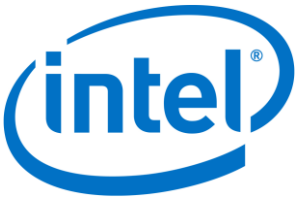

Intel® Rack Scale Design

BIOS & BMC Technical Guide

July 2016

Revision 002

Document Number: 332936-002

Contents

1 Introduction 5
1.1 Nomenclature used in this document 5
1.2 References 6
2 Intel® Rack Scale Design BMC IPMI Commands 7
3 Encoding Rules 19
3.1 Max length strings 19
3.2 Populating location info 19
4 Network configuration 20
4.1 Static mode 20
4.2 DHCP mode 20
5 BMC / CPP interaction 21
5.1 DHCP mode 21
6 BMC / BIOS iSCSI interaction 22
7 Sensors 23
7.1 Intel® Rack Scale Design NetFn 23
8 Node Discovery 25
8.1 Compute node information 25
8.2 BMC resident SMBIOS data storage 25
8.3 Node insertion state 25

Tables

Table 1 Nomenclature 5
Table 2 Reference documents 6
Table 2 Intel® Rack Scale Design BMC IPMI Command Table 7
Table 3 Sensors: Intel® Rack Scale Design NetFn 23

Revision History

Revision	Description	Date
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 detail in Table 1.

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
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

Term	Type	Description	Examples
TraySlotID	int	Tray Slot ID within Backplane. This are 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	Location
332868	Intel® Rack Scale Design GAMI API Specification	*http://intel.com/intelRSD*
332869	Intel® Rack Scale Design Pod Manager REST API Specification	*http://intel.com/intelRSD*
332870	Intel® Rack Scale Design Pod Manager Release Notes	*http://intel.com/intelRSD*
332871	Intel® Rack Scale Design Pod Manager User Guide	*http://intel.com/intelRSD*
332873	Intel® Rack Scale Design PSME REST API Specification	*http://intel.com/intelRSD*
332872	Intel® Rack Scale Design PSME Release Notes	*http://intel.com/intelRSD*
332874	Intel® Rack Scale Design PSME User Guide	*http://intel.com/intelRSD*
332877	Intel® Rack Scale Design RMM REST API Specification	*http://intel.com/intelRSD*
332876	Intel® Rack Scale Design RMM Release Notes	*http://intel.com/intelRSD*
332875	Intel® Rack Scale Design RMM User Guide	*http://intel.com/intelRSD*
332878	Intel® Rack Scale Design Storage Services API Specification	*http://intel.com/intelRSD*
332936	Intel® Rack Scale Design BIOS/BMC Tech Guide	*http://intel.com/intelRSD*
332937	Intel® Rack Scale Design Architectural Requirements Specification	*http://intel.com/intelRSD*
334611	Intel® Rack Scale Design Getting Started Guide	*http://intel.com/intelRSD*
n/a	Scalable Platforms Management API	*http://dmtf.org/standards/redfish*

§

2 Intel® Rack Scale Design 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.

Note: All Get commands require Privileged User access.

Note: All Set commands require Privileged Admin access.

Table 3 Intel® Rack Scale Design BMC IPMI Command Table

<table>
<tr><th colspan="4">Net Function = Intel General Application (“Intel® Rack Scale Design NetFn” Sensor Value), LUN = 00</th></tr>
<tr><th>Code</th><th>Command</th><th>Request, Response Data</th><th>Description</th></tr>
<tr><td>01h</td><td>Get Fan PWM</td><td>Request:
None

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</td><td>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® Rack Scale Design upper level FSC control.</td></tr>
<tr><td>02h</td><td>Set ISCSI Field</td><td>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.

Request:
byte 1 – Type of Field
00h = Default (legacy – ignored)
byte 2 – Instance of Field.

<table>
<tr><th>Fld</th><th>Name</th><th>Type</th><th>Description</th></tr>
<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>
</table>
</td><td></td></tr>
</table>

<table>
<tr><th colspan="5">Net Function = Intel General Application (“Intel® Rack Scale Design NetFn” Sensor Value), LUN = 00</th></tr>
<tr><th>Code</th><th>Command</th><th colspan="2">Request, Response Data</th><th>Description</th></tr>
<tr><td></td><td></td><td colspan="2">
<table>
<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(0-255)</td></tr>
<tr><td>15</td><td>Auth Method</td><td>byte</td><td>1 = None, 2 = CHAP, 3 = MultiCHAP</td></tr>
<tr><td>16</td><td>CHAP Username</td><td>string</td><td>Chap Username (max 32 characters)</td></tr>
<tr><td>17</td><td>CHAP Secret</td><td>string</td><td>Must be 12 to 16 characters long</td></tr>
<tr><td>18</td><td>Mutual CHAP Secret</td><td>string</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
<table>
<tr><th colspan="2">iSCSI Parameters Status Flag</th></tr>
<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>
</table></td></tr>
<tr><td>20</td><td>NIC</td><td>byte</td><td>NIC number to boot from (default 0)</td></tr>
</table>
byte 3 – Number of bytes to Set (ignored for non-string fields)
byte 4:N – data bytes to Set into Field

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
byte 2 – Maximum number of data bytes possible for Field.</td><td></td></tr>
<tr><td>03h</td><td>Get ISCSI Field</td><td colspan="2">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.

Request:
byte 1 – Instance of Field.
<table>
<tr><th>Fld</th><th>Name</th><th>Type</th><th>Description</th></tr>
<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>
</table></td><td></td></tr>
</table>

<table>
<tr><th colspan="4">Net Function = Intel General Application (“Intel® Rack Scale Design NetFn” Sensor Value), LUN = 00</th></tr>
<tr><th>Code</th><th>Command</th><th>Request, Response Data</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<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>1 = None, 2 = CHAP, 3 = 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>
</table>

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
</td><td></td></tr>
<tr><td>04h</td><td>Set Structure Block</td><td>Command removed from BMC.</td><td>Deprecated command—supplanted with Set iSCSI Field Command.</td></tr>
<tr><td>05h</td><td>Get Structure Block</td><td>Command removed from BMC.</td><td>Deprecated command—supplanted with Get iSCSI Field Command.</td></tr>
</table>

Net Function = Intel General Application ("Intel® Rack Scale Design NetFn" Sensor Value), LUN = 00			
Code	**Command**	**Request, Response Data**	**Description**
06h	Set ID Field	**Request**: byte 1 – Type of Field 00h = Default byte 2 – Instance of Field. 00h = Reserved 01h = ComponentType (reserved/RO) 02h = GUID (for programming idrom) 03h = GeoTag 04h = PodDCUID 05h = RackPUID 06h = RackPUName 07h = RackDCUID (derived on Rack) 08h = RackBPID (for Tray Only/RO) 09h = TrayRUID (derived on Tray) 0Ah = TrayPUID (derived on Tray) 0Bh = ModuleTUID (GPIO slot id, 1-based/RO) 0Ch = ModulePUID (derived on Module/RO) 0Dh = NodeMUID (uServer only 1-based/RO) 0Eh = NodePUID (uServer only derived/RO) 0Fh = ClientID (See section 3.2/RO) byte 3 – Number of bytes to Set (ignored for non-string fields) byte 4:N – data bytes to Set into Field **Response**: 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 byte 2 – Maximum number of data bytes possible for Field.	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.
07h	Get ID Field	**Request:** byte 1 – Instance of Field. 00h = Reserved 01h = ComponentType 1=Pod 2=rack 3=tray 4=module 5=sub-node 6=uServer (Multi-module) 02h = GUID 03h = GeoTag 04h = PodDCUID 05h = RackPUID 06h = RackPUName 07h = RackDCUID (derived on Rack)	

Net Function = Intel General Application ("Intel® Rack Scale Design NetFn" Sensor Value), LUN = 00			
Code	**Command**	**Request, Response Data**	**Description**
		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 = Playload is powered off 01h = Below normal operating temperature. 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 = Critial state.	This command returns the current baseboard thermal state described in the "Intel® Rack Scale Design 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 ("Intel® Rack Scale Design NetFn" Sensor Value), LUN = 00			
Code	**Command**	**Request, Response Data**	**Description**
0Ah	Get Node Info	**Request**: None **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	This command returns a byte map indicating the presence on the nodes. uServer only (not implemented on server as it is not relevant).

Net Function = Intel General Application ("Intel® Rack Scale Design NetFn" Sensor Value), LUN = 00			
Code	**Command**	**Request, Response Data**	**Description**
0Bh	Set Cooling Mode	**Request:** byte 1 – Cooling Mode [0] 0 = Tray Fans, 1 = Rack Fans. [1..7] Reserved **Response:** byte 1 – Completion code 00h = Normal 80h = Invalid context (e.g., sent to uServer Node BMC and not baseboard)	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	**Request:** None **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	
0Dh	Set Config Complete	**Request:** None **Response:** byte 1 – Completion code 00h = Normal 80h = Invalid context (e.g., sent to uServer Node BMC and not baseboard)	
0Eh	Set Config PCIe Retimer	**Request:** None **Response:** byte 1 – Completion code 00h = Normal 80h = Invalid context (e.g., sent to uServer Node BMC and not baseboard)	BMC debug command for internal use only.
0Fh	Get uBMC Versions	**Request:** None **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	uBMC on uServer node.
10h	Reset Node Bridge-IC	**Request:** byte 1:3 – Vendor specific magic value (Refer to vendor for magic value)	Reset Node's Bridge-IC (used by firmware update)

Net Function = Intel General Application ("Intel® Rack Scale Design NetFn" Sensor Value), LUN = 00			
Code	**Command**	**Request, Response Data**	**Description**
		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)	
11h	Set Node MAC Address	**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 **Response:** byte 1 – Completion code 00h = Normal C7h = Invalid length C9h = Invalid parameter D3h = Node not present	Set compute node MAC address.
12h	Get Node MAC Address	**Request:** byte 1 – 0-based network interface index **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	Get compute node MAC address.
13h	Set Node Insertion State	**Request:** byte 1 – Node insertion state 01h = Node insertion occurred All other values are reserved. **Response:** byte 1 – Completion code 00h = Normal C7h = Invalid length C9h = Invalid parameter	Set node insertion state. This command is called when this node insertion was detected by its parent.
20h	Get MDR Data Region Status	**Request:** byte 1 – Data Region 01h = SMBIOS table All other values are reserved. **Response:** byte 1 – Completion code	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® Rack Scale Design.

Net Function = Intel General Application ("Intel® Rack Scale Design NetFn" Sensor Value), LUN = 00			
Code	**Command**	**Request, Response Data**	**Description**
		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	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 determined that the data needs to be refreshed, use the following read / write / lock commands to perform the update.
21h	Set MDR Region Update Complete	**Request**: byte 1 – Session Lock Handle byte 2 – Data Region. 01h = SMBIOS region All other values are reserved. **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.	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. Session Lock Handle is returned by the "Get MDR Region Lock" command.
22h	MDR Region Read	**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. **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	This command is utilized by external applications to retrieve the specified data region. Length of 0 is invalid (returns C9h) Data is not returned if region is marked as invalid (CBh)

Net Function = Intel General Application ("Intel® Rack Scale Design NetFn" Sensor Value), LUN = 00			
Code	**Command**	**Request, Response Data**	**Description**
		byte 4:N – Data	
23h	MDR Region Write	**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. **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 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)	Session Lock Handle is returned by the "Get MDR Region Lock" command. Data length of 0 is invalid (will return C9h). Limitations of the IPMI stack may limit the maximum bytes possible to write (e.g., RMCPv1.0 limits to only ~250 bytes) The length of the region used is automatically increased (provided the write does not exceed the maximum region size) upon successful writes. 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	This command locks the specified data region. Only the BIOS should use the strict lock. All other external applications should use the pre-emptible 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.

Net Function = Intel General Application ("Intel® Rack Scale Design NetFn" Sensor Value), LUN = 00			
Code	**Command**	**Request, Response Data**	**Description**
		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).	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 02h = Host to FP and Midplane on server, Host Node 3 on uServer 03h = N/A on server, BMC on uServer byte 2 – Serial Debug On **Note: Only effects Selector 02h on Server.** 00h = Debug connected to Midplane 01h = Debug connected to Debug Card FFh = N/A for uServer **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)	BMC debug command for internal use only.
7Ch	Get Serial Debug Selector	**Request:** None **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 FP and Midplane on server, Host Node 3 on uServer 03h = N/A on server, BMC on uServer byte 3 – Serial Debug On **Note: Only effects Selector 02h on Server.** 00h = Debug connected to Midplane 01h = Debug connected to Debug Card FFh = N/A for uServer	BMC debug command for internal use only.
7Dh	Set Debug Level and Flags	**Request:** byte 1 – Debug Flags byte 2 – Debug Level. **Response:** byte 1 – Completion code 00h = Normal	BMC debug command for internal use only.

Net Function = Intel General Application ("Intel® Rack Scale Design NetFn" Sensor Value), LUN = 00			
Code	**Command**	**Request, Response Data**	**Description**
		byte 2 – Debug Flags byte 3 – Debug Level	
7Eh	Get Debug Level and Flags	**Request:** byte 1 – (optional) extended debug level flags request **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. FFh = Most amount of debug output.	BMC debug command for internal use only.
7Fh	Debug of fan PWM and Thermal State	**Request:** byte 1 – Forced Thermal State byte 2 – Forced Fan PWM MAX value that will be returned **Response:** byte 1 – Completion code 00h = Normal byte 2 – Forced Thermal State byte 3 – Forced PWM MAX value that will be returned	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. 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. Both parameters can take on ANY value from 0 to 255 (0xFF).

§

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® Rack Scale Design NetFn

The Intel® Rack Scale Design 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 Intel® Rack Scale Design NetFn sensor is not found, then use NetFn 0x38.

Table 4 Sensors: Intel® Rack Scale Design NetFn

Sensor Name	Sensor #	Entity ID	Instance	Sensor Type
Intel® Rack Scale Design 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"
```

§

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. This SMBIOS data can be retrieved using Intel® Rack Scale Design 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 no match, the BIOS sends a copy of the SMBIOS table to the BMC. If match, the BMC already has the SMBIOS data that matches from the BIOS. No SMBIOS data transfer is needed.

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

8.3 Node insertion state

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

It requires a complete system boot in order for BMC to obtain the SMBIOS data from the BIOS.

§