserved [0] – Node Present byte 6 – Node 3 Info [1:7] – Reserved [0] – Node Present byte 7 – Node 4 Info [1:7] – Reserved [0] – Node Present byte 8 – Node 5 Info [1:7] – Reserved [0] – Node Present byte 9 – Node 6 Info [1:7] – Reserved [0] – Node Present byte 10 – Node 7 Info [1:7] – Reserved [0] – Node Present byte 11 – Node 8 Info [1:7] – Reserved [0] – Node Present byte 12 – Node 9 Info [1:7] – Reserved [0] – Node Present byte 13 – Node 10 Info [1:7] – Reserved [0] – Node Present byte 14 – Node 11 Info [1:7] – Reserved [0] – Node Present</p>	<p>This command returns a byte map indicating the presence on the nodes. uServer only (not implemented on server as it is not relevant)</p>



Net Function = Intel General Application ("Rackscale NetFn" Sensor Value), LUN = 00			
Code	Command	Request, Response Data	Description
0Bh	Set Cooling Mode	<p>Request: byte 1 – Cooling Mode [0] 0 = Tray Fans, 1 = Rack Fans. [1..7] Reserved</p> <p>Response: byte 1 – Completion code 00h = Normal 80h = Invalid context (e.g., sent to uServer Node BMC and not baseboard)</p>	Set by Tray Manager as part of startup. If fans are set to "Rack Fans", the BMC should still generate a PWM (as if the fans were there), but not monitor tach (since it does not have this) and not generate the responses that it would typically generate if the tach was out of range.
0Ch	Get Cooling Mode	<p>Request: None</p> <p>Response: byte 1 – Completion code 00h = Normal 80h = Invalid context (e.g., sent to uServer Node BMC and not baseboard) byte 2 – Cooling Mode [0] 0 = Tray Fans, 1 = Rack Fans. [1..7] Reserved</p>	
0Dh	Set Config Complete	<p>Request: None</p> <p>Response: byte 1 – Completion code 00h = Normal 80h = Invalid context (e.g., sent to uServer Node BMC and not baseboard)</p>	
0Eh	Set Config PCIe Retimer	<p>Request: None</p> <p>Response: byte 1 – Completion code 00h = Normal 80h = Invalid context (e.g., sent to uServer Node BMC and not baseboard)</p>	BMC debug command for internal use only.
0Fh	Get uBMC Versions	<p>Request: None</p> <p>Response: byte 1 – Completion code 00h = Normal D3h = Node not present byte 2 – uBMC FW version major byte 3 – uBMC FW version minor byte 4 – uBMC boot loader version major byte 5 – uBMC boot loader version minor</p>	uBMC on uServer node.
10h	Reset Node Bridge-IC	<p>Request: byte 1:3 – Vendor specific magic value (Refer to vendor for magic value)</p>	Reset Node's Bridge-IC (used by firmware update)



Net Function = Intel General Application ("Rackscale NetFn" Sensor Value), LUN = 00			
Code	Command	Request, Response Data	Description
		<p>Response: byte 1 – Completion code 00h = Normal 80h = Invalid context (only valid when sent to node) C7h = Invalid length C9h = Invalid parameter (magic value) D3h = Node not present FFh = Unspecified Error (problem talking to Bridge-IC)</p>	
11h	Set Node MAC Address	<p>Request: byte 1 – 0-based network interface index byte 2 – PCI bus number for the device byte 3 – PCI device number for the device byte 4 – PCI function number for the device byte 5:10 – MAC address</p> <p>Response: byte 1 – Completion code 00h = Normal C7h = Invalid length C9h = Invalid parameter D3h = Node not present</p>	Set compute node MAC address.
12h	Get Node MAC Address	<p>Request: byte 1 – 0-based network interface index</p> <p>Response: byte 1 – Completion code 00h = Normal C7h = Invalid length C9h = Invalid parameter D3h = Node not present byte 2 – 0-based network interface index byte 3 – PCI bus number for the device byte 4 – PCI device number for the device byte 5 – PCI function number for the device byte 6:11 – MAC address</p>	Get compute node MAC address.
13h	Set Node Insertion State	<p>Request: byte 1 – Node insertion state 01h = Node insertion occurred All other values are reserved.</p> <p>Response: byte 1 – Completion code 00h = Normal C7h = Invalid length C9h = Invalid parameter</p>	Set node insertion state. This command is called when this node insertion was detected by its parent.
14h	Set OOB Boot Options	<p>The Set/Get OOB Boot Options command set is the extension of the Set/Get iSCSI Field commands. It is used to direct the BIOS on the system OOB boot following a sled power up or reset.</p> <p>Notes: For multi-byte parameter data field, it makes LSB first and MSB last.</p>	



Net Function = Intel General Application ("Rackscale NetFn" Sensor Value), LUN = 00																																																		
Code	Command	Request, Response Data	Description																																															
		<p>Request: byte 1 – [7] = Parameter Valid Flag. This is a "global" valid flag for all parameters, not just for a single parameter. 0b – Updates are in-progress for all parameters. Parameters are NOT ready for query. 1b – All parameter updates are completed. Parameters are valid to use. [6:0] = Parameter ID.</p>																																																
		<table border="1"> <thead> <tr> <th>Parameter ID</th> <th>Data Length</th> <th>Parameter Data</th> <th>Data Description</th> </tr> </thead> <tbody> <tr> <td rowspan="4">0 (Control Parameter)</td> <td>1</td> <td>Parameter Version #</td> <td>Parameter Version: 2 The Control Parameter Version does not only define the Control Parameter structure, but also defines the overall supporting parameters and types. In the Version #1, it supports Control, Initiator, NIC, Target and Heap parameters.</td> </tr> <tr> <td>1</td> <td>Parameter Index</td> <td>Parameter Index: 0</td> </tr> <tr> <td>4</td> <td>Boot Flags</td> <td>OOB iSCSI Settings Flag: 0: OOB iSCSI Settings in BIOS is disabled. 1: OOB iSCSI Settings in BIOS is enabled.</td> </tr> <tr> <td>6</td> <td>Total Data Length for Parameter</td> <td></td> </tr> <tr> <td></td> <td></td> <td> <table border="1"> <thead> <tr> <th>Parameter ID</th> <th>Data Length</th> <th>Parameter Data</th> <th>Data Description</th> </tr> </thead> <tbody> <tr> <td rowspan="6">1 (Initiator Parameter)</td> <td>1</td> <td>Parameter Version #</td> <td>Parameter Version: 2</td> </tr> <tr> <td>1</td> <td>Parameter Index</td> <td>Parameter Index: 0</td> </tr> <tr> <td>2</td> <td>Initiator Name Offset</td> <td>Offset from the beginning of the Heap Parameter for the iSCSI Qualified Name (IQN) for the Initiator.</td> </tr> <tr> <td>2</td> <td>Initiator Name Length</td> <td>Initiator Name string length. The terminating NULL is not counted as part of the string length and is not in the Heap. Zero (0) length means no data entry in the Heap Parameter.</td> </tr> <tr> <td>2</td> <td>Connection Wait Time</td> <td>Connection Wait Time in milliseconds (100 - 20000 milliseconds).</td> </tr> <tr> <td>2</td> <td>Connection Retry Count</td> <td>Connection Retry Count (0 - 16).</td> </tr> <tr> <td></td> <td>10</td> <td>Total Data Length for Parameter</td> <td></td> </tr> </tbody> </table> </td> <td></td> </tr> </tbody> </table>	Parameter ID	Data Length	Parameter Data	Data Description	0 (Control Parameter)	1	Parameter Version #	Parameter Version: 2 The Control Parameter Version does not only define the Control Parameter structure, but also defines the overall supporting parameters and types. In the Version #1, it supports Control, Initiator, NIC, Target and Heap parameters.	1	Parameter Index	Parameter Index: 0	4	Boot Flags	OOB iSCSI Settings Flag: 0: OOB iSCSI Settings in BIOS is disabled. 1: OOB iSCSI Settings in BIOS is enabled.	6	Total Data Length for Parameter				<table border="1"> <thead> <tr> <th>Parameter ID</th> <th>Data Length</th> <th>Parameter Data</th> <th>Data Description</th> </tr> </thead> <tbody> <tr> <td rowspan="6">1 (Initiator Parameter)</td> <td>1</td> <td>Parameter Version #</td> <td>Parameter Version: 2</td> </tr> <tr> <td>1</td> <td>Parameter Index</td> <td>Parameter Index: 0</td> </tr> <tr> <td>2</td> <td>Initiator Name Offset</td> <td>Offset from the beginning of the Heap Parameter for the iSCSI Qualified Name (IQN) for the Initiator.</td> </tr> <tr> <td>2</td> <td>Initiator Name Length</td> <td>Initiator Name string length. The terminating NULL is not counted as part of the string length and is not in the Heap. Zero (0) length means no data entry in the Heap Parameter.</td> </tr> <tr> <td>2</td> <td>Connection Wait Time</td> <td>Connection Wait Time in milliseconds (100 - 20000 milliseconds).</td> </tr> <tr> <td>2</td> <td>Connection Retry Count</td> <td>Connection Retry Count (0 - 16).</td> </tr> <tr> <td></td> <td>10</td> <td>Total Data Length for Parameter</td> <td></td> </tr> </tbody> </table>	Parameter ID	Data Length	Parameter Data	Data Description	1 (Initiator Parameter)	1	Parameter Version #	Parameter Version: 2	1	Parameter Index	Parameter Index: 0	2	Initiator Name Offset	Offset from the beginning of the Heap Parameter for the iSCSI Qualified Name (IQN) for the Initiator.	2	Initiator Name Length	Initiator Name string length. The terminating NULL is not counted as part of the string length and is not in the Heap. Zero (0) length means no data entry in the Heap Parameter.	2	Connection Wait Time	Connection Wait Time in milliseconds (100 - 20000 milliseconds).	2	Connection Retry Count	Connection Retry Count (0 - 16).		10	Total Data Length for Parameter		
Parameter ID	Data Length	Parameter Data	Data Description																																															
0 (Control Parameter)	1	Parameter Version #	Parameter Version: 2 The Control Parameter Version does not only define the Control Parameter structure, but also defines the overall supporting parameters and types. In the Version #1, it supports Control, Initiator, NIC, Target and Heap parameters.																																															
	1	Parameter Index	Parameter Index: 0																																															
	4	Boot Flags	OOB iSCSI Settings Flag: 0: OOB iSCSI Settings in BIOS is disabled. 1: OOB iSCSI Settings in BIOS is enabled.																																															
	6	Total Data Length for Parameter																																																
		<table border="1"> <thead> <tr> <th>Parameter ID</th> <th>Data Length</th> <th>Parameter Data</th> <th>Data Description</th> </tr> </thead> <tbody> <tr> <td rowspan="6">1 (Initiator Parameter)</td> <td>1</td> <td>Parameter Version #</td> <td>Parameter Version: 2</td> </tr> <tr> <td>1</td> <td>Parameter Index</td> <td>Parameter Index: 0</td> </tr> <tr> <td>2</td> <td>Initiator Name Offset</td> <td>Offset from the beginning of the Heap Parameter for the iSCSI Qualified Name (IQN) for the Initiator.</td> </tr> <tr> <td>2</td> <td>Initiator Name Length</td> <td>Initiator Name string length. The terminating NULL is not counted as part of the string length and is not in the Heap. Zero (0) length means no data entry in the Heap Parameter.</td> </tr> <tr> <td>2</td> <td>Connection Wait Time</td> <td>Connection Wait Time in milliseconds (100 - 20000 milliseconds).</td> </tr> <tr> <td>2</td> <td>Connection Retry Count</td> <td>Connection Retry Count (0 - 16).</td> </tr> <tr> <td></td> <td>10</td> <td>Total Data Length for Parameter</td> <td></td> </tr> </tbody> </table>	Parameter ID	Data Length	Parameter Data	Data Description	1 (Initiator Parameter)	1	Parameter Version #	Parameter Version: 2	1	Parameter Index	Parameter Index: 0	2	Initiator Name Offset	Offset from the beginning of the Heap Parameter for the iSCSI Qualified Name (IQN) for the Initiator.	2	Initiator Name Length	Initiator Name string length. The terminating NULL is not counted as part of the string length and is not in the Heap. Zero (0) length means no data entry in the Heap Parameter.	2	Connection Wait Time	Connection Wait Time in milliseconds (100 - 20000 milliseconds).	2	Connection Retry Count	Connection Retry Count (0 - 16).		10	Total Data Length for Parameter																						
Parameter ID	Data Length	Parameter Data	Data Description																																															
1 (Initiator Parameter)	1	Parameter Version #	Parameter Version: 2																																															
	1	Parameter Index	Parameter Index: 0																																															
	2	Initiator Name Offset	Offset from the beginning of the Heap Parameter for the iSCSI Qualified Name (IQN) for the Initiator.																																															
	2	Initiator Name Length	Initiator Name string length. The terminating NULL is not counted as part of the string length and is not in the Heap. Zero (0) length means no data entry in the Heap Parameter.																																															
	2	Connection Wait Time	Connection Wait Time in milliseconds (100 - 20000 milliseconds).																																															
	2	Connection Retry Count	Connection Retry Count (0 - 16).																																															
	10	Total Data Length for Parameter																																																



Parameter ID	Data Length	Parameter Data	Data Description
2 (NIC Parameter)	1	Parameter Version #	Parameter Version: 2
	1	Parameter Index	0: NIC 0 1: NIC 1 ... N: NIC N
	1	NIC DHCP Enabled	Enabled/Disabled to obtain the NIC IP info from DHCP Root Path. 0: Disabled 1: Enabled
	1	NIC DHCP IP Version	IP version for initiator to initiate DHCP. 0: Auto (try IPv4 first, then try IPv6) 1: IPv4 2: IPv6
	16	NIC IP Address	All zeros represent no IP address specified.
	16	NIC Subnet Mask	
	16	NIC Gateway	Follow the IP Address rule above.
	16	Primary DNS	Follow the IP Address rule above.
	16	Secondary DNS	Follow the IP Address rule above.
	2	Host Name Offset	Offset from the beginning of the Heap Parameter for the host name.
	2	Host Name Length	Host Name string length. The terminating NULL is not counted as part of the string length and is not in the Heap. Zero (0) length means no data entry in the Heap Parameter.
	1	VLAN Mode	0: VLAN Disabled 1: VLAN Enabled
	2	VLAN ID	VLAN ID of (0 – 4094).
	6	MAC Address	MAC Address of NIC card where iSCSI boot from. All 0 means using System Default NIC.
Total Data Length for Parameter	97		
Parameter ID	Data Length	Parameter Data	Data Description
3 (Target Parameter)	1	Parameter Version #	Parameter Version: 2
	1	Parameter Index	Target indexes specify target boot order. (Target 0 is the Primary Target, Target 1 is the Secondary Target, etc.) 0: Target 0 1: Target 1 ... N: Target N



Net Function = Intel General Application ("Rackscale NetFn" Sensor Value), LUN = 00				
Code	Command	Request, Response Data		Description
		2	Target Name Offset	Offset from the beginning of the Heap Parameter for the iSCSI Qualified Name (IQN) for the target.
		2	Target Name Length	Target Name string length. The terminating NULL is not counted as part of the string length and is not in the Heap. Zero (0) length means no data entry in the Heap Parameter.
		1	Target DHCP Enabled	Enabled/Disabled to obtain the target info from DHCP Root Path. 0: Disabled 1: Enabled
		16	Target IP Address	All zeros represent no IP address specified.
		2	Target Port	The TCP port number on the target to connect to (0 – 65535).
		8	Target Boot LUN	The LUN number to boot from.
		1	CHAP Type	0: No CHAP 1: CHAP 2: Mutual CHAP
		2	CHAP Name Offset	Offset from the beginning of the Heap Parameter for the CHAP Username string.
		2	CHAP Name Length	CHAP Username string length. The terminating NULL is not counted as part of the string length and is not in the Heap. Zero (0) length means no data entry in the Heap Parameter.
		2	CHAP Secret Offset	Offset from the beginning of the Heap Parameter for the CHAP Secret string.
		2	CHAP Secret Length	CHAP Secret string length. The terminating NULL is not counted as part of the string length and is not in the Heap. Zero (0) length means no data entry in the Heap Parameter.
		2	Mutual CHAP Name Offset	Offset from the beginning of the Heap Parameter for the Mutual CHAP Username string.
		2	Mutual CHAP Name Length	Mutual CHAP Username string length. 0 if CHAP Type is not Mutual CHAP. The terminating NULL is not counted as part of the string length and is not in the Heap. Zero (0) length means no data entry in the Heap Parameter.
		2	Mutual CHAP Secret Offset	Offset from the beginning of the Heap Parameter for the Mutual CHAP Secret string.



Net Function = Intel General Application ("Rackscale NetFn" Sensor Value), LUN = 00					
Code	Command	Request, Response Data		Description	
			2	Mutual CHAP Secret Length Mutual CHAP Secret Length Mutual CHAP Secret string length. 0 if CHAP Type is not Mutual CHAP. The terminating NULL is not counted as part of the string length and is not in the Heap. Zero (0) length means no data entry in the Heap Parameter.	
		Total Data Length for Parameter	50		
		Parameter ID	Data Length	Parameter Data	Data Description
		4 (Heap Parameter)	2	Heap Length	Number of data bytes in the Heap Area
			2	Data bytes offset	Data location in the Heap. It is used for reading from or writing to the Heap area.
			N	Heap Area	The heap area in N bytes. The heap area can be accessed by given the Heap Offset and the number of data bytes.
		Parameter ID	Data Length	Parameter Data	Data Description
		7Fh (Admin Parameter)	1	Parameter Version #	Parameter Version: 2
			1	Clear all parameter data	1: Clear all parameter data Get returns 0.
		<p>byte 2:N – Data bytes to Set into the Parameter ID. Notes: For the Heap Parameter ID, the data bytes includes the Heap Length (2 bytes), the Data Offset (2 bytes) in the heap, and the data bytes (n bytes).</p> <p>Response: byte 1 – Completion code 00h = Normal 80h = Invalid context (e.g., sent to uServer BMC and not node) C7h = Request data length invalid C8h = Request data field length limit exceeded C9h = Parameter out of range (or reserved) D3h = Node not present</p>			
15h	Get OOB Boot Options	<p>Request: byte 1 – Parameter ID. byte 2 – Parameter Index Notes: Parameter Index value is ignored for Heap Parameter ID 4 since the Heap parameter does not contain an index. For the Heap Parameter (4), the Heap Offset and the data length are required in the Request as follows: byte 3:4 – Data Offset in the Heap. byte 5:6 – Actual number of data bytes to query.</p> <p>Response:</p>			



Net Function = Intel General Application ("Rackscale NetFn" Sensor Value), LUN = 00			
Code	Command	Request, Response Data	Description
		byte 1 – Completion code 00h = Normal 80h = Invalid context (e.g., sent to uServer BMC and not node) 81h = Parameter update in-progress Note: To read parameter data, the global Parameter Valid Flag must be set to 1. C7h = Request data length invalid C9h = Parameter ID or Parameter Index out of range (or reserved) D3h = Node not present byte 2:3 – Number of data bytes for the Parameter ID. byte 4:5 – Number of data bytes actually gotten from this query. bytes 6:N – Data bytes from the Parameter	
16h	Get PCIe EEPROM Data	Request: byte 1 – Channel (0 to 3) byte 2 – Offset to read byte 3 – Data Length to be read Response: byte 1 – Completion code 00h = Normal 80h = Invalid context (only valid when sent to node) C7h = Invalid length C9h = Invalid parameter byte 2:N – Data	Get data from PCIe* EEPROM.
20h	Get MDR Data Region Status	Request: byte 1 – Data Region 01h = SMBIOS table All other values are reserved. Response: byte 1 – Completion code 00h = Normal 80h = Invalid context C9h = Invalid region specified byte 2 – MDR version byte 3 – Data region identifier 01h = SMBIOS table All other values are reserved. byte 4 – Data validation 00h = Invalid data 01h = Valid data All other values are reserved. byte 5 – Unique data contents update count byte 6 – Lock Status 00h = Unlocked 01h = Strict lock 02h = Preemptable Lock All other values are reserved. byte 7:8 – Maximum size of region in bytes byte 9:10 – Used size of region in bytes byte 11 – Region checksum	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 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 it is determined that the data needs to be refreshed, use the following read / write / lock commands to perform the update.



Net Function = Intel General Application ("Rackscale NetFn" Sensor Value), LUN = 00			
Code	Command	Request, Response Data	Description
21h	Set MDR Region Update Complete	<p>Request: byte 1 – Session Lock Handle byte 2 – Data Region. 01h = SMBIOS region All other values are reserved.</p> <p>Response: byte 1 – Completion code 00h = Normal 80h = Invalid context 81h = Region is in use by another user. C9h = Invalid Region or Session Specified. D5h = Region is not locked.</p>	<p>This command is mainly utilized by the BIOS to indicate that its update of the specified data region is complete. This command should not be used by an external application.</p> <p>Session Lock Handle is returned by the "Get MDR Region Lock" command.</p>
22h	MDR Region Read	<p>Request: byte 1 – Data Region. 01h = SMBIOS region All other values are reserved. byte 2 – Data Length to Read. byte 3:4 – Offset to read.</p> <p>Response: byte 1 – Completion code 00h = Normal 80h = Invalid context 81h = Region is in use by another user. C4h = Request is larger than maximum allowed payload size; Request (offset + length) is larger than region length used. C7h = Requested data extends beyond length of region C9h = Invalid Region Specified; Invalid length specified; Offset beyond region length used CBh = Region marked as invalid byte 2 – Read Length. byte 3 – Update Count byte 4:N – Data</p>	<p>This command is utilized by external applications to retrieve the specified data region.</p> <p>Length of 0 is invalid (returns C9h)</p> <p>Data is not returned if region is marked as invalid (CBh)</p>
23h	MDR Region Write	<p>Request: byte 1 – Session Lock Handle byte 2 – Data Region. 01h = SMBIOS region All other values are reserved. byte 3 – Data Length byte 4:5 – Data Offset. byte 6:N – Data to be written.</p> <p>Response: byte 1 – Completion code 00h = Normal 80h = Invalid context 81h = Region is in use by another user. C4h = Requested data extends beyond length of region</p>	<p>Session Lock Handle is returned by the "Get MDR Region Lock" command.</p> <p>Data length of 0 is invalid (will return C9h).</p> <p>Limitations of the IPMI stack may limit the maximum bytes possible to write (e.g., RMCPv1.0 limits to only ~250 bytes)</p> <p>The length of the region used is automatically increased (provided the write does not exceed the maximum region size) upon successful writes.</p>



Net Function = Intel General Application ("Rackscale NetFn" Sensor Value), LUN = 00			
Code	Command	Request, Response Data	Description
		C7h = insufficient or mismatch of parameters C9h = Invalid Region Specified; data length mismatches data provided; D5h = Data region not locked. byte 2 – Data Region. 01h = SMBIOS region All other values are reserved. byte 3 – Valid Data 00h = Invalid (will always be invalid, as the region needs to be locked to successfully write and a locked region is always marked invalid until marked complete) byte 4 – Lock Status 00h = Unlocked 01h = Strict Lock 02h = Preemptable Lock All other values are reserved. byte 5:6 – Max Region Length byte 7:8 – Size of region used (in bytes)	If a write will create a hole, (offset of write is greater than the current length of bytes used), the hole is NUL (0) padded.
24h	Get MDR Region Lock	Request: byte 1 – Session Lock Handle 00h = Request lock/session handle Any other value is an existing session handle to be unlocked. byte 2 – Data Region. 01h = SMBIOS region All other values are reserved. byte 3 – Lock Type 00h = Abort; unlock without completing operation. 01h = Strict Lock; only writes by user who locked the region are allowed. 02h = Preemptable lock; strict locks override this type of lock. All other values are reserved. byte 4:5 – Timeout. Number of milliseconds allowed before lock is released. Response: byte 1 – Completion code 00h = Normal 80h = Invalid context 81h = Region is in use by another user. C9h = Invalid Region Specified; Invalid lock type; D5h = Use of reserved value (0/0xff) for Session Parameter in (lock abort) context. byte 2 – Session Lock Handle; only valid if byte 1 is a successful code (00h).	This command locks the specified data region. Only the BIOS should use the strict lock. All other external applications should use the preemptible lock. The external application has to wait until BIOS completes the data region update. While locked, the region is marked invalid until Set MDR Region Update Complete is called. A new lock automatically sets the region length used to 0. If the lock is aborted, the region will be marked as invalid and region length used as 0. If the timeout value is 0, the lock is indefinite (until either aborted, or marked as complete) The region's update count is incremented upon a successful new session lock is created.
7Bh	Set Serial Debug Selector	Request: byte 1 – Current Serial Debug Selector 00h = Host to FP on server, Host Node 1 on uServer 01h = BMC to FP on server, Host Node 2 on uServer	BMC debug command for internal use only. Requires CPLD 12A2.



Net Function = Intel General Application (“Rackscale NetFn” Sensor Value), LUN = 00			
Code	Command	Request, Response Data	Description
		<p>02h = Host to Midplane and display only FP on server, Host Node 3 on uServer 03h = BMC to Midplane and display only FP on server, BMC on uServer</p> <p>Response: byte 1 – Completion code 00h = Normal 80h = Invalid command for context (not valid command for uServer node instance) C9h = Parameter out of range (or reserved) D5h = Invalid command for context (not valid command for uServer node instance)</p>	
7Ch	Get Serial Debug Selector	<p>Request: None</p> <p>Response: byte 1 – Completion code 00h = Normal byte 2 – Current Serial Debug Selector 00h = Host to FP on server, Host Node 1 on uServer 01h = BMC to FP on server, Host Node 2 on uServer 02h = Host to Midplane and display only FP on server, Host Node 3 on uServer 03h = BMC to Midplane and display only FP on server, BMC on uServer</p>	BMC debug command for internal use only. Requires CPLD 12A2.
7Dh	Set Debug Level and Flags	<p>Request: byte 1 – Debug Flags byte 2 – Debug Level.</p> <p>Response: byte 1 – Completion code 00h = Normal byte 2 – Debug Flags byte 3 – Debug Level</p>	BMC debug command for internal use only.
7Eh	Get Debug Level and Flags	<p>Request: byte 1 – (optional) extended debug level flags request</p> <p>Response: byte 1 – Completion code 00h = Normal byte 2 – Debug Flags [0] – Fan PWM [1] – Host MAC Addresses - Deprecated [2] – Block Structures [3] – iSCSI / ID Fields [4] – Display IPMI invalid command warning [5] – Display MDR diagnostics [6:7] – Reserved byte 3 – Debug Level 00h = Default, least amount of debug output.</p>	BMC debug command for internal use only.



Net Function = Intel General Application ("Rackscale NetFn" Sensor Value), LUN = 00			
Code	Command	Request, Response Data	Description
		FFh = Most amount of debug output.	
7Fh	Debug of fan PWM and Thermal State	<p>Request: byte 1 – Forced Thermal State byte 2 – Forced Fan PWM MAX value that will be returned</p> <p>Response: byte 1 – Completion code 00h = Normal byte 2 – Forced Thermal State byte 3 – Forced PWM MAX value that will be returned</p>	<p>The purpose of this OEM command is to aid higher level software testing without having to force the physical hardware into difficult states. The command manually sets the Thermal State and/or PWM value in the BMC FW only for testing used. Actual Thermal State and fan PWM value will not be changed using this command.</p> <p>To use this OEM command the Debug Flags must have bit [0] Fan PWM SET and Debug Level must be greater than 70, prior to calling. Also clear bit [0] Fan PWM or Debug Level dropping below 70 will revert back to normal standard non-forced non-faked Fan return values.</p> <p>Both parameters can take on ANY value from 0 to 255 (0xFF).</p>

§



3 Encoding Rules

3.1 Max length strings

Any string that exceeds the max length has to be truncated and the completion code is C8h.

3.2 Populating location info

The Hierarchy is Pod->Rack->Tray->Module->Node. Every component cascades down and passes the ID info it has to the components below it in the hierarchy as part of the initialization except where fields are designated as “only” for the given component (such as the RackBPID only belongs to the Tray). The portions of the ID that can be determined with hardware (such as GPIO pins) is derived from the hardware. Hardware-derived IDs are always “1-based” even if the hardware is “0-based”.





4 Network Configuration

4.1 Static mode

If the network configuration is set to “static ip” and the last octet of the BMC IP is set to 0, then the last octet of the IP is calculated to be the (1-based) Slot ID within the Tray as derived from hardware from the GPIO pins (not necessarily the exact values).

For a BDC three slot tray, the ID's are slots 1, 2, 3 from the front view, left to right.

4.2 DHCP mode

If the network configuration is set to “DHCP”, then the BMC DHCP request encodes the “Rackscale”, the Platform (not BMC) GUID, component-type, PRLoc (PUID) and Interface in the DHCP Request Client ID in the following format:

Rackscale:[GUID]:[component type string]-BMC:[PUID]:[interface-name]

Examples:

Server BMC:

Rackscale:21EC2020-3AEA-4069-A2DD-08002B30309D:Module-BMC:1-3-1:eth1

uServer (MultiModule) BMC:

Rackscale:21EC2020-3AEA-4069-A2DD-08002B30309E:MultiModule-BMC:1-3-1:bond0

Server Host:

Rackscale:21EC2020-3AEA-4069-A2DD-08002B30309F:Module-Host:1-3-1:eth0

Rackscale:21EC2020-3AEA-4069-A2DD-08002B30309F:Module-Host:1-3-1:eth1

uServer Host:

Rackscale:21EC2020-3AEA-4069-A2DD-08002B30309C:Module-Node:1-3-1-2:eth0

If the location information is not set by the hierarchy above, then only the missing portions of the PUID are filled in with zeros (0). Example: 0-0-1. The Slot ID portion is always filled.





5 BMC / CPP Interaction

5.1 DHCP mode

The BMC sends the Client ID to the CPP in the DHCP Request package. The Client ID is described in Section 4.2 above. The DHCP server on the CPP responds with the IP addresses of the router and the NTP server to the BMC.

§



6 BMC / BIOS iSCSI Interaction

The iSCSI parameters are configured in the BMC. The command to set the iSCSI parameters is described in Section 2 above. As part of the boot process the BIOS retrieves the iSCSI parameters from the BMC and applies those iSCSI parameters to the NICs.





7 Sensors

7.1 Intel® RSD NetFn

The “Rackscale NetFn” shall be implemented as an IPMI sensor as follows. It can be queried via IPMI sensor reading command. The sensor number 7Ah can be changed based on a user specific requirement. This sensor is only for the baseboard BMC and not on the uServer nodes.

Note: If the Rackscale NetFn sensor is not found then use NetFn 0x38.

Table 4 Sensors: Rackscale NetFn

Sensor Name	Sensor #	Entity ID	Instance	Sensor Type
Rackscale NetFn	0x7A	0x33	0x00	OEM 0xC0

The “Rackscale NetFn” sensor SDR record is defined as follows:

```
.db 0x51                /* SDR Version                */
.db 0x02                /* Record Type                */
.db 27 + sizeof " Rackscale NetFn" /* Record Length = 27 + string length */

/* RECORD KEY BYTES */
.db 0x20                /* Sensor Owner ID            */
.db 0x0                 /* Sensor Owner LUN           */
.db 0x7a                /* Sensor Number, e.g. 7Ah used here */

/* RECORD BODY BYTES */
.db 0x21                /* Entity ID                   */
.db 0x0                 /* Entity Instance            */
.db 0x45                /* Sensor Initialization       */
.db 0x40                /* Sensor Capabilities         */
.db 0xc0                /* Sensor Type                 */
.db 0x70                /* Event / Reading Type       */
.dw 0x00                /* Lower Threshold Reading Mask */
.dw 0x00                /* Upper Threshold Reading Mask */
.dw 0x00                /* Settable/Readable Threshold Mask */
.db 0x00                /* Sensor Units 1              */
.db 0x31                /* Sensor Units 2 - Base Unit  */
.db 0x00                /* Sensor Units 3 - Modifier Unit */
.dw 0x1                 /* Sensor Record Sharing       */
.db 0x00                /* Positive - threshold Hysteresis value */
.db 0x00                /* Negative - threshold Hysteresis value */
```



```

.db 0x00          /* Reserved          */
.db 0x00          /* Reserved          */
.db 0x00          /* Reserved          */
.db 0x0           /* OEM               */
.db 0xc0 + sizeof " Rackscale NetFn" /* ID String Type / Length Code */
.db " Rackscale NetFn"

```

7.2 CPU health sensor

The “CPU Health Sensor” is an aggregated processor health event sensor, and shall be implemented as a standard IPMI sensor type 07h as follows. All available sensor offsets shall be implemented accordingly to reflect the sled processor health based on a sled hardware specification. See the IPMI spec Table 42-3 on the Processor Sensor Type for details. The sensor can be queried via IPMI sensor reading command. The sensor number 7Bh can be changed based on a user specific requirement.

Table 5 Sensors: CPU health sensor

Sensor Name	Sensor #	Entity ID	Instance	Sensor Type
CPU Health	0x7B	0x25	0x00	0x07

The “CPU Health” sensor SDR record is defined as follows:

```

/* SDR Record Header */
.dw 0x88          /* Record ID          */
.db 0x51          /* SDR Version        */
.db 0x02          /* Record Type        */
.db 27 + sizeof "CPU Health" /* Record Length = 27 + string length */

/* RECORD KEY BYTES */
.db 0x20          /* Sensor Owner ID    */
.db 0x0           /* Sensor Owner LUN   */
.db 0x7b          /* Sensor Number      */

/* RECORD BODY BYTES */
.db 0x25          /* Entity ID          */
.db 0x0           /* Entity Instance    */
.db 0xe3          /* Sensor Initialization */
.db 0x40          /* Sensor Capabilities */
.db 0x7           /* Sensor Type        */
.db 0x6f          /* Event / Reading Type */
.dw 0x7fff        /* Lower Threshold Reading Mask */
.dw 0x7fff        /* Upper Threshold Reading Mask */

```



```

.dw 0xff /* Settable/Readable Threshold Mask */
.db 0xc0 /* Sensor Units 1 */
.db 0x0 /* Sensor Units 2 - Base Unit */
.db 0x0 /* Sensor Units 3 - Modifier Unit */
.dw 0x1 /* Sensor Record Sharing */
.db 0x0 /* Positive - threshold Hysteresis value */
.db 0x0 /* Negative - threshold Hysteresis value */
.db 0x00 /* Reserved */
.db 0x00 /* Reserved */
.db 0x00 /* Reserved */
.db 0x0 /* OEM */
.db 0xc0 + sizeof "CPU Health" /* ID String Type / Length Code */
.db "CPU Health"

```

7.3 Memory health sensor

The “Memory Health Sensor” is an aggregated memory health event sensor, and shall be implemented as a standard IPMI sensor type 0Ch as follows. All available sensor offsets shall be implemented accordingly to reflect the sled memory health based on a sled hardware specification. See the IPMI spec Table 42-3 on the Processor Sensor Type for details. The sensor can be queried via IPMI sensor reading command. The sensor number 7Ch can be changed based on a user specific requirement.

Table 6 Sensors: memory health sensor

Sensor Name	Sensor #	Entity ID	Instance	Sensor Type
Memory Health	0x7C	0x25	0x01	0x0C

The “Memory Health” sensor SDR record is defined as following:

```

/* SDR Record Header */
.dw 0x89 /* Record ID */
.db 0x51 /* SDR Version */
.db 0x02 /* Record Type */
.db 27 + sizeof "Memory Health" /* Record Length = 27 + string length */

/* RECORD KEY BYTES */
.db 0x20 /* Sensor Owner ID */
.db 0x0 /* Sensor Owner LUN */
.db 0x7c /* Sensor Number */

/* RECORD BODY BYTES */
.db 0x25 /* Entity ID */
.db 0x1 /* Entity Instance */

```



.db 0xe3	/* Sensor Initialization	*/
.db 0x40	/* Sensor Capabilities	*/
.db 0xc	/* Sensor Type	*/
.db 0x6f	/* Event / Reading Type	*/
.dw 0x7fff	/* Lower Threshold Reading Mask	*/
.dw 0x7fff	/* Upper Threshold Reading Mask	*/
.dw 0xff	/* Settable/Readable Threshold Mask	*/
.db 0xc0	/* Sensor Units 1	*/
.db 0x0	/* Sensor Units 2 - Base Unit	*/
.db 0x0	/* Sensor Units 3 - Modifier Unit	*/
.dw 0x1	/* Sensor Record Sharing	*/
.db 0x0	/* Positive - threshold Hysteresis value	*/
.db 0x0	/* Negative - threshold Hysteresis value	*/
.db 0x00	/* Reserved	*/
.db 0x00	/* Reserved	*/
.db 0x00	/* Reserved	*/
.db 0x0	/* OEM	*/
.db 0xc0 + sizeof "Memory Health"	/* ID String Type / Length Code	*/
.db "Memory Health"		

§

8 Node Discovery

8.1 Compute node information

SMBIOS is used for providing basic characteristics of the asset of a compute node. Such compute centric information includes:

- Processor serial number
- Processor manufacturer
- Processor type
- Processor architecture
- Processor speed
- Processor capability: mmx, vmx, avx, etc.
- CPUID for processor family, model, step, etc.
- Processor core count (total/enabled)
- Processor total thread count
- Memory part number
- Memory rank
- Memory form factor
- Memory manufacturer
- Memory bank/location
- Memory width (total/data)

8.2 BMC resident SMBIOS data storage

The BMC maintains a copy of a SMBIOS table in a multi-purpose Managed Data Region (MDR) in RAM as well as in its flash filesystem. This SMBIOS data can be retrieved using Intel® RSD BMC IPMI commands either in-band or out-of-band. Once retrieved, external applications may interpret this data by decoding the information as described by the current *SMBIOS Specification*.

On all system boots, the BIOS will determine whether the BMC has already obtained the same SMBIOS information. It is done by checking the length and the checksum of the SMBIOS table against the same in the BMC during POST. If there is no match, the BIOS sends a copy of the SMBIOS table to the BMC. After the complete SMBIOS table is received, the BMC should save off the content to its flash filesystem for next BMC restart. If there is a match, the BMC already has the SMBIOS data that matches from the BIOS. No SMBIOS data transfer is needed.

On a BMC reset or power on, it will use the SMBIOS data from its flash filesystem before BIOS sends a newer one. The BMC will clear the SMBIOS data as well as CPU, DIMM, and BIOS data on AC power on.

Table 3 lists a set of MDR commands for sending and retrieving data. Due to IPMI command length limitations, multiple calls to the Intel® RSD BMC MDR IPMI interface will be required to obtain all of the embedded software information.

8.3 SMBIOS OEM records

Rack Scale Design performs a Deep Discovery of each asset within a POD. Additional information is required to fully describe the asset, such as hard disks, PCIe devices and network devices. Therefore, SMBIOS OEM records are defined as follows:



8.3.1 NIC information (Type 190) definition

This structure defines the NIC attribute and PCI information.

Table 7 NIC information (Type 190) structure definition

Offset	Spec Version	Name	Length	Value	Description
00	2.0+	Type	1 BYTE	190	OEM Structure Type
01	2.0+	Length	1 BYTE	Varies	Size of the Table
02	2.0+	Handle	2 BYTES	Varies	A unique 16-bit number of structure's handle
04	2.0+	PCI Class Code	1 BYTE	Varies	PCI Class Code is used to identify the generic function of the device
05	2.0+	Slot Number	2 BYTES	Varies	Physical Slot Number of the Slot Connect to the PCIE port
07	2.0+	Vendor ID	2 BYTES	Varies	Vendor Identification Number
09	2.0+	Device ID	2 BYTES	Varies	Device Identification Number
0B	2.0+	SubVendor ID	2 BYTES	Varies	Sub system ID and Sub Vendor ID Differentiate specific model
0D	2.0+	SubDevice ID	2 BYTES	Varies	Sub system ID and Sub Vendor ID Differentiate specific model
0F	2.0+	Link Speed	4 BYTES	Varies	Current Link Speed
13	2.0+	Link Width	4 BYTES	Varies	Negotiated Link Width
17	2.0+	Port Index	2 BYTES	Varies	Port Index in the system
19	2.0+	MAC Address	32 BYTES	Varies	Network MAC address

8.3.2 HD information (Type 191) definition

This structure defines the HDD attributes and the PCI information.

Table 8 HD information (Type 191) structure definition

Offset	Spec Version	Name	Length	Value	Description
00	2.0+	Type	1 BYTES	191	OEM Structure Type
01	2.0+	Length	1 BYTE	Varies	Size of the Table
02	2.0+	Handle	2 BYTES	Varies	A unique 16-bit number of structure's handle
04	2.0+	PCI Class Code	1 BYTE	Varies	PCI Class Code is used to identify the generic function of the device
05	2.0+	PCI Sub Class Code	1 BYTE	Varies	PCI Sub Class Code
06	2.0+	Vendor ID	2 BYTES	Varies	Vendor Identification Number
08	2.0+	Device ID	2 BYTES	Varies	Device Identification Number
0A	2.0+	SubVendor ID	2 BYTES	Varies	Sub system ID and Sub Vendor ID Differentiate specific model
0C	2.0+	SubDevice ID	2 BYTES	Varies	Sub system ID and Sub Vendor ID Differentiate specific model
0E	2.0+	HDD Index	1 BYTE	Varies	HD index in the system
0F	2.0+	HDD Name	40 BYTES	STRING	40 Bytes HD Brand Name
36	2.0+	HDD Serial Number	20 BYTES	STRING	20 Bytes of HDD's serial number



Offset	Spec Version	Name	Length	Value	Description
49	2.0+	HDD Type	1 BYTE	Varies	HDD type
4A	2.0+	HDD Size	4 BYTES	Varies	HD Size in GB

8.3.3 PCIe* information (Type 192) definition

This structure defines the PCI device information.

Table 9 PCIe information (Type 192) structure definition

Offset	Spec Version	Name	Length	Value	Description
00	2.0+	Type	1 BYTE	192	OEM Structure Type
01	2.0+	Length	1 BYTE	Varies	Size of the Table
02	2.0+	Handle	2 BYTES	Varies	A unique 16-bit number of structure's handle
04	2.0+	PCI Class Code	1 BYTE	Varies	PCI Class Code is used to identify the generic function of the device
05	2.0+	Slot Number	2 BYTES	Varies	Physical Slot Number of the Slot Connect to the PCIE port
07	2.0+	Vendor ID	2 BYTES	Varies	Vendor Identification Number
09	2.0+	Device ID	2 BYTES	Varies	Device Identification Number
0B	2.0+	SubVendor ID	2 BYTES	Varies	Sub system ID and Sub Vendor ID Differentiate specific model
0D	2.0+	SubDevice ID	2 BYTES	Varies	Sub system ID and Sub Vendor ID Differentiate specific model
0F	2.0+	Link Speed	4 BYTES	Varies	Current Link Speed
13	2.0+	Link Width	4 BYTES	Varies	Current Link Speed

8.4 Node insertion state

Upon a compute node insertion action, an insertion notification shall be sent by whoever detects the event using the Intel® RSD IPMI Set Node Insertion State command. This command should only be sent after the BMC_READY# is asserted.

After the insertion event is received, the BMC should clear its internal SMBIOS table both in RAM and in flash filesystem. It requires a complete system boot in order for BMC to obtain the SMBIOS data from the BIOS.

