Hewlett Packard Enterprise

HPE iLO IPMI User Guide

Abstract

This document provides customers with information on the implementation of the Intelligent Platform Management Interface in HPE iLO, including the available commands.

Part Number: 30-585D730D-002
Published: October 2021
Edition: 1

Notices

The information contained herein is subject to change without notice. The only warranties for Hewlett Packard Enterprise products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.

Confidential computer software. Valid license from Hewlett Packard Enterprise required for possession, use, or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

Links to third-party websites take you outside the Hewlett Packard Enterprise website. Hewlett Packard Enterprise has no control over and is not responsible for information outside the Hewlett Packard Enterprise website.

Acknowledgments

Windows® is a registered trademark of Microsoft Corporation in the United States and/or other countries.

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

Intel® and Itanium® are trademarks of Intel Corporation in the U.S. and other countries.

All third-party marks are property of their respective owners.

Revision history

Part number	Publication date	Edition	Summary of changes
30-585D730D-002	October 2021	1	Update for IPMI **instrusion sensor**
30-585D730D-001	April 2021	1	• Updated for iLO 5 v2.42 • Liquid cooling module support
882431-006	November 2020	1	• Updated for iLO 5 v2.31. • Added support for the following power supply sensors: Input Power, Input Voltage, Output Voltage, Input Current, and Output Current.

Contents

Introduction and key concepts

Intelligent Platform Management (IPMI)

The term Intelligent Platform Management (IPMI), refers to autonomous monitoring and recovery features implemented directly in platform management hardware and firmware. The key characteristics of IPMI are available independently of the main processors, BIOS, and operating system. These characteristics include:

- Inventory
- Monitoring
- Logging
- Recovery control

Platform management functions are available even when the system is in a powered down state.

IPMI capabilities are a key component in providing enterprise-class management for HA systems. Platform status information is obtained and recovery actions initiated in situations where system management software and normal in-band management mechanisms are unavailable.

The independent monitoring, logging, and access functions available through IPMI provide a level of manageability built-in to the platform hardware. This manageability supports systems with no system management software available for the particular operating system, or the end user who elects not to load or enable the system management software.

Key concepts

- **Sensor Data Model**
- **FRU**
- **Standardized timers**
- **iLO security modes**
- **Interoperability modes**

Sensor Data Model

The IPMI Sensor Model provides access to monitored information including:

- Temperatures
- Power supplies
- Fan status
- Power
- CPU utilization

Instead of providing direct access to the monitoring hardware, IPMI provides access by abstracted sensor commands, such as the `Get Sensor Reading` command, implemented via a management controller. This approach isolates software

from changes in the platform management hardware implementation. Sensors return analog or discrete readings and events are either discrete or threshold-based. Sensors are classified according to:

- Type of reading/event
- Type of sensor
- Type of entity

Event types, sensor types, and monitored entities are represented using numeric codes defined in the IPMI specification. IPMI avoids reliance on strings for management information and using numeric codes facilitates:

- Internalization
- Automated handling by higher-level software
- Reduces management controller code and data space requirements

Sensor owner identification

A particular Sensor Data Record (SDR) within the Sensor Data Record Repository (SDRR) and a particular event within the Sensor Event Log (SEL) must contain information to identify the sensor owner. For management controllers, a slave address, LUN, and channel number identify the sensor owner. For system software, a software ID identifies the sensor owner. These fields are used in event messages. Events from management controllers are identified by an 8-bit field where the upper 7 bits represent the slave address or the system software ID. The least significant bit is `0` if the value represents a slave address and `1` if the value represents a system software ID.

The sensor number is not part of the sensor Owner ID, but is a separate field used to identify a sensor associated with the sensor owner. This combination of sensor owner ID and sensor number uniquely identifies a sensor in the system.

Table 1: Sensor owner ID and sensor number field definition

IPMB Sensor Owner ID	System Sensor Owner ID
7:1 slave address (7 bits)	system software ID (7 bits)
0 0b (ID is a slave address)	0 1b (ID is a software ID)
LUN (2 bits)	sensor number (8 bits, FFh = reserved)
channel number (4 bits)	
sensor number (1 bit, FFh = reserved)	

Sensor type code

Each sensor has a sensor type code and are defined in the following table. Sensor type codes are used in SDRs and event messages. An example of a sensor type code is code 0x1, which indicates a temperature sensor.

Table 2: Some HPE iLO sensor type codes

Sensor type	Sensor type code	Reading type code
Temperature	0x1	0x1
Fan	0x4	0xA
Fan redundancy	0x4	0xB
Health LED	0x18	0x07
UID LED	0xC0	0x70
Power supply	0x8	0x6F
Power supply redundancy	0x8	0xB

For a complete listing of sensor type codes, see the **IPMI specification**.

Sensor number and naming conventions

To facilitate customer scripts, iLO has allowed for consistent sensor numbering of most sensors. Additionally, iLO supports two types of naming conventions:

- Legacy—This mode uses the standard ProLiant naming convention. For temperature sensors, the format is `<Sensor Number> <Description>`.
- Programmatic—This mode uses `SensorType_ReadingUnits_Instance`.

The naming convention mode can be changed on iLO by using an OEM `ipmi` command.

More information

Standard sensor list

System event log and event messages

The BMC provides a centralized, nonvolatile SEL. Having the SEL and logging functions managed by the BMC helps ensure that post-mortem logging information is available if a failure disables the system processors.

A set of IPMI commands allows the SEL to be read and cleared, and for events to be added to the SEL. The common request message used for adding events to the SEL is an event message. Event messages are sent to the BMC through the system interface by system software or the IPMB by satellite controllers that detect events and log them to the SEL. The controller that generates an event message to another controller through IPMB is the IPMB Event Generator. The controller that receives event messages is the IPMB Event Receiver.

The BMC can be considered an IPMB event generator where it sends itself events on its own internal (virtual) IPMB.

Event messages are special messages sent by management controllers when they detect significant or critical system management events. Some examples include messages for events, such as:

- temperature threshold exceeded
- fan failure
- power fault

The event message generator notifies the system by sending an Event Request Message to the event receiver device.

When the event receiver gets a valid event message, it sends a response message to the event message generator and transfers the event to the SEL. The event receiver does not interpret event messages so that new event message types can be added into the system without impacting event receiver implementation.

iLO 5 2.30 and later supports 512 SEL entries.

NOTE: Earlier versions of iLO 5 support 256 SEL entries, and iLO 4 supports 64 SEL entries.

iLO supports two modes of SEL operation. The modes of operation can be switched by using an OEM IPMI command. The default mode is auto-rollover.

- Auto-Rollover—Allows the oldest event entry to be flushed to make room for a new entry. This mode is referred to in the DCMI specification.
- IPMI Compliant—Drops the newest events in favor of keeping the already retained events when the SEL is full.

Table 3: SEL event records

Byte	Field	Description
1 2	Record ID	ID used for SEL record access. The Record ID values 0000h and FFFFh have special meaning in the Event Access commands and must not be used as Record ID values for stored SEL event records.
3	Record type	[7:0]—Record type 02h = system event record C0h-DFh = OEM timestamped, bytes 8–16 OEM defined E0h-FFh = OEM nontimestamped, bytes 4–16 OEM defined
4 5 6 7	Timestamp	Time when event was logged. LS byte first.

Table Continued

Byte	Field	Description
8 9	Generator ID	RqSA and LUN if event was generated from IPMB. Software ID if event was generated from system software. **Byte 1** [7:1]—7-bit I^2C. Slave address, or 7–bit system software ID [0] • 0b = ID is IPMB slave address • 1b = System software ID **Byte 2** [7:4]—Channel number. Channel that received the event message 0h if the event message was received through the system interface, primary IPMB, or internally generated by the BMC. [3.2]—Reserved. Write as 00b. [1.0]—IPMB device LUN if byte 1 holds slave address, otherwise 00b.
10	EvM Rev	Event message-format version (=04h for events in the IPMI v2.0 specification, 03h for IPMI v1.0 event messages).[1]
11	Sensor type	Sensor Type Code for sensor that generated the event.
12	Sensor #	Number of sensors that generated the event.
13	Event dir 1 Event type	**Event dir** [7] • 0b = Assertion event • 1b = Deassertion event **Event type** Type of trigger for the event, such as, a critical threshold going high or state asserted. This parameter also indicates class of the event. Example: discrete, threshold, or OEM. The event type field is encoded using the event/reading type code.
14	Event data 1	Event request message, event data field contents.
15	Event data 2	Event request message, event data field contents.
16	Event data 3	Event request message, event data field contents.

[1] The BMC must accept platform event request messages that are in IPMI v1.0 format (EvM Rev=03h) and log them as IPMI v1.5/v2.0 records by setting the EVMRev field to 04h and setting the channel number in the Generator ID field appropriately for the channel that received the event.

SDR repository

With the extensibility and scalability of IPMI, each platform implementation can have a different population of management controllers and sensors, and different event generation capabilities. IPMI allows system management

software to retrieve information from the platform and automatically configure itself to the capabilities of the platform, enabling the use of plug and play, platform-independent instrumentation software.

Information that describes the platform management capabilities is provided by two mechanisms:

- Capability commands—Commands within the IPMI command set that return information on other commands and functions that the controller can handle.
- SDRs—Contain information about the type and number of sensors in the platform, sensor threshold support, event generation capabilities, and sensor type readings.

The primary purpose of SDRs is to describe the sensor configuration of the platform management subsystem to system software. SDRs also include records describing the number and type of devices connected to the IPMB of the system and records that describe the location and type of FRU Devices (devices that contain field replaceable unit information).

SDR formats

The general SDR format consists of three major components:

- Record header
- Record key fields
- Record body

To save space, sensors that only generate events do not require SDRs, in addition, generic system management software does not access sensors unless they are reported by SDRs.

Table 4: Sensor data record formats

Record header	Record Key fields	Record body
Record ID—Used for accessing sensor data records.	The record key bytes are the contiguous bytes following the record header. The number of bytes vary according to record type. Together, they make up a set of unique fields for a given record specifying location (for example, slave address, LUN and Bus ID) and sensor number.	Contains specific information to the sensor data record.
SDR version—Version number of the SDR specification.		
Record type—Number representing the type of record. For example, 01h = 8–bit sensor with thresholds.		
Record length—Number of bytes of data following the record length field.		

Reading the SDR repository

An application that retrieves records from the SDR repository must first read them sequentially using the **Get SDR command** command. This command returns the requested record and the record ID of the next SDR in the sequence.

NOTE:

Record IDs are not required to be sequential or consecutive and applications should not assume that SDR record IDs follow any particular numeric ordering.

Retrieve succeeding records by issuing the **Get SDR command** using the `next record` ID returned in the previous response. This is continued until the `End of Record` ID is encountered.

After all the desired records have been read, the application can randomly access the records according to their Record ID. An application that seeks to access records randomly must save a data structure that retains the record key information according to the Record ID.

IMPORTANT:

Record IDs can change with time; it is important for applications to first verify that the Record Key information matches the record retrieved.

If the Record ID is no longer valid for a Record Key, then, access the SDR records again as, by issuing **Get SDR command** until the record matches the Record ID.

An application can tell if records have changed by examining the most recent addition timestamp using the **Get SDR repository info command**.

If the record information has changed, an application does not need to list out the entire contents of all records. The **Get SDR command** allows a partial read of the SDR; an application can search for a given Record Key by just retrieving that portion of the record.

FRU

The IPMI specifications include support for storing and accessing multiple sets of nonvolatile FRU data for different modules in the system. An enterprise-class system typically has FRU information for each major system board such as:

- System Baseboard
- Adapters
- Storage Controllers
- Memory modules

FRU data includes:

- Serial number
- Part number
- Model
- Asset tag

IPMI FRU information is accessible through the IPMB and management controllers. The information can be retrieved at any time, independent of the main processor, BIOS, system software, or OS, through out-of-band interfaces, such as the LAN. FRU information is still available when the system is powered down.

With these capabilities FRU information is available under failure conditions when access mechanisms that rely on the main processor are unavailable. This facilitates the creation of automated remote inventory and service applications. IPMI does not seek to replace other FRU or inventory data mechanisms such as those provided by SM BIOS and PCI vital product data. Rather, IPMI FRU information is used to complement that information or provide information access out-of-band or under system down conditions.

Standardized timers

Watchdog timer

IPMI provides a standardized interface for a system watchdog timer that can also be used for BIOS, operating system, and OEM applications. The timer can be configured to automatically generate selected actions when it expires; including power off, power cycle, reset, and interrupt. The timer function automatically logs the expiration event. Setting `0` for the timeout interval result causes the timeout action to be initiated immediately. This provides a means for devices on the IPMB, such as remote management cards, to use the watchdog timer to initiate emergency reset and other recovery actions dependent on the capability of the timer.

POH counter

The optional power-on hours (POH) counter is supported and returns a counter value proportional to the system operating power-on hours.

Timestamp format

A timestamp is a key component of event logging and tracking changes to the SDRs and the SDRR.

Time is an unsigned, 32–bit value representing the local time as the number of seconds from `00:00:00,January 1, 1970`.

The timestamps used for SDR and SEL records are specified in relative local time (for example, the difference between the timestamp does not include the GMT offset). Converting the timestamp to a GMT-based time requires adding the GMT offset for the system and is obtained from system software level interfaces. IPMI commands do not store or return GMT offset for the system. Applications can use ANSI C time standard library routines for converting the SEL timestamp into other time formats.

Special timestamp values

- `0xFFFFFFFF`—Indicates an invalid or unspecified time value.
- `0x00000000` through `0x20000000`—Indicates events that occur after initialization of the SEL device up to when the timestamp is set with the system time value. These timestamp values are relative to the completion of the SEL devices initialization, and not to January 1, 1970.

iLO security modes

- **Production**

  The default state of the server when received by the customer. The network interface is secure, but the host interface is open. This allows iLO to be compatible with existing Gen 9 software. In this mode, the IPMI over LAN UDP port is disabled by default due to known security issues inherent to 2.0 sessions. The IPMI over LAN port can be enabled through IPMI, Redfish, or Web GUI if needed.

- **High security**

  In the high security state, iLO is secure on the network and on the host interface. Encryption and authentication on the host interface differentiates this mode from production mode. FIPS-level cryptography is required on the network interface. The high security state is intended for customers who require high security, but do not need the restrictions imposed by FIPS and the Suite B modes. In this mode, the IPMI over LAN UDP port is disabled by default due to known security issues inherent to 2.0 sessions. Additionally, IPMI over the system interface is limited to user level commands and a few additional operator level commands required by OS IPMI infrastructures.

- **FIPS**

  This security mode is intended to meet the security requirements outlined FIPS 140-2 level 1 and Common Criteria. When in FIPS Validated mode, interfaces that do not meet FIPS requirements are shut off or restricted. These include SNMP v1 and IPMI. iLO is required to be in FIPS Validated mode when operating in accordance with its Common Criteria certificate. In this mode, the IPMI over LAN UDP port is disabled by default due to known security issues

inherent to 2.0 sessions. Additionally, IPMI over the system interface is limited to user level commands and a few additional operator level commands required by OS IPMI infrastructures.

- **CNSA**

  This mode is intended to meet the requirements of FIPS 140-2 level 1, Common Criteria, and CNSA (ex-NSA Suite B for Secret and Top Secret installations). In all other respects it is the same as FIPS mode.

Interoperability modes

- HPE Open

  Allows improved interoperability with IPMI open source tools. HPE open mode specifically utilizes the new HPE IANA. This enables customers to easily identify the open mode products and allow open source tools to automatically avoid the extensions done for legacy HPE products. This is the default mode for iLO.

  - Utilizes HPE IANA.
  - Utilizes compact sensor records for discrete sensors.
  - Combined sensors (Discretes with additional analog readings) are disabled. Separate sensors for analog readings and discrete readings.
  - Fan sensors use reading type of 0x07 Severity.
  - Disabled OEM health sensor in favor of utilizing standard health with reading type of 0x07 Severity.

- Legacy

  Provides compatibility for customer scripts written for previous generations. Legacy mode specifically utilizes the HP IANA to remain compatible with existing scripts and open source tool extensions.

  - Utilizes HP IANA.
  - Utilizes full sensor records for discrete sensors.
  - Combined sensors (Discretes with additional analog readings) are enabled.
  - Fan sensors use reading type of 0x0A Availability.
  - OEM health sensor enabled.

- HPE Open and Legacy modes can be switched by utilizing the following scripts or through utilizing the individual IPMI commands listed later in the manual.

```
Usage: hpe-open-mode.sh [-5|--ipmi1.5-enable][-e|--udp-lan-enable]
                        [-n|--no-ilo-reset][-s|--sensor-names-programmatic]
                        [-v|--verbose][-q|--quiet][-l <file>|--log=<file>]
                        [-h|-?|--help][-V|--version]
```

- The HPE Open mode allows for enablement of 1.5 as well as the 2.0 IPMI over LAN sessions. Sensor naming can be either legacy or programmatic in this mode.

```
Usage: hpe-legacy-mode.sh [-5|--ipmi1.5-enable][-d|--udp-lan-disable]
                          [-n|--no-ilo-reset][-v|--verbose][-q|--quiet]
                          [-l <file>|--log=<file>]
                          [-h|-?|--help] [-V|--version]
```

IPMI Topology

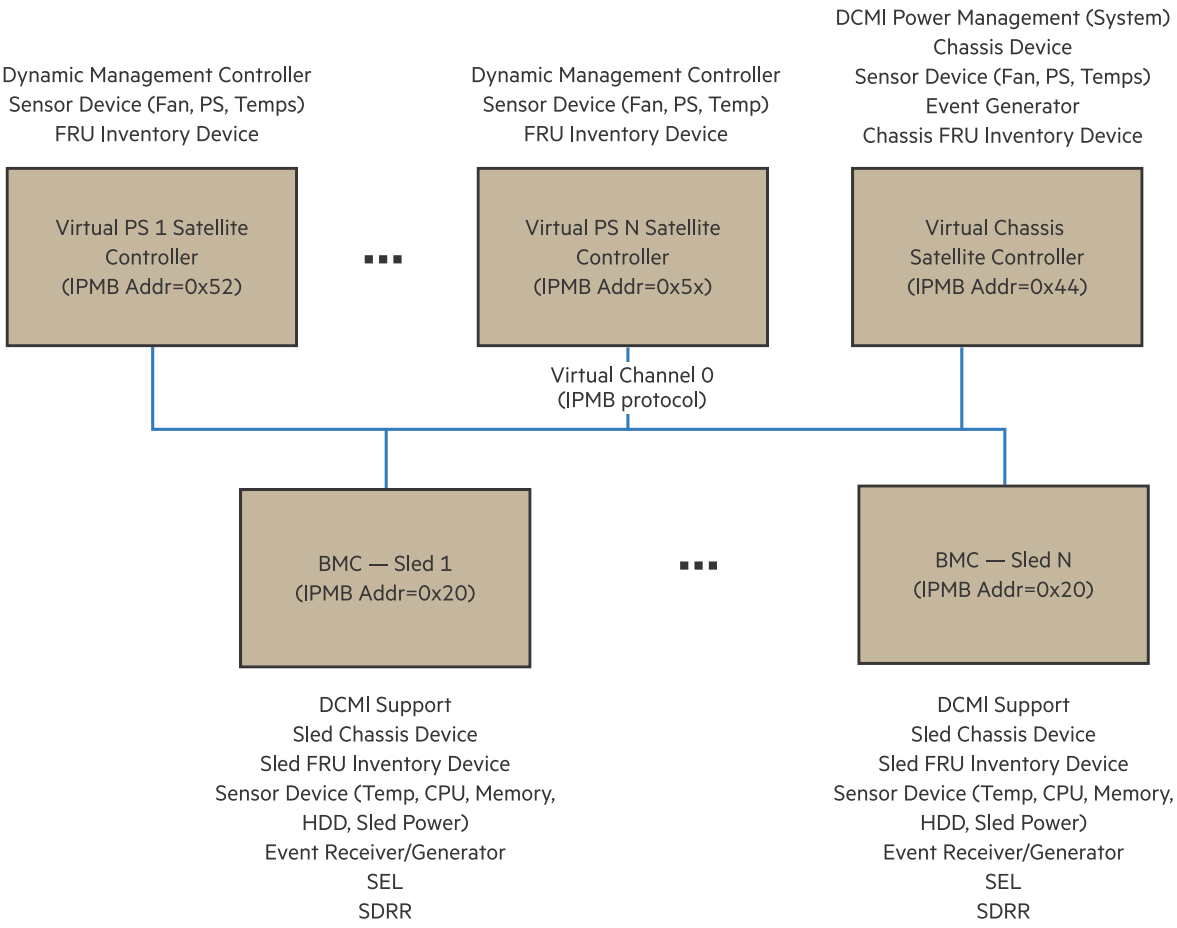


Figure 1: HPE iLO virtual IPMI topology

The following table summarizes the various functions available for each virtual management controller type.

Table 5: HPE iLO Virtual Management Controller functions

Function	Description	Applicable Virtual Management Controller		
		BMC	Chassis	Power Supply
IPM Device	The BMC must implement the mandatory IPM Device commands. If an IPMB is provided, the mandatory commands must be accessible from the IPMB unless otherwise noted.	X	X	X
System Interface	The implementation must provide MC access through one of the specified IPMI system interfaces.	X		

Table Continued

Function	Description	Applicable Virtual Management Controller		
		BMC	Chassis	Power Supply
SDRR	The BMC must provide an SDRR to hold Sensor, Device Locator, and Entity Association records for all sensors in the platform management subsystem. This does not need to include SDRs for sensors that only generate events. If the SDRR is writable, it is recommended that at least 20% additional space is provided for add-in platform management extensions. The SDRR must be accessible through the system interface. If an IPMB is provided, the SDRR must be readable through that interface as well. SDR update through the IPMB interface is optional. SDRR access when the system is powered up or in ACPI S1 sleep is mandatory. Access when the system is powered down or in a >S1 sleep state is optional.	X		
IPMB Interface	The IPMB is highly recommended, but optional. The BMC must provide the system interface to the IPMB. If an IPMB is implemented, at least one of the specified IPMB connectors must be provided. Refer to the IPMB Protocol specification for connector definition. In addition, the BMC must implement a message channel that allows messages to be sent from the IPMB to the system interface, and vice versa, and any other mandatory IPMB support functions and commands.	X	X	X
Watchdog Timer	The BMC must provide the standardized Watchdog Timer interface, with support for system reset action. Certain functions within the Watchdog Timer are optional. For more information, see **Watchdog timer**.	X		
Event Receiver	The BMC must implement an Event Receiver function and accept Event Messages through the system interface. If an IPMB is provided, the Event Receiver function must also accept Event Messages from the IPMB. Event Receiver operation while the system is powered up or in ACPI S1 sleep is mandatory. Operation when the system is powered down or in a >S1 sleep state is optional.	X		
SEL Interface	The BMC must provide a System Event Log interface. The event log must hold at least 16 entries. SEL access must be provided through the system interface. The SEL must be fully accessible through all mandatory SEL commands and all supported interfaces to the BMC whenever the system is powered up or in ACPI S1 sleep state. SEL read access is always mandatory whenever the BMC is accessible, and through any interface that is operational, regardless of system power state.	X		
FRU Inventory	The BMC must provide a logical Primary FRU inventory device, accessible through the `Write-` and `Read FRU Data` commands. The `FRU Inventory Device Info` command must also be supported. It is highly recommended that all other management controllers also provide a Primary FRU inventory device. (This was optional in IPMI 1.0.)	X	X	X

Table Continued

Function	Description	Applicable Virtual Management Controller		
		BMC	Chassis	Power Supply
Initialization Agent	The initialization agent function is one where the BMC initializes event generation and sensors both internally and on other management controllers according to initialization settings stored in the SDR for the sensor.	X	X	X
Sensors	The BMC can provide sensors. A typical server BMC would provide sensors for baseboard temperature, voltage, and chassis intrusion monitoring.	X	X	X
Internal Event Generation	The BMC must generate internal events for the Watchdog Timer. It is highly recommended that sensors generate events to eliminate the need for system management software to poll sensors, and to provide postmortem failure information in the SEL. Internal event generation for sensors is optional, but highly recommended, particularly for environmental sensors (for example, temperature and voltage).	X	X	X
External Event Generation	The BMC can be designed to accept the Set Event Receiver command to allow it to be set as an IPMB Event Generator and send its event messages to another management controller. This feature would primarily be used for development and test purposes.	X	X	X
LAN Messaging	Ability for the BMC to send and receive IPMI Messaging over LAN.	X		
LAN Alerting	Ability to send an Alert over the LAN.	X		
Bridging Support	The ability to transfer IPMI request and response messages between two interfaces connected to the BMC. The following support is required if the corresponding interfaces are supported: • LAN <–> IPMB • LAN <–> System Interface	X	X	X
Platform Event Filtering (PEF) and Alert Policies	Ability for BMC to perform a selectable action on an event. This capability is mandatory if paging or alerting is supported. Certain actions within PEF are optional. Refer to the sections on PEF for information. The Alert action and Alert Policies are mandatory if serial/modem or LAN alerting is supported.	X		
DCMI	Data Center Management Interface	X		
DCMI Power Management	Power Readings/Limits		X	

Discovering managed entities using IPMItool

Use the following procedure to query an SDRR from the BMC.

Procedure

1. Enter `sdr list all` to show all management controller records.

   Management controller records include:

   - Chassis controller—Statically assigned and should always be present. Even though always assigned address 0x44, the record should be parsed to learn the address and channel number (IPMB=0).
   - Power supply controllers—Full complement of records always present. Dynamic indication in record indicates to application that the controller may or may not be present. When not present, the controller will “NACK”. Even though always assigned address 0x52-0x58, records should be parsed to learn the address and channel number (IPMB=0).

IPMItool

IPMItool is a simple command-line interface to systems that support the IPMI 1.5/2.0 specifications. IPMItool provides the following:

- Ability to read the SDRR and print sensor values.
- Display the contents of the system event log.
- Print field replaceable unit information.
- Read and set LAN configuration parameters.
- Perform remote chassis power control.

IPMItool takes advantage of IPMI-over-LAN interfaces, but it can use the system interface as provided by a kernel device driver such as Open IPMI. IPMItool is available under a BSD license.

System Management Software is complex and makes platform management only part of a much larger management picture. However, many system administrators and developers rely on command-line tools that can be scripted. IPMItool takes a different approach to SMS and provides a command-line oriented tool. Therefore, it is not designed to replace the Open IPMI library. Where possible, it supports printing comma-separated values for output to facilitate parsing by other scripts or programs. It is designed to run quick command response functions that can be as simple as turning the system on or off or as complex as reading in the sensor data records and extracting and printing detailed sensor information for each record.

For example:

```
root@USER:/# ipmitool -I lanplus -H <iLO IP address> -U <username> -P <password> sdr
list mcloc
ChasMgmtCtlr1    | Static MC @ 44h   | ok
PsMgmtCtlr1      | Dynamic MC @ 52h  | ok
PsMgmtCtlr2      | Dynamic MC @ 54h  | ok
PsMgmtCtlr3      | Dynamic MC @ 56h  | ok
PsMgmtCtlr4      | Dynamic MC @ 58h  | ok
```

Out of band commands

BMC out of band command

All commands to the BMC are directed and o not require any bridging.

Single-bridging out of band command

You must single-bridge out of band commands to reach specific chassis management controller or power supply management controller to discover management controller addresses.

Enter the `sdr list all` command at the BMC to discover chassis and power supply management controller addresses. Once you have these addresses, they can be used to bridge to these additional controllers. For example:

- Chassis Controller

  `-b 0 -t 0x44`

- Power Supply

  Controller 1

  `-b 0 -t 0x52`

- Power Supply

Controller 2

```
-b 0 -t 0x54
```

Where:

- `-b <ipmi channel number>`
- `-t <target slave address>`

About interface types

IPMItool supports dynamic loading of interfaces that correspond to low-level communication methods for accessing IPMI systems. The most common of these methods are the System Interface provided by the OpenIPMI Linux kernel driver and IPMI over LAN interfaces.

System interface

There are multiple types of system interfaces, and they are all similar enough to enable a single driver like OpenIPMI to support them all. The varieties of system interfaces include KCS, BT, and SSIF. All of these interfaces are supported in recent versions of the OpenIPMI driver for the Linux kernel. IPMItool uses this driver to access the system interface through a character device node at `/dev/ipmi0`. To use this interface with IPMItool, provide the `-I open` parameter on the command line.

iLO supports KCS and a proprietary high-speed interface, CHIF. Hewlett Packard Enterprise supplies an OpenIPMI-based driver for CHIF so it can be used with IPMItool.

LANPlus interface

The LANPlus interface communicates with the BMC over an Ethernet LAN connection using UDP over IPv4 and IPv6. The LANPlus interface uses the RMCP+ protocol. RMCP+ facilitates:

- Improved authentication.
- Improved data integrity checks.
- Encryption.
- Ability to carry multiple types of payloads.

Generic Serial Over LAN support requires RMCP+, so the IPMItool `sol activate` command requires the use of LANPlus.

RMCP+ session establishment uses a symmetric challenge-response protocol called Remote Authenticated Key-Exchange Protocol (RAKP) which allows the negotiation of many options.

NOTE: IPMItool does not allow you to specify the value of every option, defaulting to the most obvious settings marked as required in the 2.0 specification. Authentication and integrity HMACS are produced with SHA1, and encryption is performed with AES-CBC-128. Role-level logins are not supported.

IPMItool must be linked with the OpenSSL library to perform the encryption functions and support the LANPlus interface. If the required packages are not found, it will not be compiled and supported. To link IPMItool with the OpenSSL Library, run the following command:

```
ipmitool -I lanplus -H <hostname> [-U <username>] [-P <password>] <command>
```

A host name must be given on the command line to use the LAN interface with IPMItool.

The `-C` option allows the authentication integrity and encryption algorithms to be used for LANPlus sessions based on the cipher suite ID found in IPMI 2.0. The default cipher suite is 3, which specifies the following algorithms:

- RAKP-HMAC-SHA1 authentication
- HMAC-SHA1–96 integrity
- AES-CBC-128 encryption

Raw Get Device ID to chassis satellite controller over LAN

```
# ipmitool -I lanplus -H <iLO IP Address> -u <username> -p <password> -L Administrator
-b 0 -t 0x44 raw 6 1

15 01 02 01 02 29 0b 00 00 00 85 00 00 00 00
```

Powering on a system over LAN

```
# ipmitool -I lanplus -H <iLO IP Address> -u <username> -p <password> -L Administrator
chassis power on

Chassis Power Control: Up/On
```

Activating SOL on a system over LAN

```
# ipmitool -I lanplus -H <iLO IP Address> -u <username> -p <password> -L Administrator
sol activate
```

Features

Instead of directly accessing the monitoring hardware for device entry, IPMI provides access to sensor data through abstracted messaging commands. Some common types of sensors that can be found in the system include baseboard and processor temperature sensors, processor and DIMM presence sensors, fan speed and failure monitoring, and baseboard, processor and SCSI terminating voltage sensors. The amount of data available for each sensor can be overwhelming. By default, IPMItool displays only the sensor name, reading, and status. Considerably more output can be seen by enabling the verbose output option.

To facilitate discovery of features, IPMI includes a set of records called SDRs kept in a single centralized nonvolatile storage area. These records include software information such as how many sensors are present, sensor type, sensor events, threshold information, and more. This feature allows software to interpret and present sensor data without prior knowledge about the platform.

Output from `sdr list all` command

```
root@USER:/# ipmitool -I lanplus -H <iLO IP Address> -u <username> -p <password>
sdr list all
UID Light         | 0x00              | ok
Health LED        | 0x00              | ok
01-Inlet Ambient  | 20 degrees C      | ok
02-CPU 1          | 40 degrees C      | ok
03-CPU 2          | 40 degrees C      | ok
04-DIMM P1 1-3    | disabled          | ns
05-DIMM P1 4-6    | 27 degrees C      | ok
06-DIMM P2 1-3    | disabled          | ns
07-DIMM P2 4-6    | 23 degrees C      | ok
08-HD Max         | 35 degrees C      | ok
09-Chipset        | 44 degrees C      | ok
10-VR P1          | 30 degrees C      | ok
.
.
```

```
.
PsMgmtCtlr3       | Dynamic MC @ 56h   | ok
PsMgmtCtlr4       | Dynamic MC @ 58h   | ok
```

See **Verbose output examples** for an example of verbose output from the `sdr list all` command.

Events

Events are messages sent by the management controller when system management events are detected. Some examples of events are temperature threshold exceeded, voltage threshold exceeded, and correctable ECC memory errors. Events are processed, and they are usually logged in the SEL. This is similar to the SDR because it provides a centralized nonvolatile storage area for platform events that are logged autonomously by the management controller or directly with event messages sent from the host.

There is an abundance of information available from an event log entry. By default, IPMItool displays only the basic data for an event and the sensor that triggered it. Detailed information is available with the verbose option.

Memory status event

iLO supports several types of memory errors. For example, `Correctable ECC logging limit reached`, `Uncorrectable ECC`, `Memory Device Disabled`, and `Configuration Error`. When the BIOS detects a memory-related error, it notifies iLO. iLO updates the corresponding memory status sensor and logs an SEL, as shown in the following example.

```
linux-e8j8:~ # ipmitool -H <iLO IP Address> -u <username> -p <password> -I lanplus sdr
get 'Mem_Stat_C01S01'

Sensor ID              : Mem_Stat_C01S01 (0x0)
 Entity ID             : 32.0 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Memory
                         [Presence Detected]
 Assertions Enabled    : Memory
                         [Correctable ECC logging limit reached]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

linux-e8j8:~ # ipmitool -H <iLO IP Address> -u <username> -p <password> -I lanplus
sel list

   2 | 06/23/2015 | 06:23:59 | Memory | Correctable ECC logging limit reached | Asserted
   3 | 06/23/2015 | 06:24:04 | Chassis #0xac | Transition to Non-critical from OK |
       Asserted
   4 | 06/23/2015 | 06:26:25 | Memory #0x48 | Uncorrectable ECC | Asserted
   5 | 06/23/2015 | 06:26:25 | Memory | Uncorrectable ECC | Asserted
   6 | 06/23/2015 | 06:27:18 | Memory | Memory Device Disabled | Asserted
   7 | 06/23/2015 | 06:27:44 | Memory | Configuration Error | Asserted
```

Correctable error detected

The following example shows the output when a Correctable ECC error is detected by iLO.

```
linux-e8j8:~ # ipmitool -H <iLO IP Address> -u <username> -p <password> -I lanplus
sdr get 'Mem_Stat_C01S01'

Sensor ID              : Mem_Stat_C01S01 (0x0)
 Entity ID             : 32.0 (Memory Device)
```

```
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Memory
                         [Correctable ECC logging limit reached]
                         [Presence Detected]
 Assertions Enabled    : Memory
                         [Correctable ECC logging limit reached]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

linux-e8j8:~ # ipmitool -H <iLO IP Address> -u <username> -p <password> -I lanplus
sel get 2

SEL Record ID          : 0002
 Record Type           : 02
 Timestamp             : 06/23/2015 06:23:59
 Generator ID          : 0120
 EvM Revision          : 04
 Sensor Type           : Memory
 Sensor Number         : 00
 Event Type            : Sensor-specific Discrete
 Event Direction       : Assertion Event
 Event Data (RAW)      : 00ffff
 Event Interpretation  : Missing
 Description           : Correctable ECC logging limit reached

Sensor ID              : Mem_Stat_C01S01 (0x0)
 Entity ID             : 32.0
 Sensor Type (Discrete): Memory
 States Asserted       : Memory
                         [Correctable ECC logging limit reached]
                         [Presence Detected]
```

CPU status event

When the BIOS detects a processor-related error, it notifies iLO. iLO updates the corresponding CPU status sensor and logs an SEL, as shown in the following example. The supported CPU errors are `IERR`, `Configuration Error`, and `Uncorrectable machine check exception`.

```
linux-e8j8:~ # ipmitool -H <iLO IP Address> -u <username> -p <password> -I lanplus
sdr get 'CPU_Stat_C1'

Sensor ID              : CPU_Stat_C1 (0x12)
 Entity ID             : 3.1 (Processor)
 Sensor Type (Discrete): Processor (0x07)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Processor
                         [Presence detected]
 Assertions Enabled    : Processor
                         [IERR]
                         [Configuration Error]
                         [Uncorrectable machine check exception]
 OEM                   : 0

linux-e8j8:~ # ipmitool -H <iLO IP Address> -u <username> -p <password>
-I lanplus sel list
```

```
 8 | 06/23/2015 | 07:14:50 | Processor #0x12 | IERR | Asserted
 9 | 06/23/2015 | 07:16:17 | Processor #0x12 | Uncorrectable machine check
    exception | Asserted
 a | 06/23/2015 | 07:16:51 | Processor #0x12 | Configuration Error | Asserted
```

IERR detected

The following example shows the output when an IERR state is detected by iLO.

```
linux-e8j8:~ # ipmitool -H <iLO IP Address> -u <username> -p <password> -I lanplus
sdr get 'CPU_Stat_C1'

Sensor ID              : CPU_Stat_C1 (0x12)
 Entity ID             : 3.1 (Processor)
 Sensor Type (Discrete): Processor (0x07)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Processor
                         [IERR]
                         [Presence detected]
 Assertions Enabled    : Processor
                         [IERR]
                         [Configuration Error]
                         [Uncorrectable machine check exception]
 OEM                   : 0

linux-e8j8:~ # ipmitool -H <iLO IP Address> -u <username> -p <password>-I lanplus
sel get 8

SEL Record ID          : 0008
 Record Type           : 02
 Timestamp             : 06/23/2015 07:14:50
 Generator ID          : 0020
 EvM Revision          : 04
 Sensor Type           : Processor
 Sensor Number         : 12
 Event Type            : Sensor-specific Discrete
 Event Direction       : Assertion Event
 Event Data (RAW)      : 00ffff
 Event Interpretation  : Missing
 Description           : IERR

Sensor ID              : CPU_Stat_C1 (0x12)
 Entity ID             : 3.1
 Sensor Type (Discrete): Processor
 States Asserted       : Processor
                         [IERR]
                         [Presence detected]

FRU Device Description : CPU 1 (ID 16)
 Product Manufacturer  : Advanced Micro Devices, Inc.
 Product Name          : AMD EPYC 7302P 16-Core Processor
```

PCIe and OS NMI

The PCIe and OS NMI sensors are event-only sensors. When the BIOS detects a PCIe-related error, it notifies iLO. iLO logs an SEL, as shown in the following example.

If a user initiates an NMI from the iLO web interface, iLO logs an SEL, as shown in the following example. The BIOS logs an event in the Integrated Management Log.

```
linux-e8j8:~ # ipmitool -H <iLO IP Address> -u <username> -p <password>
-I lanplus sdr elist event

PCI-E Bus Error  | C4h | ns  | 49.0 | Event-Only
OS_NMI           | C5h | ns  | 35.0 | Event-Only

linux-e8j8:~ # ipmitool -H <iLO IP Address> -u <username> -p <password>  -I lanplus
sel list
   b | 06/23/2015 | 07:26:19 | Critical Interrupt #0xc4 | Bus Uncorrectable error |
       Asserted
   c | 06/23/2015 | 07:29:12 | Critical Interrupt #0xc5 | NMI/Diag Interrupt |
       Asserted

linux-e8j8:~ # ipmitool -H <iLO IP Address> -u <username> -p <password> -I lanplus
sel get 0x0b

SEL Record ID          : 000b
 Record Type           : 02
 Timestamp             : 06/23/2015 07:26:19
 Generator ID          : 0001
 EvM Revision          : 04
 Sensor Type           : Critical Interrupt
 Sensor Number         : c4
 Event Type            : Sensor-specific Discrete
 Event Direction       : Assertion Event
 Event Data (RAW)      : 08ffff
 Event Interpretation  : Missing
 Description           : Bus Uncorrectable error

Sensor ID              : PCI-E Bus Error (0xc4)
Entity ID              : 49.0 (PCI Express Bus)
Sensor Type            : Critical Interrupt (0x13)


linux-e8j8:~ # ipmitool -H <iLO IP Address> -u <username> -p <password>-I lanplus
sel get 0x0c

SEL Record ID          : 000c
 Record Type           : 02
 Timestamp             : 06/23/2015 07:29:12
 Generator ID          : 0001
 EvM Revision          : 04
 Sensor Type           : Critical Interrupt
 Sensor Number         : c5
 Event Type            : Sensor-specific Discrete
 Event Direction       : Assertion Event
 Event Data (RAW)      : 00ffff
 Event Interpretation  : Missing
 Description           : NMI/Diag Interrupt

Sensor ID              : OS_NMI (0xc5)
Entity ID              : 35.0 (Operating System)
Sensor Type            : Critical Interrupt (0x13)
```

Inventory

IPMI supports multiple sets of nonvolatile FRU information for different parts in the system. This feature provides access to data such as the serial number, part number, asset tag, and other information about major modules in the system. Some examples of the major modules include the baseboard, chassis, processors, memory, power supplies, and the management controller. This information is available when the system is powered off or nonoperational. It facilitates the creation of automated remote inventory and service applications. IPMItool can read and display full FRU information for the system as well as detailed descriptions of power supplies and full DIMM SPD data.

Output from the `fru print` command

```
root@USER:/# ipmitool -I lanplus -H <iLO IP address> -u <username> -p <password>
fru print

FRU Device Description : Builtin FRU Device (ID 0)
 Chassis Type          : Rack Mount Chassis
 Chassis Serial        : XXXXXXXXXX
 Board Mfg Date        : Tue Apr 25 05:30:00 2017
 Board Mfg             : HPE
 Board Product         : ProLiant DL580 Gen10
 Board Serial          : XXXXXXXXXX
 Board Part Number     : WC1949777007
 Product Manufacturer  : HPE
 Product Name          : ProLiant DL580 Gen10
 Product Part Number   : WC1949777007
 Product Serial        : XXXXXXXXXX
.
.
.
Product Version       : 01
Product Serial        : XXXXXXXXXXXXXX
```

Chassis management

This feature provides standardized chassis status and control functions that allow a remote system to be turned on and off or rebooted without manual intervention. It also provides commands for causing the chassis to physically identify itself with an implementation-dependant mechanism. Some example mechanisms include turning on visible lights, displaying messages on an LCD, or emitting beeps through a speaker. IPMItool supports the available chassis management commands. Using it can eliminate trips to the data center or server room to reset a machine or identify a system that you want to remove from a rack.

Sample `chassis power` commands

```
root@USER:/# ipmitool -I lanplus -H <iLO IP address> -u <username> -p <password> sdr
list all
ZoMC             | Static MC @ 20h   | ok
254              | Log FRU @FEh f0.60 | ok
IPMB0 Phys Link  | 0x00              | ok
ChasMgmtCtlr1    | Static MC @ 44h   | ok
PsMgmtCtlr1      | Dynamic MC @ 52h  | ok
PsMgmtCtlr2      | Dynamic MC @ 54h  | ok
PsMgmtCtlr3      | Dynamic MC @ 56h  | ok
PsMgmtCtlr4      | Dynamic MC @ 58h  | ok
CaMC             | Static MC @ A6h   | ok
CaMC             | Static MC @ B8h   | ok
CaMC             | Static MC @ DAh   | ok
CaMC             | Static MC @ A8h   | ok
CaMC             | Static MC @ 82h   | ok
```

```
CaMC             | Static MC @ 84h  | ok
CaMC             | Static MC @ 8Eh  | ok
CaMC             | Static MC @ 90h  | ok
```

```
root@USER:/# ipmitool -I lanplus -H <iLO IP address> -u <username> -p <password>
-T 0x82 -b 7 -t 0x72 power status
Chassis Power is off
```

The provided examples show only a portion of the available output. The full output is richer and tells a full story about the system health and status. In addition, verbose output options that increase the output information are available. For more information, see **Verbose output examples**.

ipmitool

Syntax

```
ipmitool [-chvV] [-I open <command>]
```

```
ipmitool [-chvV] -I lan -H <hostname>
[-p <port>]
[-U <username>]
[-A <authtype>]
[-L <privlvl>]
[-aEPf <password>]
[-o <oemtype>]
<command>
```

```
ipmitool [-chvV] -I lanplus -H <hostname>
[-p <port>]
[-U <username>]
[-L <privlvl>]
[-aEPf <password>]
[-o <oemtype>]
[-C <ciphersuite>]
<command>
```

Description

IPMItool allows you to manage the IPMI functions of a local system through a kernel device driver. You can manage a remote system by using IPMI v1.5 and IPMI v2.0. Supported features include printing FRU information, LAN configuration, sensor readings, and remote chassis power control.

IPMI management of a local system interface requires the installation and configuration of a compatible IPMI kernel driver. On Linux systems, this driver is called `OpenIPMI`. It is included in standard Linux distributions.

Options

`-a`

Prompt for the remote server password.

`-A <authtype>`

Specify the authentication type to use during IPMI v1.5 LAN session activation. Supported types are NONE, PASSWORD, MD5, or OEM.

`-c`

Present output in CSV format. Not available with all commands.

-C *<ciphersuite>*

The remote server authentication, integrity, and encryption algorithms to use for IPMI v2 lanplus connections. The default value is 3. It specifies RAKP-HMAC-SHA1 authentication, HMAC-SHA1–96 integrity, and AES-CBC-128 encryption algorithms.

-E

The remote server password is specified by the environment variable *ipmi_password*.

-f *<password_file>*

Specifies a file containing the remote server password. If this option is absent, or if the *<password_file>* is empty, the password defaults to NULL.

-h

Get basic usage help from the command line.

-H *<address>*

The remote server address can be *IP address* or *hostname*. This option is required for the LAN and LANPLUS interfaces.

-I *<interface>*

Selects the IPMI interface. Supported interfaces are displayed in the usage help output.

-L *<privlvl>*

Force session privilege level. The default value is *admin*.

-m *<local address>*

Sets the local IPMB address. The default value is *0x20*.

-o *<oemtype>*

Selects the OEM type. Use `-o list` to view the list of supported OEM types.

-p *<port>*

Sets the remote server UDP port. The default value is 623.

-P *<password>*

A remote server password specified on the command line. Hewlett Packard Enterprise does not recommend specifying a password on the command line.

NOTE: If a password method is not specified, the IPMI tool prompts the user for a password. If no password is entered, the remote server password is set to NULL.

-t *<target address>*

Bridge IPMI requests to the remote target address.

-U *<username>*

Remote server username. The default value is **NULL**.

-v

Increases the verbose output level. This option can be specified multiple times to increase the level of debug output. For example, specifying this option three times results in hexdumps of all incoming and outgoing packets.

-V

Displays version information.

IPMItool raw command syntax and example

Syntax

Target command towards specific virtual controller.

- `-b <ipmi channelnumber>`
- `-t <target slave address>`
- `-m <source slave address>`
- Chassis controller: `-b 0 -t 0x44 -m 0x20`
- Power supply A controller: `-b 0 -t 0x52 -m 0x20`
- Power supply B controller: `-b 0 -t 0x54 -m 0x20`
- Power supply C controller: `-b 0 -t 0x56 -m 0x20`
- Power supply D controller: `-b 0 -t 0x58 -m 0x20`

Example

Raw Get Device ID to chassis satellite controller over LAN:

```
ipmitool -I lanplus -H <iLO IP address> -u <username> -p <password> -L Administrator
-b 0 -t 0x44 -m 0x20 raw 6 1

15 01 02 01 02 29 0b 00 00 00 85 00 00 00 00
```

Command specification

IPMI provides standardized interfaces and commands for configuring the platform management subsystem. This standardization enables cross-platform software to SDRs are an example of the interface for configuring sensor population and behavior on a system. There are also commands for configuring capabilities such as LAN and serial/ modem remote protocols, user passwords and privilege levels, platform event filtering, alert destinations, and others.

This section provides specifications for elements that apply to all requests and responses.

For more information, see **Completion codes**.

Unless otherwise noted, reserved bits and fields in commands (request messages) and responses are written as 0. Applications must ignore the state of reserved bits when they are read.

Unless otherwise specified, commands that are listed as mandatory must be accessed via LUN 00b. An implementation may elect to make any command available on any LUN or channel as long as it does not conflict with other requirements in this specification.

Command table notation

The following section includes command tables that list the data that is included in a request or a response for each command. The completion code for a response is included as the first byte of the response data field for each command. The NetFn and command byte values for each command are specified in separate tables.

The following notation is used in the command tables.

Notation	Description
Request data	Identifies the portion of the table that lists the fields that are included in the data portion of a request message for the given command.
Response data	Identifies the portion of the table that lists the fields that are included in the data portion of a response message for the given command. The completion code is always listed as the first byte in the response data field.
4	Single byte field. A single value in the byte column of a command table is used to identify a single byte field. The value represents the offset to the field within the data portion of the message. In some cases a single byte field follows a variable length field in which case the single byte offset is represented with an alphabetic variable and number representing the single byte field's location relative to the end of the variable length field. For example: N+1.

Table Continued

Notation	Description
5:7	Multi-byte field. The byte column indicates the byte offset(s) for a given field. For a multi-byte field, the first value indicates the starting offset, the second value (following the colon) indicates the offset for the last byte in the field. For example, 5:7 indicates a three-byte field spanning byte offsets 5, 6, and 7. In some cases, multi-byte fields may be variable length, in which case an alphabetic variable is used to represent the ending offset, for example: 5:N. Similarly, a field may follow a variable length field. In this case the starting value is shown as an offset relative to the notation used for the previous field, for example, if the previous field were 5:N, the next field would be shown starting at N+1. A variable length field may follow a variable length field, in which case a relative starting offset is shown with an alphabetic value indicating a relative ending offset, for example, N+1:M.
(3)	Optional Fields. When used in the byte column of the command tables, parentheses are used to indicate optional data byte fields. These can be absent or present at the choice of the party generating the request or response message. Devices receiving the message are required to accept any legal combination of optional data byte fields. Unless otherwise indicated, if an optional byte field is present, all prior specified byte fields must also be present. Similarly, if an optional byte field is absent all following byte fields must also be absent. For example, suppose a request accepts 4 data bytes. If data byte 3 was shown in parentheses as (3), it would indicate that byte 3 and following were optional. A legal request could consist of just bytes [1 and 2], bytes [1, 2, and 3,] or bytes [1, 2, 3 and 4]. A request which eliminates byte 3, but includes byte 4. (a request with data bytes [1, 2, and 4]), is illegal. Multi-byte fields that are shown as optional cannot be split. Either all bytes for the field are present or absent. For example, if a four byte multi-byte field is listed as optional, it is illegal to include the first two bytes, but not the second two bytes.

Standard command specification

This section presents the commands that are common to all IPMI devices that follow this specification's message/command interface. This includes management controllers that connect to the system via a compatible message interface, as well as IPMB devices.

Global commands

IPMI management controllers shall recognize and respond to these commands via LUN 0.

Get device ID command

This command is available to BMC, ChMC, and PSMC.

This command is used to retrieve the intelligent device's hardware revision, firmware/software revision, and sensor and event interface command specification revision information. The command also returns information regarding the additional logical device functionality (beyond application and IPM device functionality) that is provided within the intelligent device, if any.

While broad dependence on OEM-specific functionality is discouraged, two fields in the response allow software to identify controllers for the purpose of recognizing controller specific functionality. These are the device ID and the product ID fields. A controller that just implements standard IPMI commands can set these fields to unspecified.

Table 6: Device ID command response data

Response data byte number	Data field
1	Completion code
2	Device ID. 00h = unspecified
3	Device revision • [7] — 1=device provides device SDRs and 0=device does not provide device SDRs. • [6:4) — Reserved. Return as 0. • [3:0] — Device revision, binary encoded.
4	Firmware revision 1 • [7] — Device available: 0=normal operation, device firmware, SDR repository update or self-initialization in progress. Firmware or SDR repository updates can be differentiated by issuing a `get SDR` command and checking the completion code. • [6:0] — Major firmware revision, binary encoded.
5	Firmware revision 2: minor firmware revision. BCD encoded.
6	IPMI version. Holds IPMI command specification version. BCD encoded. 00h = reserved. Bits 7:4 hold the least significant digit of the revision, while bits 3:0 hold the most significant bits. For example, a value of 51h indicates revision 1.5 functionality. 02h for implementations that provide IPMI v2.0 capabilities per this specification.
7	Additional device support (formerly called IPM device support) lists the IPMI logical device commands and functions supported by the controller that are in addition to the mandatory IPM and application commands. • [7] — Chassis device • [6] — Bridge (device responds to `bridge NetFn` commands) • [5] — IPMB event generator. Device generates event messages (platform event request messages) from the IPMB. • [4] — IPMB event receiver. Device accepts event messages (platform event request messages) from the IPMB. • [3] — FRU inventory device • [2] — SEL device • [1] — SDR repository device • [0] — Sensor device

Table Continued

Response data byte number	Data field
8:10	manufacturer ID, LS byte first. The manufacturer ID is a 20-bit value that is derived from the IANA private enterprise ID (see below). Most significant four bits = reserved (0000b). 000000h = unspecified. 0FFFFFh = reserved. This value is binary encoded. For example, the ID for the IPMI forum is 7154 decimal, which is 1BF2h, and would be stored in this record as F2h, 1Bh, 00h for bytes 8 through 10, respectively.
11:12	Product ID, LS byte first. This field can be used to provide a number that identifies a particular system, module, add-in card, or board set. The number is specified according to the manufacturer given by manufacturer ID (see below). 0000h = unspecified. FFFFh = reserved.
(13:16)	Auxiliary firmware revision information. This field is optional. If present, it holds additional information about the firmware revision, such as boot block or internal data structure version numbers. The meanings of the numbers are specific to the vendor identified by manufacturer ID (see below). When the vendor-specific definition is not known, generic utilities should display each byte as 2-digit hexadecimal numbers, with byte 13 displayed first as the most significant byte.

Additional specifications and descriptions for the device ID response fields:

Table 7: Additional device ID specifications

Device ID Specification	Description
Device ID/Device instance	Specified by the manufacturer identified by the manufacturer ID field it allows controller-specific software to identify the unique application command, OEM fields, and functionality that are provided by the controller. Controllers that have different application commands, or different definitions of OEM fields, are expected to have different device ID values. Controllers that implement identical sets of applications commands can have the same device ID in a given system. Thus, a standardized controller could be produced where multiple instances of the controller are used in a system, and all have the same device ID value. (The controllers would still be differentiable by their address, location, and associated information for the controllers in the SDRs.) The device ID is typically used in combination with the product ID field such that the device IDs for different controllers are unique under a given product ID. A controller can optionally use the device ID as an instance identifier if more than one controller of that kind is used in the system. Binary encoded.
Device revision	The least significant nibble is used to identify when significant hardware changes have been made to the implementation of the management controller that cannot be covered with a single firmware release. This field is used to identify two builds off the same code firmware base, but for different board fab levels. For example, device revision "1" might be required for fab X and earlier boards, while device revision "2" would be for fab Y and later boards. Binary encoded and unsigned.
Firmware revision 1	Major revision of the firmware. 7-bits. It is incremented on major changes or extensions of the functionality of the firmware, such as additions, deletions, or modifications of the command set. The device available bit is used to indicate whether normal command set operation is available from the device, or if it is operating in a state where only a subset of the normal commands are available. Typically because the device is in a firmware update state. It may also indicate that full command functionality is not available because the device is in its initialization phase or an SDR update is in progress. The revision information obtained when the device available bit is 1 is indicative of the code version that is in effect. The version information may vary with the device available bit state. Binary encoded and unsigned.
Firmware revision 2	Minor revision of the firmware, incremented for minor changes such as bug fixes. BCD encoded.

Table Continued

Device ID Specification	Description
IPMI version	The version of the IPMI specification in which the controller is compatible, indicating conformance with this document including event message formats and mandatory command support. The value is 02h for implementations that provide IPMI v2.0 capabilities per this specification. BCD encoded with bits 7:4 holding the least significant digit of the revision and bits 3:0 holding the most significant bits.
Additional device support	Indicates the logical device support that the device provides in addition to the IPM and application logical devices.
Manufacturer ID	Uses IANA (**http://www.iana.org/**). SMI network management private enterprise codes (enterprise numbers) for identifying the manufacturer responsible for the specification of functionality of the vendor (OEM) -specific commands, codes, and interfaces used in the controller. For example, an event in the SEL could have OEM values in the event record. An application that parses the SEL could extract the controller address from the event record contents and use it to send the `get device ID` command and retrieve the manufacturer ID. A manufacturer-specific application could then do further interpretation based on prior knowledge of the OEM field, while a generic cross-platform application would typically just use the ID to present the manufacturer's name alongside uninterpreted OEM event values. The manufacturer ID is for the manufacturer that defines the functionality of the controller, which is not necessarily the manufacturer that created the physical microcontroller. For example, vendor A may create the controller, but it gets loaded with vendor B's firmware. The manufacturer ID would be for vendor B, since they defined the controller's functionality. The manufacturer ID value from the `get device ID` command does not override manufacturer or OEM ID fields that are explicitly defined as part of a command or record format. If no vendor-specific functionality is defined, it is recommended that the field be loaded with the manufacturer ID for the manufacturer that is responsible for the controller's firmware, or the value FFFFh to indicate unspecified. Binary encoded and unsigned.

Table Continued

Device ID Specification	Description
Product ID	Used in combination with the manufacturer ID and device ID values to identify the product-specific element of the controller-specific functionality. This number is specified by the manufacturer identified by the manufacturer ID field. Typically, a controller-specific application would use the product ID to identify the type of board, module, or system that the controller is used in, instead of using the data from the FRU information associated with the controller.
Auxiliary firmware revision information	This field is optional. If present, it holds additional information about the firmware revision, such as boot block or internal data structure version numbers. The meanings of the numbers are specific to the vendor identified by manufacturer ID. When the vendor-specific definition is not known, generic utilities should display each byte as 2-digit hexadecimal numbers, with byte 13 displayed first as the most significant byte.

Cold reset command

This command is available to the MC. This command is not required to return a response in all implementations.

This command directs the responder's device to reinitialize its event, communication, and sensor functions. This causes the default setting of interrupt enables, event message generation, sensor scanning, threshold values, and other power up default state to be restored. If the device incorporates a self test, the self test also runs at this time.

Table 8: Cold reset command response data

Response data byte number	Data field
1	Completion code

NOTE: The `cold reset` command is provided for platform development, test, and platform-specific initialization and recovery actions. The system actions of the `cold reset` command are platform specific. Issuing a `cold reset` command could have adverse effects on system operation, particularly if issued during run-time. Therefore, the `cold reset` command should not be used unless all the side-effects for the given platform are known.

It is recognized that there are conditions where a given controller may not be able to return a response to a cold reset request message. Therefore, though recommended, the implementation is not required to return a response to the `cold reset` command. Applications should not rely on receiving a response as verification of the completion of a `cold reset` command.

Warm reset command

This command is available to the MC.

This command directs the responder's device to reset communications interfaces, but current configurations of interrupt enables, thresholds, and so on are left alone. A warm reset does not initiate the self test. The intent of the `warm reset` command is to provide a mechanism for cleaning up the internal state of the device and its communication interfaces. A `warm reset` resets communication state information such as sequence number and retry tracking, but does not reset interface configuration information such as addresses, enables, and so on. An application may try a warm reset if it determines a non-responsive communication interface, but it must also be capable of handling the side effects.

Table 9: Warm reset command response data

Response data byte number	Data field
1	Completion code

Get self test results command

This command is available to MC, ChMC, and PSMC.

This command directs the device to return its self test results, if any. A device implementing a self test normally runs that test on device power up as well as after `cold reset` commands. A device is allowed to update this field during operation if it has tests that run while the device is operating. Devices that do not implement a self test always return a 56h for this command.

While the self test only runs at particular times, the `get self test results` command can be issued any time the device is in a ready for commands state.

Table 10: Get self test results command response data

Response data byte number	Data field
1	Completion code.
2	• 55h — No error. All self tests passed. • 56h — Self test function not implemented in this controller. • 57h — Corrupted or inaccessible data or devices. • 58h — Fatal hardware error (system should consider MC inoperative). This indicates that the controller hardware (including associated devices such as sensor hardware or RAM) may need to be repaired or replaced. • FFh — Reserved. • All other — Device-specific internal failure. Refer to the particular device specification for definition.
3	For byte 2 =: • 55h, 56h, FFh: 00h • 58h, all other: Device-specific. • 57h: self-test error bitfield. A return of 57h does not imply that all tests were run, just that a given test has failed. For example, 1b means failed, 0b means unknown. • [7] — 1b = Cannot access SEL device. • [6] — 1b = Cannot access SDR repository. • [5] — 1b = Cannot access MC FRU device. • [4] — 1b = IPMB signal lines do not respond. • [3] — 1b = SDR repository empty. • [2] — 1b = Internal use area of MC FRU corrupted. • [1] — 1b = Controller update boot block firmware corrupted. • [0] — 1b = Controller operational firmware corrupted.

Get ACPI power state command

This command is available to the MC.

The command is used to retrieve the present power state information that has been set into the controller. This is an independent setting from the system power state that may not necessarily match the actual power state of the system. Unspecified bits and codes are reserved and returned as 0.

Table 11: Get ACPI power state command response data

Response data byte number	Data field		
1	Completion code		
2	ACPI system power state		
	[7] — Reserved		
	[6:0] — System power state enumeration		
	00h	S0 / G0	Working
	01h	S1	Hardware context maintained, typically equates to processor/chip set clocks stopped
	02h	S2	Typically equates to stopped clocks with processor/cache context lost
	03h	S3	Typically equates to suspend-to-RAM
	04h	S4	Typically equates to suspend-to-disk
	05h	S5 / G2	Soft off
	06h	S4/S5	Soft off, cannot differentiate between S4 and S5
	07h	G3	Mechanical off
	08h	sleeping	Sleeping - cannot differentiate between S1-S3
	09h	G1 sleeping	Sleeping - cannot differentiate between S1-S4
	0Ah	Override	S5 entered by override
	20h	Legacy on	Legacy on (indicates on for system that does not support ACPI or has ACPI capabilities disabled)
	21h	Legacy off	Legacy soft-off
	2Ah	Unknown	Power state has not been initialized, or device lost track of power state
3	ACPI device power state		

Table Continued

Response data byte number	Data field	
	[7] — Reserved	
	[6:0] — Device power state enumeration	
00h	D0	
01h	D1	
02h	D2	
03h	D3	
02h	Unknown	Power state has not been initialized, or device lost track of power state

Broadcast get device ID command

This command is available to MC, ChMC, and PSMC. It is only relevant to satellite controllers.

This command is the broadcast version of the `get device ID` command which provides for the discovery of intelligent devices on the IPMB only. Request is formatted as an entire IPMB application request message, from the RsSA field through the second checksum, with the message prefixed with the broadcast slave address, 00h. Response format is same as the regular get device ID response.

The `broadcast get device ID` command is not bridged but can be delivered to the IPMB using master write-read commands.

To perform a discovery, the command is repeatedly broadcast with a different rsSA slave address parameter field specified in the command. The device that has the matching physical slave address information shall respond with the same data it would return from a regular (non-broadcast) `get device ID` command. Since an IPMB response message carries the responder's slave address, the response to the broadcast provides a positive confirmation that an intelligent device exists at the slave address given by the rsSA field in the request.

An application driving discovery then cycles through the possible range of IPMB device slave addresses to find the population of intelligent devices on the IPMB.

See **Get device ID command** for information on the fields returned by the `broadcast get device ID` command response. The IPMB message format for the `broadcast get device ID` ID request exactly matches that for the `get device ID` command, with the exception that the IPMB message is prefixed with the 00h broadcast address. The following illustrates the format of the IPMB broadcast get device ID request message:

Broadcast (00h)	rsSA	netFn/rsLUN	check1	rqSA	rqSeq/rqLUN	Cmd (01h)	check2

Figure 2: Broadcast get device ID request message

Addresses 00h-0Fh and F0h-FFh are reserved for I^2C functions and not used for IPM devices on the IPMB. These addresses can therefore be skipped if using the `broadcast get device ID` command to scan for IPM devices. The remaining fields follow the regular IPMB definitions.

In order to speed the discovery process on the IPMB, a controller should drop off the bus as soon as it sees that the rsSA in the command does not match its rsSA.

IPMI messaging support commands

This section defines the commands used to support the system messaging interfaces. This includes control bits for using the MC as an event receiver and SEL device. SMM messaging and event message buffer support is optional. Use of IPMI support for SMI and SMM messaging is deprecated. Configuration interface support for enabling or disabling SMM messaging and corresponding SMI has been removed from the specification. If SMM messaging were implemented using the IPMI infrastructure, it would now be done as an OEM-proprietary capability.

System software that is not explicitly aware of the particular platform's use of SMI messaging must assume that the any SMI options have been pre-configured by the controller, system BIOS, or other software. Therefore, runtime system software should not reconfigure SMI options, nor should it access the event message buffer if it finds that event message buffer interrupt is mapped to SMI. The effects of SMS accessing the event message buffer when it is configured for SMI are unspecified.

Set BMC global enables command

This command is available to the MC.

This command is used to enable message reception into message buffers, and any interrupt associated with that buffer getting full. The OEM0, OEM 1, and OEM 2 flags are available for definition by the OEM/system integrator. Generic system management software must not alter these bits.

Table 12: Set BMC global enables command request and response data

Request data byte number	Data field		
1	This field is set to xxxx_100xb on power-up and system resets. If the implementation allows the receive message queue interrupt to be enabled/disabled, the default for bit 0 should be 0b.		

	[7]	OEM 2 Enable	Generic system management software must do a 'read-modifywrite' using the
			`get BMC global enables`
			and
			`set BMC global enables`
			commands to avoid altering this bit.

	[6]	OEM 1 Enable	
	[5]	OEM 0 Enable	
	[4]	Reserved	
	[3]	1b =	Enable system event logging (enables/disables logging of events to the SEL - with the exception of events received over the system interface. Event reception and logging via the system interface is always enabled.) SEL logging is enabled by default whenever the MC is first powered up. It is recommended that this default state also be restored on system resets and power on.
	[2]	1b =	Enable event message buffer. Error completion code returned if written as 1 and the event message buffer not supported.
	[1]	1b =	Enable event message buffer full interrupt.

Table Continued

[0]	1b =	Enable receive message queue interrupt (this bit also controls whether KCS communication interrupts are enabled or disabled. An implementation is allowed to have this interrupt always enabled.)

NOTE: If the event message buffer full or receive message queue interrupt are not supported, an implementation can elect to return a CCh error completion code for the `set BMC global enables` command if an attempt is made to enable the interrupt (this is the recommended implementation). Alternatively, the implementation can accept the command, but must return 0b for the corresponding bit in the `get BMC global enables`.

Response data byte number	Data field
1	Completion code

Get BMC global enables command

This command is available to the MC.

This command is used to retrieve the present setting of the global enables. The OEM0, OEM 1, and OEM 2 flags are available for definition by the OEM/system integrator. Generic system management software must ignore these bits.

Table 13: Get BMC global enables command response data

Response data byte number	Data field		
1	Completion code		
2	[7]	1b =	OEM 2 enabled
	[6]	1b =	OEM 1 enabled
	[5]	1b =	OEM 0 enabled
	[4]	Reserved	
	[3]	1b =	System event logging enabled
	[2]	1b =	Event message buffer enabled
	[1]	1b =	Event message buffer full interrupt enabled
	[0]	1b =	Receive message queue interrupt enabled (this bit also indicates whether KCS communication interrupt is enabled or disabled.)
	If the receive message queue or event message full interrupts are not implemented the corresponding interrupt enabled status bit must return as 0b.		

Clear message flags command

This command is available to the MC.

This command is used to flush unread data from the receive message queue or event message buffer. This will also clear the associated buffer full/message available flags. See **Get message flags command** .

Table 14: Clear message flags command request and response data

Request data byte number	Data field		
1	[7]	1b =	Clear OEM 2
	[6]	1b =	Clear OEM 1
	[5]	1b =	Clear OEM 0
	[4]	Reserved	
	[3]	1b =	Clear watchdog pre-timeout interrupt flag
	[2]	Reserved	
	[1]	1b =	Clear event message buffer
	[0]	1b =	Clear receive message queue
	If the receive message queue or event message full interrupts are not implemented the corresponding interrupt enabled status bit must return as 0b.		
Response data byte number	**Data field**		
1	Completion code. Implementations are not required to return an error completion code if an attempt is made to clear the event message buffer flag but the event message buffer is not supported.		

Get message flags command

This command is available to the MC.

This command is used to retrieve the present message available states. The OEM0, OEM 1, and OEM 2 flags are available for definition by the OEM/system integrator. Generic system management software must ignore these bits.

Table 15: Get message flags command response data

Request data byte number	Data field		
1	Completion code		
2	Flags		
	[7]	1b =	OEM 2 data available.
	[6]	1b =	OEM 1 data available.
	[5]	1b =	OEM 0 data available.
	[4]	Reserved.	
	[3]	1b =	Watchdog pre-timeout interrupt occurred.
	[2]	Reserved.	
	[1]	1b =	Event message buffer full. Return as 0 if event message buffer is not supported, or when the event message buffer is disabled.
	[0]	1b =	Receive message available. One or more messages ready for reading from receive message queue.

Enable message channel receive command

This command is available to the MC.

This command is used to enable and disable message reception into the receive message queue from a given message channel. The command provides a mechanism to allow SMS to only receive messages from channels that it intends to process, and provides a disable mechanism in case the receive message queue is being erroneously or maliciously flooded with requests on a particular channel. It does not affect the ability for SMS to transmit on that channel. Only the SMS message channel is enabled by default. All other channels must be explicitly enabled by BIOS or system software, as appropriate. It is recommended that a destination unavailable completion code be returned if a request message to SMS is rejected because reception has been disabled.

Table 16: Enable message channel receive command request and response data

Request data byte number	Data field
1	Channel number • [7:4] — Reserved • [3:0] — Channel number

Table Continued

2	Channel state • [7:2] — Reserved • [1:0] ◦ 00b = Disable channel ◦ 01b = Enable channel ◦ 10b = Gen channel enable/disable state ◦ 11b = Reserved

Response data byte number	Data field
1	Completion code
2	Channel number • [7:4] — Reserved • [3:0] — Channel number
3	Channel state • [7:1] — Reserved • [0] ◦ 1b = Channel enabled ◦ 0b = Channel disabled

Get message command

This command is available to the MC.

This command is used to get data from the receive message queue.

Software is responsible for reading all messages from the message queue even if the message is not the expected response to an earlier send message. System management software is responsible for matching responses up with requests.

The `get message command` includes an inferred privilege level that is returned with the message. This can help avoid the need for software to implement a separate privilege level and authentication mechanism. For example: A user activates a session with a maximum privilege level of Administrator on a multi-session channel, and an MD5 authentication type was negotiated. That user-level authentication is disabled. A user that has user or higher privilege can place messages into the receive message queue by sending them to LUN 10b, or by using the `send message` command. If the packet has authentication type = MD5, the packet is assigned an inferred privilege level based the on the present operating privilege level for the user (set using the `set session privilege level` command). If, before sending the packet, the user had set their privilege level to Operator, the packet would be assigned an inferred privilege level of Operator. This means an authenticated (signed) packet can be assigned different inferred privilege levels based on the present operating privilege set by the `set session privilege level` command.) If the message is received in a packet that has authentication type = none, the packet is assigned an inferred privilege level of User, since that is the lowest privilege level for which that type of authentication is accepted.

Now suppose that the remote user had used the receive message queue as a way to send a message to system management software that requests a soft shutdown of the operating system. The message would either have an inferred

privilege level of Operator or User depending on whether it was sent as an authenticated message or not. SMS can then use that inferred privilege level as part of deciding whether to accept and process the message or not. For single-session channels, the inferred privilege level is always set to the present operating privilege level. For session-less channels, the inferred privilege level is set to none, indicating that there was no IPMI-specified authentication operating on the channel from which the message was received.

Table 17: Get message command response data

Response data byte number	Data field
1	Completion code. Generic, plus the command specific completion code: 80h = data not available (queue / buffer empty). Implementation of this completion code is mandatory. The code eliminates the need for system software to always check the message buffer flags to see if there is data left in the receive message queue. If a non-OK, non-80h completion is encountered - software must check the message flags to get the empty/non-empty status of the receive message queue.

2	Channel number
	
	• [7:4] Inferred privilege level for message.
	
	When the MC receives a message for the receive message queue, it assigns an inferred privilege level to the message as follows:
	
	If the message is received from a session-based channel, it is initially assigned a privilege level that matches the maximum requested privilege level that was negotiated via the
	
	`activate session`
	
	command.
	
	If per-message authentication is enabled, but user-level authentication is disabled, the MC assigns a level of User to any messages that are received with an authentication type = none. (Per-message and user-level authentication options only apply to multi-session channels).
	
	The MC then lowers the assigned privilege limit, if necessary, based on the present session privilege limit that was set via the
	
	`set session privilege level`
	
	command.
	
	If the channel is session-less such as IPMB), the MC returns none for the privilege level.
	
	◦ 0h = None (unspecified)
	
	◦ 1h = Callback level
	
	◦ 2h = User level
	
	◦ 3h = Operator level
	
	◦ 4h = Administrator level
	
	◦ 5h = OEM proprietary level
	
	• [3:0] channel number

3:N	Message data. Maximum length and format dependent on protocol associated with the channel.

The following table indicates the contents of the Message Data field from the get message response according to the channel type and channel protocol that was used to place the message in the receive message queue.

Table 18: Get message data fields

	Originating channel type	Channel protocol	Message data for received requests (RQ) and responses (RS)
1	IPMB (I²C)	IPMB¹	RQ: netFn/rsLUN, chk1, rqSA, rqSeq/rqLUN, cmd, <data>, chk2 RS: netFn/rqLUN, chk1, rsSA, rqSeq/rsLUN, cmd, completion code, <data>, chk2
4	802.3 LAN	IPMB	RQ: Session handle, rsSWID, netFn/rsLUN, chk1, rqSWID or rqSA, rqSeq/rqLUN, cmd, <data>, chk2 RS: Session handle, rqSWID, netFn/rsLUN, chk1, rsSWID or rsSA, rqSeq/rsLUN, cmd, completion code, <data>, chk2
5	Asynch. serial/modem (RS-232)	IPMB (basic mode, terminal mode, and PPP mode)	RQ: Session handle, rsSWID, netFn/rsLUN, chk1, rqSWID or rqSA, rqSeq/rqLUN, cmd, <data>, chk2 RS: Session handle, rqSWID, netFn/rsLUN, chk1, rsSWID or rsSA, rqSeq/rsLUN, cmd, completion code, <data>, chk2 **NOTE:** When LUN 10b is used to deliver a message to SMS from a terminal mode remote console, the MC inserts fixed values for the SWIDs and LUNs in the message. Messages from the remote console are always returned as coming from SWID 40h (81h) LUN 00b, and going to SMS SWID 20h (41h) LUN 00b.
6	Other LAN	IPMB	RQ: Session handle, rsSWID, netFn/rsLUN, chk1, rqSWID or rqSA, rqSeq/rqLUN, cmd, <data>, chk2 RS: Session handle, rqSWID, netFn/rsLUN, chk1, rsSWID or rsSA, rqSeq/rsLUN, cmd, completion code, <data>, chk2
7	PCI SMBus	IPMI-SMBus	RQ: rsSA, Netfn(even)/rsLUN, 00h, rqSA, rqSeq/rqLUN, CMD, <data>, PEC RS: rqSA or rqSWID, NetFn(odd)/rqLUN, 00h, rsSA or rsSWID, rqSeq/rsLUN, CMD, completion code, <data>, PEC
8	SMBus v1.0/1.1		
9	SMBus v2.0		
10	Reserved for USB 1.x	n/a	n/a

Table Continued

	Originating channel type	Channel protocol	Message data for received requests (RQ) and responses (RS)
11	Reserved for USB 2.x	n/a	n/a
12	System interface	BT, KCS, SMIC	RQ: Session handle, rsSWID, netFn/rsLUN, chk1, rqSWID or rqSA, rqSeq/rqLUN, cmd, <data>, chk2 RS: Session handle, rqSWID, netFn/rsLUN, chk1, rsSWID or rsSA, rqSeq/rsLUN, cmd, completion code, <data>, chk2

[1] This message data matches the IPMB message format with the leading slave address omitted. The format includes checksums. In order to verify those checksums, they must be calculated as if 20h (MC slave address) was the value that was used as the slave address when the checksums were calculated per [IPMB]. 20h is always used for the checksum calculation for the receive message queue data whenever IPMB is listed as the originating bus and with IPMB as the channel protocol.

Send message command

This command is available to the MC.

The `send message` command is used for bridging IPMI messages between channels, and between the SMS and a given channel.

For IPMI v2.0 the `send message` command has been updated to include the ability to indicate whether a message must be sent authenticated or with encryption (for target channels on which authentication and/or encryption are supported and configured).

Table 19: Send message command request and response data

Request data byte number	Data field		
1	Channel number		
	[7:6]	00b =	No tracking. The MC reformats the message for the selected channel but does not track the originating channel, sequence number, or address information. This option is typically used when software sends a message from the system interface to another media. Software will typically use no tracking when it delivers sends a message from the system interface to another channel, such as IPMB. In this case, the software formats the encapsulated message so that when it appears on the other channel, it appears to have been directly originated by MC LUN 10b. See **MC LUN 10b** for more information.

Table Continued

		01b =	Track request. The MC records the originating channel, sequence number, and addressing information for the requester, and then reformats the message for the protocol of the destination channel. When a response is returned, the MC looks up the requester's information and format the response message with the framing and destination address information and reformats the response for delivery back to the requester. This option is used for delivering IPMI request messages from non-SMS (non-system interface) channels. See **Send Message command with response tracking** for more information.
		10b =	Send raw (optional). This option is primarily provided for test purposes. It may also be used for proprietary messaging purposes. The MC simply delivers the encapsulated data to the selected channel in place of the IPMI message data. If the channel uses sessions, the first byte of the message data field must be a session handle. The MC must return a non-zero completion code if an attempt is made to use this option for a given channel and the option is not supported. It is recommended that completion code CCh be returned for this condition.
		11b =	Reserved
	[5]	1b =	Send message with encryption. MC returns an error completion code if this encryption is unavailable.
		0b =	Encryption not required. The message is sent unencrypted if that option is available under the given session. Otherwise, the message is sent encrypted.
	[4]	1b =	Send message with authentication. MC returns an error completion code if this authentication is unavailable.
		0b =	Authentication not required. Behavior is dependent on whether authentication is used and whether the target channel is running an IPMI v1.5 or IPMI v2.0/RMCP+ session, as follows: • IPMI v1.5 sessions default to sending the message with authentication if that option is available for the session. • IPMI v2.0/RMCP+ sessions send the message unauthenticated if that option is available under the session. Otherwise, the message is sent with authentication.
	[3:0]	Channel number where to send the message	
2:N	Message data. Format dependent on target channel type.		

Request data byte number	**Data field**

Table Continued

1	Completion code Generic, plus additional command-specific completion codes: 80h = Invalid session handle. The session handle does not match up with any currently active sessions for this channel. If channel medium = IPMB, SMBus, or PCI management bus (This status is recommended for applications that need to access low-level I^2C or SMBus devices). • 81h = Lost arbitration • 82h = Bus error • 83h = NAK on write
(2:N)	Response data This data will only be present when using the `send message` command to originate requests from IPMB or PCI management bus to other channels such as LAN or serial/modem. It is not present in the response to a `send message` command delivered via the system interface.

NOTE: The MC does not parse messages that are encapsulated in a `send message` command and does not know what privilege level should be associated with an encapsulated message. Thus, messages that are sent to a session using the `send message` command are always output using the authentication type that was negotiated when the session was activated.

The following table summarizes the contents of the message data field when the `send message` command is used to deliver an IPMI message to different channel types. In most cases, the format of message information the message data field follows are the same used for the IPMB, with two typical exceptions:

- When the message is delivered to channels without physical slave devices, a SWID field takes the place of the slave address field.
- When the message is delivered to a channel that supports sessions, the first byte of the message data holds a session handle.

Table 20: Send message data fields

	Target channel type	Target channel protocol	Message data for sending requests (RQ) and responses (RS)
1	IPMB (I^2C)	IPMB	RQ: rsSA, netFn/rsLUN, chk1, rqSA, rqSeq/rqLUN, cmd, <data>, chk2 RS: rqSA, netFn/rqLUN, chk1, rsSA, rqSeq/rsLUN, cmd, completion code, <data>, chk2
4	802.3 LAN	IPMB+session header	RQ: Session handle, rsSWID, netFn/rsLUN, chk1, rqSWID or rqSA, rqSeq/rqLUN, cmd, <data>, chk2 RS: Session handle[1], rqSWID, netFn/rsLUN, chk1, rsSWID or rsSA, rqSeq/rsLUN, cmd, completion code, <data>, chk2

Table Continued

	Target channel type	Target channel protocol	Message data for sending requests (RQ) and responses (RS)
5	Asynch. serial/modem (RS-232)	IPMB (basic mode, terminal mode, and PPP mode)	RQ: Session handle[1], rsSWID, netFn/rsLUN, chk1, rqSWID or rqSA, rqSeq/rqLUN, cmd, <data>, chk2 RS: Session handle[1], rqSWID, netFn/rsLUN, chk1, rsSWID or rsSA, rqSeq/rsLUN, cmd, completion code, <data>, chk2 **NOTE:** Terminal mode has a single, fixed SWID for the remote console. Software using send message to deliver a message to a terminal mode remote console should use their SWID or slave address as the source of the request or response, and the terminal mode SWID (40h) as the destination.
6	Other LAN	IPMB	RQ: Session handle[1], rsSWID, netFn/rsLUN, chk1, rqSWID or rqSA, rqSeq/rqLUN, cmd, <data>, chk2 RS: Session handle[1], rqSWID, netFn/rsLUN, chk1, rsSWID or rsSA, rqSeq/rsLUN, cmd, completion code, <data>, chk2
7	PCI SMBus	IPMI-SMBus	RQ: rsSA, netFn/rsLUN, chk1, rqSA, rqSeq/rqLUN, cmd, <data>, chk2 RS: rqSA, netFn/rqLUN, chk1, rsSA, rqSeq/rsLUN, cmd, completion code, <data>, chk2
8	SMBus v1.0/1.1		
9	SMBus v2.0		
10	Reserved for USB 1.x	n/a	n/a
11	Reserved for USB 2.x	n/a	n/a
12	System interface		RQ: rsSA, netFn/rsLUN, chk1, rqSA, rqSeq/rqLUN, cmd, <data>, chk2 RS: rqSA, netFn/rqLUN, chk1, rsSA, rqSeq/rsLUN, cmd, completion code, <data>, chk2 **NOTE:** MC adds session handle information when it puts the message into the receive message queue .

[1] The session handle identifies a particular active session on the given channel. The MC assigns a different value to each time a new session is activated. A typical implementation keeps track of the last value that was assigned and increment it before assigning it to a new active session when the `activate session` command has been accepted. Software must include this field for channels where the `get channel info` command indicates that the channel supports sessions.

Get system GUID command

This command is available to the MC.

This optional, though highly recommended, command can be used to return a GUID (also known as a UUID), for the managed system to support the remote discovery process and other operations. The format of the ID follows the octet format specified in [RFC 4122]. [RFC4122] specifies four different versions of UUID formats and generation algorithms suitable for use for a GUID in IPMI.

- Time based — version 1 (0001b)
- Name based:
  - version 3 (0011b) MD5 hash
  - version 4 (0100b) Pseudo-random
  - version 5 SHA1 hash

[SMBIOS] does not specify a particular specification or version for UUID generation. In general, if it remains unspecified, the version 1 format is recommended by the IPMI specification for new system implementations. However, versions 3, 4, or 5 formats are also allowed. A system GUID should not change over the lifetime of the system.

If the MC is on a removable card that can be moved to another system, the vendor of the card or system vendor should provide a mechanism for generating a new system GUID or retrieving the SMBIOS UUID from the given system.

Since the GUID is typically permanently assigned to a system, an interface that would allow the GUID to be configured or changed is not specified. For systems that support [SMBIOS], the system GUID that is returned by the MC should match the UUID field value in the SMBIOS system information (type 1) record.

The session header (session request data and session response data) values shown in the following table illustrate the values that would be used to execute a `get system GUID` command outside of an active session. The get system GUID is always accepted outside of an active session. The `get system GUID` command can also be executed within the context of an active session (providing the user is operating at higher than callback privilege). When the `get system GUID` command is executed in the context of an active session, the session header fields must have correct values according to the authentication, session ID, and session sequence number information that was negotiated for the session.

Session header fields request and response data prior when used prior to session activation

authentication type = NONE

session seq# = null (0's)

Session ID = null (0's)

AuthCode = NOT PRESENT

Table 21: Get system GUID command response data

Response data byte number	Data field
1	Completion code
2:17	GUID bytes 1 through 16.

Set system info parameters command

This command is available to the MC.

This command is used for setting system information parameters such as system name and BIOS/system firmware revision information.

Table 22: Set system info parameters command request and response data

Request data byte number	Data field
1	Parameter selector
2:n	Configuration parameter data, per **Get system info parameters command**.
Response data byte number	**Data field**
1	Completion code. Generic plus the following command-specific completion codes: • 80h = parameter not supported • 81h = attempt to set the set in progress value (in parameter #0) when not in the set complete state. (This completion code provides a way to recognize that another party has already claimed the parameters). • 82h = attempt to write read-only parameter

Get system info parameters command

This command is available to the MC.

This command is used for retrieving system information parameters from the set system info parameters command.

Table 23: Get system info parameters command request and response data

Request data byte number	Data field
1	[7] • 0b = get parameter • 1b = get parameter revision only [6:0] — reserved
2	Parameter selector
3	Set selector. Selects a given set of parameters under a given parameter selector value. 00h if parameter does not use a set selector.
4	Block selector (00h if parameter does not require a block number)
Response data byte number	**Data field**
1	Completion code Generic codes, plus the command-specific completion code: 80h = parameter not supported.

Table Continued

2	[7:0] — Parameter revision. Format: • MSN = present revision • LSN = oldest revision parameter that is backward compatible • 11h for parameters in this specification
The following data bytes are not returned when the get parameter revision only bit is 1b.	
3:N	Configuration parameter data, per the following *System info parameters* table. . If the rollback feature is implemented, the MC makes a copy of the existing parameters when the set in progress state becomes asserted. (See the set in progress parameter #0). While the set in progress state is active, the MC returns data from this copy of the parameters, plus any uncommitted changes that were made to the data. Otherwise, the MC returns parameter data from non-volatile storage.

Table 24: System info parameters

Parameter	#	Parameter data (non-volatile unless otherwise noted)[1]		
Set in progress (volatile)	0	Data 1 - This parameter is used to indicate when any of the following parameters are being updated, and when the updates are completed. The bit is primarily provided to alert software that some other software or utility is in the process of making changes to the data. An implementation can also elect to provide a rollback feature that uses this information to decide whether to roll back to the previous configuration information, or to accept the configuration change. If used, the roll back restores all parameters to their previous state. Otherwise, the change takes effect when the write occurs.		
		[7:2]	Reserved	
		[1:0]	00b =	Set complete. If a system reset or transition to powered down state occurs while set in progress is active, the MC goes to the set complete state. If rollback is implemented, going directly to set complete without first doing a commit write causes any pending write data to be discarded.
			01b =	Set in progress indicating that some utility or other software is presently doing writes to parameter data. It is a notification flag only, it is not a resource lock. The MC does not provide any interlock mechanism that would prevent other software from writing parameter data while set in progress value is present on these bits.

Table Continued

Parameter	#	Parameter data (non-volatile unless otherwise noted)[1]	
		10b =	Commit write (optional). This is only used if a rollback is implemented. The MC saves the data that has been written since the last time the set in progress and then go to the set in progress state. An error completion code is returned if this option is not supported.
		11b =	Reserved
System firmware version	1	System firmware version string in text. System firmware that requires multiple strings to represent version information can separate those strings using 00h as the delimiter for ASCII+LATIN1 and UTF-8 encoded string data, or 0000h for UNICODE encoded string data. For IA32 and EMT64 utilizing non-EFI system firmware, it is recommended that four blocks (64 bytes) of storage be provided. For EFI-based systems, 256 bytes is recommended. **NOTE:** System firmware may optionally include a routine that checks during POST to see if this parameter is up-to-date with the present firmware version, and if not, update it automatically. Other implementations may elect to automatically update this parameter when system firmware updates occur.	
		Data 1 —	set selector = 16-byte data block number to access, 0 based. Two data blocks (32-bytes) for string data required, at least three recommended. Number of effective characters is dependent on the encoding selected in string data byte 1.
		Data 2:17 —	16-byte block for system firmware name string data For the first block of string data (set selector = 0), the first two bytes indicate the encoding of the string and its overall length as follows: String data byte 1: • [7:4] — Reserved • [3:0] — Encoding ◦ 0h = ASCII+Latin1 ◦ 1h = UTF-8 ◦ 2h = UNICODE ◦ All other = Reserved String data byte 2: [7:0] - String length (in bytes, 1-based)

Table Continued

Parameter	#	Parameter data (non-volatile unless otherwise noted)[1]
System name	2	System name. A name for the overall system to be associated with the MC. This may or may not match other names that are used for the system.
		Data 1 — set selector = 16-byte data block number to access, 0 based. Two data blocks (32-bytes) for string data required, at least three recommended. Number of effective characters will be dependent on the encoding selected in string data byte 1.
		Data 2:17 - 16-byte block for system name string data For the first block of string data (set selector = 0), the first two string data bytes indicate the encoding of the string and its overall length as follows. There is no required value to be set or used for any bytes that are past the string length. String data byte 1: • [7:4] — Reserved • [3:0] — Encoding ◦ 0h = ASCII+Latin1 ◦ 1h = UTF-8 ◦ 2h = UNICODE ◦ All other = Reserved String data byte 2: [7:0] - String length (in bytes, 1-based)
Primary operating system name (non-volatile)	3	Primary operating system name. The OS that the system boots to for this MC according to the default configuration of its system firmware. In systems that may have multiple physical partitions, this reflects the OS for the partition that holds the given MC. For systems that have virtual machine capability being utilized (where more than one virtual systems may be sharing a physical MC), it is recommended that this value hold the name of the virtual machine monitor (VMM) software or VMM type).
		Data 1 Set selector = 16-byte data block number to access, 0 based. Two data blocks (32-bytes) for string data required, at least three recommended. Number of effective characters will be dependent on the encoding selected in string data byte 1.

Table Continued

Parameter	#	Parameter data (non-volatile unless otherwise noted)[1]
		Data 2:17 16-byte block for system name string data For the first block of string data (set selector = 0), the first two bytes indicate the encoding of the string and its overall length as follows. There is no required value to be set or used for any bytes that are past the string length. String data byte 1: • [7:4] — Reserved • [3:0] — Encoding ◦ 0h = ASCII+Latin1 ◦ 1h = UTF-8 (ls-byte first) ◦ 2h = UNICODE (ls-byte first) ◦ All other = Reserved String data byte 2: [7:0] - string length (in bytes, 1-based)
Operating system name (volatile)	4	Present operating system name. The name of the operating system that is presently running and able to access this MC's system interface. The MC automatically clears this value by zeroing out the string length on system power cycles and resets. In systems that may have multiple physical partitions, this reflects the OS for the partition that holds the given MC. For systems that have virtual machine capability being utilized (where more than one virtual systems may be sharing a physical MC), it is recommended that this value hold the name of the virtual machine monitor (VMM) software or VMM type.
		Data 1 Set selector = 16-byte data block number to access, 0 based. Two data blocks (32-bytes) for string data required, at least three recommended. Number of effective characters is dependent on the encoding selected in string data byte 1.

Table Continued

Parameter	#	Parameter data (non-volatile unless otherwise noted)[1]
	Data 2:17	16-byte block for system name string data For the first block of string data (set selector = 0), the first two bytes indicate the encoding of the string and its overall length as follows. There is no required value to be set or used for any bytes that are past the string length. String data byte 1: • [7:4] — Reserved • [3:0] — Encoding ◦ 0h = ASCII+Latin1 ◦ 1h = UTF-8 ◦ 2h = UNICODE ◦ All other = Reserved String data byte 2: [7:0] - String length (in bytes, 1-based)
OEM	192 … 255	This range is available for special OEM system information parameters.

[1] Choice of system manufacturing defaults for non-volatile parameters is left to the system manufacturer unless otherwise specified.

Master write-read command

This command can be used for low-level I^2C/SMBus write, read, or write-read access to the IPMB or private busses behind a management controller. The command can also be used for providing low-level access to devices that provide an SMBus slave interface.

NOTE: In HPE iLO, this command is not available over LAN.

Table 25: Master write-read command request and response data

IPMI request data byte number	Data field	
1	Bus ID:	
	[7:4]	Channel number (ignored when bus type=1b
	[3:1]	Bus ID, 0–based (always 000b for public bus ([bus type=0b])
	[0]	Bus type:

Table Continued

	0b =	Public (for example, IPMB or PIC Management) bus. The channel number value is used to select the target bus.
	1b =	Private bus (the bus ID value is used to select the target bus.)
2	Requested maximum privilege level	
	[7:1]	Slave address
	[0]	Reserved. Write a 0.
3	Read count. Number of bytes to read, 1 based. 0 equals not bytes to read. **The maximum read count should be at least 34 bytes.** This allows the command to be used for an SMBus Block Read. This is required if the command provides access to an SMBus or IPMB. Otherwise, if FRU SEEPROM devices are accessible, at least 31 bytes must be supported. Note than an implementation that supports fewer bytes can be supported if none of the devices to be accessed can handle the recommended minimum.	
4:N	Data to write. This command should support **at least 35 bytes** of write data. This allows the command to be used for an SMBus Block Write with PEC. Otherwise, if FRU SEEPROM devices are accessible, at least 31 bytes must be supported. Note that an implementation is allowed to support fewer bytes if none of the devices to can handle the recommended minimum.	
IPMI response data byte number	**Data field**	
1	Completion code A management controller shall return an error Completion Code if an attempt is made to access an unsupported bus. This is a generic response but also may include the following command specific codes: • 81h—Lost arbitration • 82h—Bus error • 83h—NAK on write • 84h—Truncated read	
(2:M)	Bytes read from specified slave address. This field will be absent if the read count is 0. The controller terminates the I^2C transaction with a STOP condition after reading the requested number of bytes.	

Get channel authentication capabilities command

This command is available to the MC.

This command is sent in unauthenticated (clear) format. This command is used to retrieve capability information about the channel from which the message is delivered, or for a particular channel. The command returns the authentication

algorithm support for the given privilege level. When activating a session, the privilege level passed in this command is normally the same requested maximum privilege level that is used for a subsequent `activate session` command.

MC implementations of IP-based channels must support the `get channel authentication capabilities` command using the IPMI v1.5 packet format. BMCs that support IPMI v2.0 RMCP+ must also support the command using the IPMI v2.0/RMCP+ format.

The `get channel authentication capabilities` command can also be used as a no-op “ping” to keep a session from timing out.

Session header fields request and response data prior when used prior to session activation

authentication type = NONE/payload type = IPMI message

session seq# = null (0’s)

Session ID = null (0’s)

AuthCode = NOT PRESENT

Table 26: Get channel authentication capabilities command request and response data

IPMI request data byte number	Data field		
1	Channel number		
	[7]	1b =	Get IPMI v2.0+ extended data. If the given channel supports authentication but does not support RMCP+ (such as a serial channel), then the response data should return with bit [5] of byte 4 = 0b, byte 5 should return 01h,
		0b =	Backward compatible with IPMI v1.5. Result response data only returns bytes 1:9, bit [7] of byte 3 (authentication type support) and bit [5] of byte 4 returns as 0b, bit [5] of byte 5 returns 00h.
	[6:4]	Reserved	
	[3:0]	Channel number	
		0hBh, Fh =	Channel numbers
		Eh =	Retrieve information for the channel from which this request was issued.
2	Requested maximum privilege level		
	[7:4]	Rreserved	
	[3:0]	Requested privilege level	
		0h =	Reserved
		1h =	Callback level

Table Continued

		2h =	User level
		3h =	Operator level
		4h =	Administrator level
		5h =	OEM proprietary level

IPMI response data byte number	Data field		
1	Completion code		
2	Channel number on which the authentication capabilities is being returned. If the channel number in the request was set to Eh, this returns the channel number for the channel where the request was received.		
3	Authentication type support. Returns the setting of the authentication type enable field from the configuration parameters for the given channel that corresponds to the requested maximum privilege level.		
	[7]	1b =	IPMI v2.0+ extended capabilities available. See Extended capabilities field below.
		0b =	IPMI v1.5 support only.
	[6]	Reserved	
	[5:0]	IPMI v1.5 authentication type(s) enabled for given requested maximum privilege level.	
		All bits:	1b = Supported 0b = Authentication type not available for use
		[5]	OEM proprietary (per OEM identified by the IANA OEM ID in the RMCP ping response)
		[4]	Straight password / key
		[3]	Reserved
		[2]	MD5
		[1]	MD2
		[0]	None

Table Continued

4	[7:6]	Reserved
	[5]	KG status (two-key login status). Applies to v2.0/RMCP+ RAKP authentication only. Otherwise, ignore as reserved.
		0b = KG is set to default (all 0's). User key KUID is used in place of KG in RAKP. (Knowledge of KG not required for activating session.)
		1b = KG is set to non-zero value. (Knowledge of both KG and user password (if not anonymous login) required for activating session.)
	Following bits apply to IPMI v1.5 and v2.0:	
	[4]	Per-message authentication status
		0b = Per-message authentication is enabled. Packets to the MC must be authenticated per authentication type used to activate the session, and the user level authentication setting.
		1b = Per-message authentication is disabled. Authentication type none accepted for packets to the MC after the session has been activated.
	[3]	User level authentication status
		0b = User level authentication is enabled. User level commands must be authenticated per authentication type used to activate the session.
		1b = User level authentication is disabled. Authentication type none accepted for user level commands to the MC.
	[2:0]	Anonymous login status. This parameter returns values that tells the remote console whether there are users on the system that have 'null' usernames. This can be used to guide the way the remote console presents login options to the user.
		[2] 1b = Non-null usernames enabled. (One or more users are enabled that have non-null usernames).
		[1] 1b = Null usernames enabled. (One or more users that have a null username, but non-null password, are presently enabled).
		[0] 1b = Anonymous login enabled. (A user that has a null username and null password is presently enabled).
5	For IPMI v1.5: - Reserved	
	For IPMI v2.0+: - Extended capabilities	
	[7:2]	Reserved

Table Continued

	[1]	1b =	Channel supports IPMI v2.0 connections
	[0]	1b =	Channel supports IPMI v1.5 connections
6:8	OEM ID. IANA enterprise number for OEM/Organization that specified the particular OEM authentication type for RMCP. Least significant byte first. Return 00h, 00h, 00h if no OEM authentication type available.		
9	OEM auxiliary data. Additional OEM-specific information for the OEM authentication type for RMCP. Return 00h if no OEM authentication type available.		

Get Channel Cipher Suites Command

This command can be executed prior to establishing a session with the MC. The command is used to look up what authentication, integrity, and confidentiality algorithms are supported. The algorithms are used in combination as 'Cipher Suites'. This command only applies to implementations that support IPMI v2.0/RMCP+ sessions.

The data is accessed 16-bytes at a time starting from List Index field value of 0 in the request and then repeating the request incrementing the List Index field each time until fewer than 16-bytes of algorithm data (or no algorithm data) is returned in the response, or the maximum List Index value has been reached.

A given Cipher Suite may only be available for establishing a session at a particular maximum privilege level or lower. For example, a Cipher Suite that has a privilege level of 'Admin' can therefore be used for any privilege level, while a privilege level of User can only be used for establish sessions with a Maximum Requested Privilege Level of User or Callback.

Because the authentication algorithm specifies the steps for authenticating the user, it is a necessary part of session establishment. Therefore an authentication algorithm number is required for all Cipher Suites. It is possible that a given Cipher Suite may not specify use of an integrity or confidentiality algorithm. If the Cipher Suite has integrity and/or confidentiality of 'none', then all the same steps for establishing a session are used (open session request/response, RAKP messages) - but the integrity (AuthCode) and confidentiality fields will be absent in packets for that are sent under the session.

Table 27: Get channel cipher suites command request and response data

IPMI Request Data Byte	Data field	
1	Channel Number	
	[7:4]	Reserved
	[3:0]	Channel number 0h-Bh, Fh = Channel numbers Eh = retrieve information for channel this request was issued on.

Table Continued

2	Payload Type. [7:6] - reserved [5:0] - Payload Type number Typically 00h (IPMI). The Payload Type number is used to look up the Security Algorithm support when establishing a separate session for a given payload type.	
3	List Index.	
	[7]	1b = list algorithms by Cipher Suite 0b = list supported algorithms[1]
	[6]	Reserved
	[5:0]	List index (00h-3Fh). 0h selects the first set of 16, 1h selects the next set of 16, and so on. 00h = Get first set of algorithm numbers. The MC returns sixteen (16) bytes at a time per index, starting from index 00h, until the list data is exhausted, at which point it will 0 bytes or <16 bytes of list data.

IPMI Response Data Byte	**Data field**
1	Completion Code
2	Channel Number Channel number that the Authentication Algorithms are being returned for. If the channel number in the request was set to Eh, this will return the channel number for the channel that the request was received on.
(3:18)	Cipher Suite Record data bytes, per Table 22-18, Cipher Suite Record Format. Record data is 'packed'; there are no pad bytes between records. It is possible that record data will span across multiple List Index values. The MC returns sixteen (16) bytes at a time per index, starting from index 00h, until the list data is exhausted, at which point it will 0 bytes or <16 bytes of list data.

[1] When listing numbers for supported algorithms, the MC returns a list of the algorithm numbers for each algorithm that the MC supports on a given channel. Each algorithm is listed consecutively and only listed once. There is no requirement that the MC return the algorithm numbers in any specific order.

Cipher suite records

The data from the Get Channel Cipher Suites command is issued as Cipher Suite records. Tag bits are used to delimit different fields in the record. Each record starts off with a "Start Of Record" byte. This byte can be 30h or 31h, indicating that the Start Of Record byte is followed either by an Cipher Suite ID, or by a OEM Cipher Suite ID plus OEM IANA.

Following the header bytes are algorithm number bytes for the different algorithms that form the Cipher Suite. Each byte is tagged with the type of algorithm the number is for. Cipher Suite records are required to list algorithms in the order: Authentication Algorithm number first, Integrity Algorithm numbers next, and Confidentiality Algorithm numbers last.

If more than one algorithm of a given type is listed in the Cipher Suite Record, then any one of the algorithms can be used in combination with the other types. For example, if a Cipher Suite response returns both MD5 and MD2 as Authentication and Integrity algorithms, and xRC4 for confidentiality, then the allowed combinations are [MD2, MD2, xRC4], [MD2, MD5, xRC4], [MD5, MD2, xRC4], and [MD5, MD5, xRC4]. A remote console can negotiate for those combinations when establishing a session.

Table 28: Cipher suite record format

Size	Tag bits [7:6]	Tag bits [5:0]
2 or 5		This field starts off with either a C0h or C1h "Start of Record" byte, depending on whether the Cipher Suite is a standard Cipher Suite ID or an OEM Cipher Suite, respectively. **Standard cipher suite ID** • Byte 1: [7:0] = 1100_0000b. Start of Record, Standard Cipher Suite. • Byte 2 (Data following C0h (1100_0000b) start of record byte): Cipher Suite ID—This value is used a numeric way of identifying the Cipher Suite on the platform. It's used in commands and configuration parameters that enable and disable Cipher Suites. **OEM cipher suite** • Byte 1: [5:0] = 1100_0001b — Start of Record, OEM Cipher Suite. • Byte 2 (Data following C1h (1100_0001) start of record byte): OEM Cipher Suite ID Byte 3:5 - OEM IANA Least significant byte first. 3-byte IANA for the OEM or body that defined the Cipher Suite.
1	00b	[5:0] = Authentication Algorithm Number. A Cipher Suite is only allowed to utilize one Authentication algorithm.
var	01b	[5:0] = Integrity Algorithm Number(s).
var	10b	[5:0] = Confidentiality Algorithm Number(s).

Cipher suite ID numbers

The following table provides the number ranges and assignments for Cipher Suite IDs. The Cipher Suite ID values are used as a way to identify different Cipher Suites in configuration parameters and IPMI commands.

The OEM IDs do not correspond to a particular Cipher Suite, but are handles that can be used to identify the Cipher Suite on a particular implementation of a MC. I.e. the OEM Cipher Suite corresponding to “80h” can be different from one MC to the next. These handles can, however, be used in configuration parameters and commands the same way as the IPMI-defined Cipher Suite IDs.

The Get Channel Cipher Suites command will return the algorithms used to form a given Cipher Suite (those numbers can then be used by a remote console in the commands for establishing a session). For OEM defined Cipher Suites, the Get Channel Cipher Suites command will also return the IANA for the OEM or body that defined the Cipher Suite.

Table 29: Cipher suite ID numbers

ID	Characteristics	Cipher Suite	Authentication Algorithm	Integrity Algorithm(s)	Confidentiality Algorithm(s)
0	no password	00h, 00h, 00h	RAKP-none	None	None
1	S	01h, 00h, 00h	RAKP-HMAC-SHA1	None	None
2	S, A	01h, 01h, 00h		HMAC-SHA1-96	None
3	S, A, E	01h, 01h, 01h			AES-CBC-128
4	S, A, E	01h, 01h, 02h			xRC4-128
5	S, A, E	01h, 01h, 03h			xRC4-40
6	S	02h, 00h, 00h	RAKP-HMAC-MD5	None	None
7	S, A	02h, 02h, 00h		HMAC-MD5-128	None
8	S, A, E	02h, 02h, 01h			AES-CBC-128
9	S, A, E	02h, 02h, 02h			xRC4-128
10	S, A, E	02h, 02h, 03h			xRC4-40
11	S, A	02h, 03h, 00h		MD5-128	None
12	S, A, E	02h, 03h, 01h			AES-CBC-128
13	S, A, E	02h, 03h, 02h			xRC4-128
14	S, A, E	02h, 03h, 03h			xRC4-40
80h-BFh	OEM specified	OEM specified	OEM specified	OEM specified	OEM specified

Table Continued

ID	Characteristics	Cipher Suite	Authentication Algorithm	Integrity Algorithm(s)	Confidentiality Algorithm(s)
C0h-FFh	reserved	-	-	-	-

Key:

S = Authenticated session setup (correct role, username and password/key required to establish session.)

A = Authenticated payload data supported.

E = Authentication and encrypted payload data supported.

Set session privilege level command

This command is available to the MC.

This command is sent in authenticated format. When a session is activated, the session is set to an initial privilege level. A session that is activated at a maximum privilege level of callback is set to an initial privilege level of callback and cannot be changed. All other sessions are initially set to user level, regardless of the maximum privilege level requested in the `activate session` command. The remote console must raise the privilege level of the session using this command in order to execute commands that require a greater-than-user level of privilege.

This command cannot be used to set a privilege level higher than the lowest of the privilege level set for the user (via the `set user access` command) and the privilege limit for the channel that was set via the `set channel access` command. The specification allows a session to be used across multiple channels. The maximum privilege limit and authentication are based on the user privilege and channel limits. Since these can vary on a per channel basis, an implementation cannot simply assign a single privilege limit to a given session but must authenticate incoming messages according to the specific settings for the channel and the user on a per-channel basis.

Table 30: Set session privilege level command request and response data

IPMI request data byte number	Data field
1	Requested privilege level • [7:4] — Reserved • [3:0] — Privilege level ◦ 0h — No change, just return present privilege level ◦ 1h — Reserved ◦ 2h — Change to user level ◦ 3h — Change to operator level ◦ 4h — Change to administrator level ◦ 5h — Change to OEM proprietary level ◦ All other = Reserved

Table Continued

IPMI response data byte number	Data field
1	Completion code. Generic, plus following command specific: • 80h = Requested level not available for this user • 81h = Requested level exceeds channel and/or user privilege limit • 82h = Cannot disable user level authentication
2	New privilege level (or present level if return present privilege level was selected.)

Close session command

This command is used to immediately terminate a session in progress. It is typically used to close the session that the user is communicating over, though it can be used to terminate other sessions in progress (provided that the user is operating at the appropriate privilege level, or the command is executed over a local channel, such as the system interface).

Table 31: Close session command request and response data

IPMI request data byte number	Data field
1:4	Session ID. For IPMI v2.0/RMCP+ this is the Managed System Session ID value that was generated by the MC, not the ID from the remote console. If Session ID = 0000_0000h then an implementation can optionally enable this command to take an additional byte of parameter data that allows a session handle to be used to close a session.
(5)	Session Handle. Only present if Session ID = 0000_0000h.
IPMI response data byte number	**Data field**
1	Completion code. • 87h = Invalid session ID in request • 88h = Invalid Session Handle in request • 82h = Cannot disable user level authentication

Get session info command

This command is available to the MC.

This command is used to get information regarding which users presently have active sessions, and, when available, addressing information for the party that has established the session. A portion of the response is dependent on the type of channel.

For IPMI v2.0, a previously reserved field has been defined to hold a value indicating whether a session operating on a channel of channel type = 802.3 LAN is presently using IPMI v1.5 or v2.0/RMCP+ protocols.

Table 32: Get session info command request and response data

IPMI request data byte number	Data field
1	Session index. This value is used to select entries in a logical sessions table maintained by the management controller. Information for all active sessions can be retrieved by incrementing the session index from 1 to N, where N is the number of entries in the active sessions table. • 00h = Return information for the active session associated with the session where this command was received. • N = Get information for Nth active session • FEh = Look up session information according to the session handle passed in this request. • FFh = Look up session information according to the session ID passed in this request.
Present if session index = FEh	
2	Session handle. 00h = reserved.
Present if session index = FFh:	
2:5	Session ID. ID of session to look up session information. For IPMI v2.0/RMCP+ this is the session ID value that was generated by the MC, not the ID from the remote console.
IPMI response data byte number	**Data field**
1	Completion code
2	Session handle presently assigned to active session. FFh = reserved. Return 00h if no active session associated with given session index.
3	Number of possible active sessions. This value reflects the number of possible entries (slots) in the sessions table. • [7:6] — Reserved • [5:0] — Session slot count. 1-based.
4	Number of currently active sessions on all channels on this controller. • [7:6] — Reserved • [5:0] — Active session count. 1-based. 0=no currently active sessions.
The following parameters are returned only if there is an active session corresponding to the given session index:	
5	User ID for selected active session. • [7:6] — Reserved • [5:0] — User ID. 000000b = reserved.

Table Continued

6	Operating privilege level • [7:4] — Reserved • [3:0] — Ppresent privilege level at which the user is operating.
7	[7:4] — Session protocol auxiliary data. For channel type = 802.3 LAN: • 0h = IPMI v1.5 • 1h = IPMI v2.0/RMCP+ Channel in which the session was activated. [3:0] - Channel number
The following bytes 8:18 are optionally returned if channel type = 802.3 LAN:	
8:11	IP address of remote console (MS-byte first). Address that was received in the `activate session` command that activated the session
12:17	MAC address (MS-byte first). Address that was received in the `activate session` command that activated the session.
18:19	Port number of remote console (LS-byte first). Port number that was received in UDP packet that held the `activate session` command that activated the session (for IPMI v1.5 packets) or that was used for in the packet for RAKP Message 3 (for IPMI v2.0 / RMCP+ packets).
The following bytes 8:13 are returned if channel type = asynch. serial/modem:	
8	Session/channel activity type: • 0 = IPMI messaging session active • 1 = Callback messaging session active • 2 = Dial-out alert active • 3 = TAP page active
9	Destination selector for active call-out session. 0 otherwise. • [7:4] - Reserved • [3:0] - Destination selector. Destination 0 is always present as a volatile destination that is used with the `alert immediate` command.

Table Continued

10:13	If PPP connection, the IP address of the remote console. (MS-byte first). 00h, 00h, 00h, 00h otherwise.
The following additional bytes 14:15 are returned if channel type = asynch. serial/modem and connection is PPP:	
14:15	Port address of remote console (LS-byte first). Address that was received in the `activate session` command that activated the session.

Get AuthCode command

This command is available to the MC.

This command is used to send a block of data to the MC, whereupon the MC returns a hash of the data together concatenated with the internally stored password for the given channel and user. This command allows a remote console to send an AuthCode and data block to system software on a remote platform, whereby the system software can validate the AuthCode by comparing it with the AuthCode returned by the MC. This enables the MC to serve as a validation agent for remote requests that come through local system software instead of through a remote session directly with the MC.

The following is an outline of potential use of this capability. Remote console software could request that system software perform a particular operation. In response, local system software could deliver a challenge string to the remote console, which would be required to hash it with the desired password and return the AuthCode to the local system software. The local system software would then perform the requested operation only if it found that the AuthCode matched the one returned by the MC. The local software would typically implement mechanisms to bind the challenge string to the requested operation to ensure that the challenge string and AuthCode combination only applied to a given instance of the requested operation, and even from a particular remote console.

- Managed system delivers a random number token, S, to the console. In this example, the console uses S to identify a particular request. The managed system tracks outstanding S values, and expires them either because a valid message was received from a console that used that token, or because the token was not used within a specified interval.
- Console determines: X = data to be authenticated
    - K1 = 16-byte signature of X and a sequence number = hash(X, S, SW_Authentication_Type). Where SW_Authentication_Type is any signature algorithm management software wishes to use for providing a signature given X and S.
    - K2 = 16-byte hash of K1 and the password = hash(K1, PWD, Authentication_Type). Where Authentication_Type in this case is one of the supported authentication types for the given MC.
- Console sends X, S, and K2 to software agents on the managed system.
- Software agent on the managed system calculates K1 from X and S that it received by locally calculating K1=hash1(X, S, SW_Authentication_Type). The software also verifies that S is a valid outstanding token.
- Managed system passes K1 to MC. MC internally looks up password based on the user ID passed in the

  `get authcode`

  command and produces: K2

  BMC

  = hash(K1, PWD, Authentication_Type).
- Managed system accepts data if software agents finds that K2 = K2

  BMC

  .

Table 33: Get AuthCode command request and response data

IPMI request data byte number	Data field
1	[7:6] - Authentication type / Integrity algorithm number • 00b = IPMI v1.5 AuthCode algorithms • 01b = IPMI v2.0/RMCP+ algorithm number For [7:6] = 00b, IPMI v1.5 AuthCode number: • [5:4] - Reserved • [3:0] - Hash type ◦ 0h = Reserved ◦ 1h = MD2 ◦ 2h = MD5 ◦ 3h = Reserved ◦ 4h = Reserved (change from IPMI v1.5). This results in an error completion code. ◦ 5h = OEM proprietary ◦ All other = Reserved For [7:6] = 01b, IPMI v2.0/RMCP+ Integrity algorithm number 5:0] - Integrity algorithm number. The user password is used as the starting key for the Integrity algorithm, instead of session-dependent keys such as the session integrity key. The “none” Integrity number (0) is illegal and results in an error completion code.
2	Channel number • [7:4] - Reserved • [3:0] - Channel number
3	User ID. (Software will typically have to use the `get user name` command to look up the user ID from a user name).
4:19	Data to hash (must be 16 bytes).
IPMI response data byte number	**Data field**
1	Completion code.
For IPMI v1.5 AuthCode number:	

Table Continued

2:17	AuthCode =
For IPMI v2.0 Integrity algorithm number	
(2:21)	Resultant hash, per selected Integrity algorithm. Up to 20 bytes. An implementation can elect to return a variable length field based on the size of the hash for the given integrity algorithm, or can return a fixed field where the hash data is followed by 00h bytes as needed to pad the data to 20 bytes.

Set channel access command

This command is available to the MC.

This command is used to configure whether channels are enabled or disabled, whether alerting is enabled or disabled for a channel, and to set which system modes channels are available under. This configuration is saved in non-volatile storage associated with the MC. The choice of factory default setting for the non-volatile parameters is left to the implementer or system integrator.

The active (volatile) settings can be overwritten to allow run-time software to make temporary changes to the access. The volatile settings are overwritten from the non-volatile settings whenever the system is reset or transitions to a powered off state.

An implementation can elect to provide a subset of the possible access mode options. If a given access mode is not supported, the command-specific completion code 83h (access mode not supported) must be returned.

Table 34: Set channel access command request and response data

Request data byte number	Data field		
1	[7:4] — Reserved [3:0] — Channel number		
2	[7:6]	00b =	Do not set or change channel access.
		01b =	Set non-volatile channel access according to bits [5:0].
		10b =	Set volatile (active) setting of channel access according to bits [5:0].
		11b =	Reserved.
	[5]	PEF alerting enable/disable. This bit globally gates whether PEF alerts can be issued from the given channel. Setting this to enable PEF alerting is a necessary part of enabling alerts for the channel, but for alerts to be generated, the PEF and channel configuration must also be set to enable alerting. Setting this bit to enable does not alter the PEF configuration or the alerting settings in the channel's configuration parameters. For example, if PEF is not configured for generating an alert, enabling PEF alerting with this bit does not change that configuration. Setting this bit to disable blocks PEF -generated alerts regardless of the PEF and channel configuration parameters.	
		0b =	Enable PEF alerting.

Table Continued

	1b =	Disable PEF alerting on this channel. The `alert immediate` command can still be used to generate alerts.
[4]	Per-message authentication enable/disable. This bit is ignored for channels (such as serial/modem) that do not support per-message authentication.	
	0b =	Enable per-message authentication.
	1b =	Disable per-message authentication. Authentication is required to activate any session on this channel, but authentication is not used on subsequent packets for the session.
[3]	User level authentication enable/disable. Optional. Return a CCh invalid data field error completion code if an attempt is made to set this bit, but the option is not supported.	
	0b =	Enable user level authentication. All user level commands are to be authenticated per the authentication type that was negotiated when the session was activated.
	1b =	Disable user level authentication. Allow user level commands to be executed without being authenticated. If the option to disable user level command authentication is accepted, the MC will accept packets with authentication type set to none if they contain user level commands. For outgoing packets, the MC returns responses with the same authentication type that was used for the request.
[2:0]	Access mode for IPMI messaging. (PEF alerting is enabled/disabled separately from IPMI messaging, see bit 5).	
	000b =	Disabled. Channel disabled for IPMI messaging.
	001b =	Pre-boot only. Channel only available when system is in a powered down state or in BIOS before start of boot.
	010b =	Always available. Channel always available for communication regardless of system mode. BIOS typically dedicates the serial connection to the MC.
	011b =	Shared. Same as always available, but BIOS typically leaves the serial port available for software use.
3	Channel privilege level limit. This value sets the maximum privilege level that can be accepted on the specified channel.	
[7:6]	00b =	Do not set or change channel privilege level limit.

Table Continued

		01b =	Set non-volatile privilege level limit according to bits [3:0].
		10b =	Set volatile setting of privilege level limit according to bits [3:0].
		11b =	Reserved.
	[5:4]	Reserved.	
	[3:0]	Channel privilege level limit.	
		0h =	Reserved.
		1h =	Callback level.
		2h =	User level.
		3h =	Operator level.
		4h =	Administrator level.
		5h =	OEM proprietary level.

Response data byte number	Data field		
1	Completion code. Generic, plus the following command-specific completion codes:		
	82h =	Set not supported on selected channel (for example, channel is session-less).	
	83h =	Access mode not supported.	

Get channel access command

This command is available to the MC.

This command is used to return whether a given channel is enabled or disabled, whether alerting is enabled or disabled for the entire channel, and under what system modes the channel can be accessed.

Table 35: Get channel access command request and response data

Request data byte number	Data field		
1	[7:4] — Reserved [3:0] — Channel number		
2	[7:6]	00b =	Reserved

Table Continued

		01b =	Get non-volatile channel access
		10b =	Get present volatile (active) setting of channel access
		11b =	Reserved
	[5:0]	Reserved	
Response data byte number	**Data field**		
1	Completion code. Generic, plus the command-specific completion code:		
	82h =	Command not supported for selected channel (for example, the channel is session-less.)	
2	[7:6]	Reserved.	
	[5]	0b =	Alerting enabled.
		1b =	Alerting disabled.
	[4]	Per-message authentication enable/disable. This bit is unspecified for channels (such as serial/modem) that do not support per-message authentication.	
		0b =	Per message authentication enabled.
		1b =	Per message authentication disabled.
	[3]	User level authentication enable	
		0b =	User level authentication enabled.
		1b =	User level authentication disabled.
	[2:0]	Access mode	
		0h =	Disabled. Channel disabled for communication.
		1h =	Pre-boot only. Channel only available when system is in a powered down state or in BIOS before start of boot.
		2h =	Always available. Channel always available for communication regardless of system mode. BIOS typically dedicates the serial connection to the MC.
		3h =	Shared. Same as always available, but BIOS typically leaves the serial port available for software use.

Table Continued

3	Channel privilege level limit. This value returns the maximum privilege level that can be accepted on the specified channel.	
	[7:4]	Reserved.
	[3:0]	Channel privilege level limit.
		0h = Reserved.
		1h = Callback level.
		2h = User level.
		3h = Operator level.
		4h = Administrator level.
		5h = OEM proprietary level.

Get channel info command

This command returns media and protocol information about the given channel. The channel protocol may vary with changes to the configuration parameters associated with the channel.

Table 36: Get channel info command request and response data

IPMI request data byte number	Data field	
1	[7:4] — Reserved [3:0] — Channel number. Use Eh to get information about the channel from which this command is being executed.	
IPMI response data byte number	**Data field**	
1	Completion code	
2	[7:4]	Reserved
	[3:0]	Actual channel number. This value typically matches the channel number passed in the request, unless the request is for channel E, in which case the response returns the actual channel number.
3	[7:4]	Reserved

Table Continued

	[6:0]	7-bit channel medium type	
4	Channel protocol type:		
	[7:5]	Reserved	
	[4:0]	5-bit channel IPMI messaging protocol type	
5	Session support		
	[7:6]	00b =	Channel is session-less
		01b =	Channel is single-session
		10b =	Channel is multi-session
		11b =	Channel is session-based (return this value if a channel could alternate between single- and multi-session operation, as can occur with a serial/ modem channel that supports connection mode auto-detect)
	Number of sessions that have been activated on the given channel.		
	[5:0]	Active session count. 1-based. 00_0000b = no sessions have been activated on this channel.	

Table Continued

6	Vendor ID (IANA enterprise number) for OEM/organization that specified the channel protocol. Least significant byte first. Returns the IPMI IANA for IPMI-specification defined, non-OEM protocol type numbers other than OEM. The IPMI enterprise number is: 7154 (decimal). This gives the values F2h, 1Bh, 00h for bytes 6 through 8, respectively. This value is returned for all channel protocols specified in this document, including PPP.
9:10	Auxiliary channel info For channel = Fh (system interface): • Byte 1: SMS interrupt type ◦ 00h-0Fh = IRQ 0 through 15, respectively ◦ 10h-13h = PCI A-D, respectively ◦ 14h = SMI ◦ 15h = SCI ◦ 20h-5Fh = System interrupt 0 through 63, respectively ◦ 60h = Assigned by ACPI / Plug 'n Play BIOS ◦ FFh = No interrupt / unspecified ◦ All other = Reserved • Byte 2: Event message buffer interrupt type. See values for byte 1. For OEM channel types: Byte 1:2 = OEM specified per OEM identified by vendor ID. All other channel types: Byte 1:2 = reserved.

Set user access command

This command is available to the MC.

This command is used to configure the privilege level and channel accessibility associated with a given user ID. If this command is not supported, then a single null user (User 1) per channel is assumed and the privilege level and channel access are determined solely by the settings returned by the `get channel access limits` command. If implemented, this command must support at least the null user (User 1). The number of additional users supported is left to the implementer.

NOTE: The limits set using the `set channel access` command take precedence over the `set user access` command settings. That is, if a given channel is limited to user level then all users are limited to user level operation regardless of what their user access levels were set to using the `set user access` command. Changes made to the user access and privilege levels may not take affect until the next time the user establishes a session.

Table 37: Set user access command request and response data

Request data byte number	Data field		
1	[7]	0b =	Do not change any of the following bits in this byte.

Table Continued

	1b =	Enable changing the following bits in this byte.
[6]	User restricted to callback	
	0b =	User privilege limit is determined by the user privilege limit parameter, below, for both callback and non-callback connections.
	1b =	User privilege limit is determined by the user privilege limit parameter for callback connections, but is restricted to callback level for non-callback connections. Thus, a user can only initiate a callback when they call in to the MC, but once the callback connection has been made, the user could potentially establish a session as an operator.
[5]	User link authentication enable/disable (used to enable whether this user's name and password information will be used for link authentication, for example PPP CHAP) for the given channel. Link authentication itself is a global setting for the channel and is enabled/disabled via the serial/modem configuration parameters. • 0b = disable user for link authentication • 1b = enable user for link authentication	

[4]	User IPMI messaging enable/disable (used to enable/disable whether this user's name and password information will be used for IPMI messaging. In this case, IPMI messaging refers to the ability to execute generic IPMI commands that are not associated with a particular payload type. For example, if IPMI messaging is disabled for a user, but that user is enabled for activating the SOL payload type, then IPMI commands associated with SOL and session management, such as
	`get SOL configuration parameters`
	and
	`close session`
	are available, but generic IPMI commands such as
	`get SEL time`
	are unavailable.)
	• 0b = Disable user for IPMI messaging
	• 1b = Enable user for IPMI messaging	

[3:0]	Channel number	
2	User ID • [7:6] — Reserved • [5:0] — User ID. 000000b = Reserved.	
3	User limits	
[7:4]	Reserved	

Table Continued

	[3:0]	User privilege limit. Determines the maximum privilege level that the user is allowed to switch to on the specified channel. 0h = Reserved 1h = Callback 2h = User 3h = Operator 4h = Administrator 5h = OEM proprietary Fh = No access
(4)	User session limit (optional). Sets how many simultaneous sessions can be activated with the username associated with this user. If not supported, the username can be used to activate as many simultaneous sessions as the implementation supports. Return a CCh invalid data field error completion code if an attempt is made to set a non-zero value in this field, but the option is not supported.	
	[7:4]	Reserved
	[3:0]	User simultaneous session limit. 1-based. 0h = only limited by the implementations overall support for simultaneous sessions.
Response data byte number	**Data field**	
1	Completion code. **NOTE:** An implementation does not return an error completion code if the user access level is set higher than the privilege limit for a given channel. To bring attention to this condition, the software must check the channel privilege limits set using the `set channel access` command and provide notification of any mismatch.	

Get user access command

This command is available to the MC.

This command is used to retrieve channel access information and enabled/disabled state for the given user ID. The command also returns information about the number of supported users.

Table 38: Get user access command request and response data

Request data byte number	Data field	
1	[7:4]	Reserved
	[3:0]	Channel number
2	[7:6]	Reserved
	[3:0]	User ID. 000000b = reserved.

Table Continued

Response data byte number	Data field		
1	Completion code. **NOTE:** An implementation does not return an error completion code if the user access level is set higher than the privilege limit for a given channel. To bring attention to this condition, the software must check the channel privilege limits and provide notification of the mismatch.		
2	Maximum number of user IDs. 1-based. Count includes User 1. A value of 1 indicates only User 1 is supported. • [7:6] — Reserved • [5:0] — Maximum number of user IDs on this channel.		
3	Count of currently enabled user IDs (1-based). A value of 0 indicates that all users, including User 1, are disabled. This is equivalent to disabling access to the channel.		
	[7:6]	User ID enable status (for IPMI v2.0 errata 3 and later implementations).	
		00b =	User ID enable status unspecified. (For backward compatibility with pre-errata 3 implementations. IPMI errata 3 and later implementations should return the 01b and 10b responses.)
		01b =	User ID enabled via `set user password` command.
		10b =	User ID disabled via `set user password` command.
		11b =	Reserved.
	[5:0]	Count of currently enabled user IDs on this channel which indicates how many user ID slots are presently in use.	
4	Count of user IDs with fixed names, including User 1 (1-based). Fixed names in addition to User 1 are required to be associated with sequential user IDs starting from User ID 2. • [7:6] — Reserved • [5:0] — Count of user IDs with fixed names on this channel.		
5	Channel access		
	[7]	Reserved	
	[6]	0b =	User access available during call-in or callback direct connection.

Table Continued

	1b =	User access available only during callback connection.
For pre- IPMI v2.0 errata 3 implementations: bits 5:4 are used for determining the count of currently enabled user IDs in byte 3. Either bit being set to 1b represents an enabled user ID. For IPMI v2.0 errata 3 and later implementations: the count of enabled User IDs is based on the user IDs that are presently enabled as reflected in byte 3, bits [7:6], user ID enable status. **NOTE:** Some pre- IPMI v2.0 errata 3 implementations may automatically clear bits [5:4], and may also prevent them from being set, while the user ID is disabled. IPMI v2.0 errata 3 and later implementations should not alter bits [5:4] based on whether or not a user ID is enabled.		
[5]	0b =	User disabled for link authentication
	1b =	User enabled for link authentication
[4]	0b =	User disabled for IPMI messaging
	1b =	User enabled for IPMI messaging
[3:0]	User privilege limit for given channel	
	0h =	Reserved
	1h =	Callback
	2h =	User
	3h =	Operator
	4h =	Administrator
	5h =	OEM proprietary
	Fh =	No access. This value does not add to, or subtract from, the number of enabled user IDs

Set user name command

This command is available to the MC.

This command adds a new user ID. The names are stored as a logical array within non-volatile storage associated with the management controller. Names are stored and retrieved using the user ID as the index into the logical array. There is no configurable name for User ID 1. User ID 1 is reserved for the null user name, User 1. Null user is not supported.

The management controller does not prevent duplicate user names from being enabled for the same channel. It is the responsibility of configuration software to ensure that duplicate user names are not enabled simultaneously for the same channel.

Having duplicate user names does not cause functional problems with the MC because the MC uses the first username match that it finds. However, it could be confusing to the user if they have duplicate user names enabled for a given channel, since only the settings for the first encountered user name would be used by the MC.

This command is highly recommended for session-based channels. It is also recommended that the implementation support multiple users with configurable user names.

Table 39: Set user name command request and response data

Request data byte number	Data field
1	User ID • [7:6] — Reserved • [5:0] — User ID. 000000b = reserved. (User ID 1 is permanently associated with User 1, the null user name).
2:17	User name string in ASCII, 16 bytes, max. Strings with fewer than 16 characters are terminated with a null (00h) character and 00h padded to 16 bytes. When the string is read back using the `get user name` command, those bytes return as 0's.
Response data byte number	**Data field**
1	Completion code.

Get user name command

This command is available to the MC.

This command is used to retrieve user name information that was set using the `set user name` command. Configuration software can use this command to retrieve user names.

Table 40: Get user name command request and response data

Request data byte number	Data field
1	User ID • [7:6] — Reserved • [5:0] — User ID. 000000b = reserved.
Response data byte number	**Data field**
1	Completion code.
2:17	User name string in ASCII, 16 bytes, max. Strings with fewer than 16 characters are terminated with null (00h) characters filling in the remaining bytes. MC does not check to see whether string data is printable or not. Only character that MC interprets is null (00h).

Set user password command

This command is available to the MC.

This command is used to set and change user passwords and to enable and disable user IDs. If no password protection is desired for a given user, the password must be stored as an ASCII null-string. The management controller firmware forces the remaining fifteen bytes to 00h and stores the password as sixteen bytes of 00h.

The password is stored as a 16-byte or 20-byte (for IPMI v2.0/RMCP+) octet string. All values (0-255) are allowed for every byte. The management controller does not check the format or interpret values that are passed in with this command.

Software is allowed to place additional restrictions on what passwords can be entered, in which case it is the responsibility of configuration software and console software to stay in synch with that definition. For example, remote console software could restrict passwords to the printable ASCII character set in order to simplify direct keyboard entry. If this is done, any companion configuration utility should ensure that the user does not configure the managed system with non-printable passwords. Otherwise, it would be possible for the management controller to be configured with passwords that could not be entered via the remote console utility.

Table 41: Set user password command request and response data

Request data byte number	Data field		
1	User ID For IPMI v2.0, the MC supports 20-byte passwords for all supported user IDs that have configurable passwords. The MC maintains an internal tag that indicates whether the password was set as a 16-byte or as a 20-byte password. A 16-byte password can be used in algorithms that call for a 20-byte password. In this case, the 16-byte password is padded with 0's to 20- bytes. The test password operation returns the test failed error completion code if an attempt is made to test a password that was stored as a 20-byte password as a 16-byte password (per password size bit 7), and vice versa. The test password operation can be used to determine whether a password has been stored as 16-bytes or 20-bytes. A password that has been stored as a 20-byte password cannot be used for establishing an IPMI v1.5 session. If it is necessary to configure the same password for both IPMI v2.0 and IPMI v1.5 access, it must be set as a 16-byte password.[1]The password is padded with 0's as necessary for IPMI v2.0 / RMCP+ use.		
	[7]	Password size	
		1b =	Set 20-byte user password/key.
		0b =	Set 16-byte user password/key (IPMI v1.5 backward compatible).
	[6]	Reserved	
	[5:0]	User ID. 000000b = reserved. (User ID 1 is permanently associated with User 1, the null user name).	
2	[7:2]	Reserved	
	[1:0]	Operation	
		00b =	Disable user

Table Continued

	01b =	Enable user
	10b =	Set password
	11b =	Test password. Compares the password data given in the request with the presently stored password and returns an OK completion code if there is a match. Otherwise, an error completion code is returned. (See the completion code description in the response data.)
For password size = 16 bytes:		
3:18	Password data. This is a required, fixed length field when used for the set and test password operations. If the password is entered as an ASCII string, it must be null (00h) terminated and 00h padded if the string is shorter than 16 bytes. This field is not needed if the operation is disable user or enable user. If this field is present for those operations, the MC ignores the data.	
For password size = 20 bytes:		
3:22	Password data. This is a required, fixed length field when used for the set- and test password operations. If the password is entered as an ASCII string, it must be null (00h) terminated and 00h padded if the string is shorter than 20 bytes. This field is not needed if the operation is disable user or enable user. If this field is present for those operations, the MC ignores the data.	
Response data byte number	**Data field**	
1	Completion code. Generic, plus the command-specific completion codes:	
	80h =	Mandatory. Password test failed. Password size is correct, but the password data does not match the stored value.
	81h =	Mandatory. Password test failed. Wrong password size was used.

[1] The same user name can be used with different passwords on different channels. The MC scans user names until it finds the first one that is enabled for a particular channel. Thus, it is possible for a MC implementation to be configured to allow a 20-byte password on one channel, and a 16-byte password on another channel for the same user name. This requires multiple user entries.

RMCP+ support and payload commands

This sections list the commands associated with discovering, enabling, and activating payloads under IPMI v2.0/RMCP+ as well as updates and additions to IPMI commands to support IPMI v2.0/RMCP+ sessions, authentication, and configuration.

NOTE: The following commands remain available for payloads if IPMI messaging payload type is disabled:

- Deactivate payload
- Suspend/resume payload encryption (as defined for given payload)
- Get payload activation status
- Get channel payload version command
- Set session privilege level
- Close session

The following table defines the payload type numbers and ranges for Payload Type Handles.

Table 42: Payload type numbers

Number	Type	Major format version	Minor format version
Standard Payload Types			
0h	IPMI Message	1h	0h
1h	SOL (serial over LAN)	1h	0h
Session Setup Payload Types			
10h	RMCP+ Open Session Request	1h	0h
11h	RMCP+ Open Session Response	1h	0h
12h	RAKP Message 1	1h	0h
13h	RAKP Message 2	1h	0h
14h	RAKP Message 3	1h	0h
15h	RAKP Message 4	1h	0h

Activate payload command

This command is available to the MC.

This command is used for activating and deactivating a payload type under a given IPMI session. The ability to execute this command is determined via the user's privileges as assigned via the `set user payload access` command.

The `activate payload` command may return a port number that is separate from the port number for the session that the command was issued under. In this case, the remote console must establish a session on the port number that the `activate session` command returned. The remote console must then issue the `activate payload` command on that port number in order to actually activate the payload. It is possible that the remote console already had a session active on the given port number. If the privileges associated with that session are sufficient (this is typically the case

unless the remote console activated the session at a privilege level that was lower than the maximum level for the user) the remote console can re-use the existing session and just use the `activate payload` command to activate the new payload type.

BMCs may have limited resources for handling multiple sessions. It is highly recommended that a remote console avoids creating multiple sessions and shares sessions for multiple payloads whenever possible.

The `activate payload` command is only accepted over a channel on which payloads can be activated. For example, the `activate payload` command cannot be executed from the IPMB.

Table 43: Activate payload command request and response data

Request data byte number	Data field
1	• [7:6] — Reserved • [5:0] — Payload type. IPMI message payloads do not need to be explicitly activated. A payload that is required to be launched over a different port than that used to establish the initial IPMI session is only required to support the IPMI commands needed by the particular payload type.
2	Payload instance • [7:4] — Reserved • [3:0] — Payload instance. 1-based. 0h = reserved.
3:6	Auxiliary request data. Additional payload-specific parameters to configure behavior of the payload when it becomes activated. Ignored if no auxiliary data is specified for a given payload type.

Table Continued

For payload type = SOL:

- Byte 1
  - [7] — Encryption activation. The encryption algorithms specified in this document must be used with authentication. The MC returns an error completion code if an attempt is made to activate encryption without also activating authentication.
    - 1b: Activate payload with encryption. All SOL payload data from the MC is encrypted, if encryption was negotiated at the time of session activation.
    - 0b: Activate payload without encryption. MC sends all SOL payload data unencrypted, if that option is allowed. (An SOL configuration parameter allows a system to be configured to require encryption for all SOL transfers).
  - [6] — Authentication activation.
    - 1b: Activate payload with authentication. All SOL payload data from the MC is authenticated, if authentication was negotiated at the time of session activation
    - 0b: Activate payload without authentication. MC sends all SOL payload data unauthenticated, if that option is allowed. (An SOL configuration parameter allows a system to be configured to require authentication for all SOL transfers).
  - [5] — Test mode (optional). Enables DCD and DSR to be manually controlled by the remote console and the reporting of RTS and DTR state via the SOL operation/status byte. This can be used to facilitate software testing of the 16550 UART interface.
    - 1b = Activate test mode. If test mode is not supported, bit [0] of the auxiliary response data will be returned as 0b.
    - 0b = Deactivate test mode.
  - [4] — Reserved
  - [3:2] — Shared serial alert behavior. The following settings determine what happens to serial alerts if IPMI over serial and SOL are sharing the same baseboard serial controller.
    - 11b: Reserved
    - 10b: Serial/modem alerts succeed while SOL active.
    - 01b: Serial/modem alerts deferred while SOL active.
    - 00b: Serial/modem alerts fail while SOL active.
  - [1] — SOL startup handshake
    - 0b: MC asserts CTS and DCD/DSR to baseboard upon activation.
    - 1b: CTS and DCD/DSR remain deasserted after activation. Remote console must send an SOL payload packet with control field settings to assert CTS and DCD/DSR. (This enables the remote console to first alter volatile configuration settings before hardware handshake is released).
  - [0] — Reserved
- Byte 2:4 — Reserved, write a 00h

Table Continued

Response data byte number	Data field
1	Completion code. Generic plus the command-specific completion codes: (An error completion code should be returned if the payload type in the request is set to IPMI Message (0h)). • 80h: Payload already active on another session (required). This will be returned any time an attempt is made to activate a payload type when that type is already activated for another session, and when the MC only supports one instance of that payload type running at a time. • 81h: Payload type is disabled (optional). Given payload type is not configured to be enabled for activation. • 82h: Payload activation limit reached. Cannot activate given payload type because the maximum number of simultaneous instances of that payload type are already running. • 83h: Cannot activate payload with encryption. • 84h: Cannot activate payload without encryption. MC requires encryption for all payloads for given privilege level.
2:5	Auxiliary response data. LS-byte first. For payload = SOL: • [31:1] — Reserved. Return as 0's (zeroes). • [0] ◦ 0b = Test mode not supported / enabled. ◦ 1b = Test mode enabled.
6:7	Inbound payload size. Maximum size of payload data field from remote console to MC. Excludes size of confidentiality header and trailer fields, if any. 1-based.
8:9	Outbound payload size. Maximum size of payload data field from MC to remote console. Excludes size of confidentiality header and trailer fields, if any. 1-based.
10:11	Payload UDP port number. UDP port number through which the payload can be transferred. If the port number is the same as the port that was used to establish the IPMI session, then SOL payload transfers are now available under that IPMI session on that port. Otherwise, the remote console needs to establish a separate IPMI session to the specified port number using the same IP address, username and password/key information that was used to establish the IPMI session. SOL payload transfers are then available over that session. If the remote console already has an IPMI session established on that port for a different payload type, the SOL payload type is also available over that session - provided that the session was established at a privilege level that matches the privilege level and authentication required for SOL. Otherwise, the remote console needs to close that session and reestablish it at the necessary privilege level.
12:13	Payload VLAN number - FFFFh if VLAN addressing is not used.

Deactivate payload command

This command is available to the MC.

This command is used to terminate use of a given payload on an IPMI session. This type of traffic then becomes freed for activation by another session, or for possible re-activation under the present session. The `deactivate payload` command does not cause the session to be terminated. The `close session` command should be used for that

purpose. A remote console application does not need to explicitly deactivate payload(s) before terminating a session. When a session terminates, all payloads that were active under that session are automatically deactivated by the MC.

Table 44: Deactivate payload command request and response data

Request data byte number	Data field
1	• [7:6] — Reserved • [5:0] — Payload type.
2	Payload instance • [7:4] — Reserved • [3:0] — Payload instance. 1-based. 0h = reserved.
3:6	Payload auxiliary data. Additional parameters to configure behavior of the payload when it becomes deactivated. Ignored if no auxiliary data is specified for given payload type. For payload type = SOL, (no auxiliary data) write as 0000_0000h:
Response data byte number	**Data field**
1	Completion code. Generic plus the command-specific completion codes: (An error completion code should be returned if the payload type in the request is set to “IPMI Message” (0h)). • 80h: Payload already deactivated. • 81h: Payload type is disabled (optional). Given payload type is not configured to be enabled for activation.

Suspend/resume payload encryption command

This command enables a remote console to control whether payload data from the MC is sent encrypted or not. Since encryption can be a significant burden on software, this command provides a mechanism to allow higher performance by operating without encryption and only activating encryption when it is required for data confidentiality. The command can also trigger a regeneration of the encryption Initialization Vector and re-initialization of the encryption state machine for algorithms such as xRC4 that use the same initialization vector for multiple packets.

The extent at which this command can control encryption of data from the MC is dependent on the payload definition. Some payload definitions may use a mix of encrypted and unencrypted payload data transfers. For example, a payload may implement a ‘request/response’ protocol, where the MC would return an encrypted or unencrypted response based on whether the request from the remote console was encrypted or unencrypted. In this case, the command may only affect data that is autonomously generated by the MC. Other payload definitions may just use whatever encryption the session was activated with, and offer no ‘run-time’ control of encryption/decryption, while other payload definitions may be ‘stream based’ where it is desirable for the remote console to be able to select when payload data is from the MC is encrypted or not.

The Suspend/Resume Payload Encryption command is only accepted from the channel that the payload was activated on.

Table 45: Payload-specific encryption behavior

Payload Type = IPMI Messaging
• Encrypted requests from the remote console will get encrypted responses from the MC. • The Suspend/Resume Payload Encryption command controls whether asynchronous (unrequested) messages from the MC are encrypted or not. • PET Traps (which are actually separate from IPMI Messaging) are always sent unencrypted.
Payload Type = SOL
• The SOL configuration parameters allow configuring the system to require that SOL data be encrypted. • The MC will transmit SOL payload data according to encryption settings that were selected when the payload was activated unless over-ridden by SOL configuration parameters. • The Suspend/Resume Payload Encryption command controls whether SOL Payload data is encrypted or not.

Table 46: Suspend/resume payload command request and response data

IPMI request data byte number	Data field
1	[7:6] - reserved [5:0] - payload type (See Table 13-16, Payload Type Numbers)
2	Payload Instance [7:4] - reserved [3:0] - payload instance. 1-based. 0h = reserved.
3	[7:2] - reserved [4:0] - Operation • 2h = Regenerate initialization vector. For xRC4 encryption, this causes the MC to reinitialize the xRC4 state machine, reset the data offset, and deliver a new Initialization Vector value in the next encrypted packet it sends to the remote console. Because of processing delays and potential tasks in progress, the remote console may receive additional packets from the MC that are encrypted using the prior Initialization Vector before getting packets that use the new IV. • 1h = Resume/Start encryption on all transfers of specified payload data from the MC. • 0h = Suspend encryption on all transfers of specified payload messages from the MC.

Table Continued

IPMI response data byte number	Data field
1	Completion Code Generic plus the following command-specific completion codes: • 80h: Operation not supported for given payload type. • 81h: Operation not allowed under present configuration for given payload type. • 82h: Encryption is not available for session that payload type is active under. • 83h: The payload instance is not presently active.

Set channel security keys command

The Set Channel Security Keys command provides a standardized interface for initializing system unique keys that are used for the pseudo-random number generator key (KR) and the key-generation key (KG) used for RMCP+. Implementing the ability to set Kr is optional. The command is provided mainly to offer a common interface for BMCs that are not pre-configured with a KR values, or which may need their KR values to be restored if they are lost due to a data corruption or firmware update.

The command includes a mechanism that allows specified keys to be “locked”. Once locked, the key value cannot be read back or rewritten via standard IPMI commands. It is possible, however, that a firmware update or re- installation procedure may cause the keys to be cleared or unlocked. Software utilities responsible for MC initial installation and setup should check to see whether keys have been locked and if not, should initialize them appropriately and lock them.

If this command is not supported, it indicates that the keys are either permanently pre-configured, or that they are only configurable via an OEM/MC-specific mechanism.

Request data byte number	Data field
1	Channel Number [7:4] - reserved [3:0] - Channel Number **NOTE:** This command only applies to channels that support RMCP+, if the channel does not support RMCP+ the command will return an error completion code.

Table Continued

2	Operation [7:2] - reserved [1:0] - Operation • 00b = read key MC returns value of specified key, provided key has not yet been locked. Some BMCs may allow the key to be re-written if it does not match the expected value. Other BMCs may only allow one 'set' operation. If the key value has not yet been initialized, the MC will return 0's for the key value. Utility software responsible for MC installation and initial setup can use this operation to also check to see whether keys have been initialized and locked. • 01b = set key MC writes given key value to non-volatile storage. • 10b = lock key MC locks out modification or reading the key value. Once a key has been locked, it is not cannot be rewritten or read via IPMI specified commands. • 11b = reserved
3	Key ID [7:0] - key ID. • 00h = RMCP+ "KR" key (20 bytes). The "KR" key is used as a unique value for random number generation. Note: A MC implementation is allowed to share a single KR value across all channels. A utility can set KR and lock it for one channel, and then verify it has been set and locked for any other channels by using this command to read the key from other channels and checking the 'lock status' field for each channel to see if it matches and is locked. • 01h = RMCP+ "KG" key (20 bytes). "KG" key acts as a value that is used for key exchange for the overall channel. This key is cannot be locked, to ensure a password/key configuration utility can set its value. This value is used in conjunction with the user key values (passwords) in RAKP-HMAC- SHA1 and RAKP-HMAC-MD5 authentication. I.e. the remote console needs to have a-priori knowledge of both this key value and the user password setting, in order to establish a session. KG must be individually settable on each channel that supports RMCP+. • All other = reserved
(4:M)	Key value. Value for specified key. Used for "set" Operation only. Otherwise, this field is not used in the request. The MC will ignore any bytes following the 'Key ID' byte.

Response data byte number	**Data field**

Table Continued

1	Completion Code. Generic, plus following command-specific completion codes: • 80h = Cannot perform set / confirm. Key is locked (mandatory) • 81h = insufficient key bytes • 82h = too many key bytes • 83h = key value does not meet criteria for specified type of key • 84h = KR is not used. MC uses a random number generation approach that does not require a KR value.
2	7:2 - reserved. 1:0 - lock status • 00b = key is not lockable. • 01b = key is locked. • 10b = key is unlocked. • 11b = reserved
(3:N)	Key value. The MC returns the specified key value when the Operation is set to “read key”. Otherwise, the MC returns no additional bytes past the completion code.

Get system interface capabilities command

This command can be used to determine whether the SSIF supports multi-part transactions, and what size of IPMI messages can be transferred. The Get System Interface Capabilities command is mandatory for BMCs that implement multi-part writes or reads. Thus, software can assume that if the Get System Interface Capabilities command is not implemented, the interface only supports single-part writes and reads.

Request data byte number	Data field
1	System Interface Type [7:4] - reserved [3:0] - System Interface Type (For BT use the Get BT Interface Capabilities command) • 0h = SSIF • 1h = KCS • 2h = SMIC • all other = reserved

Response data byte number	Data field

Table Continued

1	Completion Code
2	Reserved. Returned as 00h.
For System Interface Type = SSIF:	
3	[7:6] - Transaction support • 00b = only single-part reads/writes supported. • 01b = multi-part reads/writes supported. Start and End transactions only. • 10b = multi-part reads/writes supported. Start, Middle, and End transactions supported. • 11b = reserved. [5:4] - reserved. [3] - PEC support. • 1b = implements PEC. MC will start using PEC in read transactions after it receives any SSIF write transaction that includes a valid PEC. The MC ceases using PEC if it receives an SSIF write transaction that does not include PEC. • 0b = does not support PEC. Note that a MC implementation may reject write transactions that include a PEC byte. [2:0] - SSIF Version 000b = version 1 (version defined in this specification).
4	Input message size in bytes. (1 based.) Number of bytes of IPMI message data that the MC can accept. This number does not include slave address, SMBus length , PEC, or SMBus CMD bytes, just the IPMI message data. A MC that just supports single-part writes would return 32 (20h) for this value. A MC that supports multi-part Start and End would return a value from 33 to 64. A MC that supports multi-part with Middle transactions would return a value from 65 to 255.
5	Output message size in bytes. (1 based.) Maximum number of bytes of IPMI message data that can be read from the MC. This number does not include slave address, SMBus length, PEC, SMBus CMD bytes, special bytes (such as the special bytes following the length byte in the multi -part read middle and end transactions) just the IPMI message data. A MC that just supports single-part reads would return 20h (32) for this value. A MC that supports multi-part Start and End would return a value from 33 to 62 (the reason this is 62 instead of 64 is that there are two special bytes after the length byte.) A MC that supports multi-part with Middle transactions would return a value from 63 to 255.
For System Interface Type = KCS or SMIC	

Table Continued

3	[7:3] - reserved [2:0] - System Interface Version 000b = version 1 (conformant with KCS or SMIC interface as defined in this specification).	
4	Input maximum message size in bytes. (1 based.) Largest number of bytes that can be transferred in a KCS FFh means 255 or more.	

Get payload activation status command

This command is available to the MC.

This command returns how many instances of a given payload type are presently activated, and how many total instances can be activated.

Table 47: Get payload activation status command request and response data

Request data byte number	Data field		
1	Payload type number - number of the standard payload type or OEM payload handle from which to retrieve status.		
Response data byte number	**Data field**		
1	Completion code		
2	Instance capacity		
	[7:4]	Reserved.	
	[3:0]	Number of instances of a given payload type that can be simultaneously activated on MC. 1-based. 0h = reserved.	
3	[7]	1b =	Instance 8 is activated.
		0b =	Instance 8 is deactivated.
	[6]	1b =	Instance 7 is activated.
		0b =	Instance 7 is deactivated.
	…		
	[0]	1b =	Instance 1 is activated.
		0b =	Instance 1 is deactivated.

Table Continued

4	[7]	1b =	Instance 16 is activated.
		0b =	Instance 16 is deactivated.
	[6]	1b =	Instance 15 is activated.
		0b =	Instance 15 is deactivated.
	…		
	[0]	1b =	Instance 9 is activated.
		0b =	Instance 9 is deactivated.

Get payload instance info command

This command is available to the MC.

This command returns information about a specific instance of a payload type. It is primarily used by software that may want to negotiate with an application that is presently using the given payload type. It accomplishes this by using the session ID returned from this command with the `get session info` command to look up the addressing information for the party that activated the payload. The application may then use that information to establish a direct dialog with the application that presently owns the payload (this inter-application communication is not defined in the IPMI specifications).

Table 48: Get payload instance info command request and response data

Request data byte number	Data field
1	Payload type number - number of the standard payload type or OEM payload handle from which to retrieve status.
2	Payload instance. 1-based. 0h = reserved.
Response data byte number	**Data field**
1	Completion code. An error completion code should be returned if the payload type in the request is set to IPMI message (0h) .

Table Continued

2:5	Session ID - ID of session on which the instance is presently activated. (The managed system session ID that the MC generated when the session was activated). 00_00_00_00h if the given instance is not activated. Remote software can use this information with the `get session info` command to identify the remote console that presently is using a given payload type.
6:13	Payload-specific information (8-bytes) For payload type = SOL: • Byte 1: Port number, a number representing the system serial port that is being redirected. 1-based. 0h = unspecified. Used when more than one port can be redirected on a system. • Byte 2: 8 = reserved.

Set user payload access command

This command is available to the MC.

This command controls whether the specified user has the ability to activate the specified payload type on the given channel. The command uses bitfields to allow a configuration utility to use a single command to set enable/disable multiple payloads at a time. Standard payloads are set separately from OEM payload enables. The command would be issued at least once with standard payloads selected to set the configuration for standard payloads, and then at least once with OEM payloads selected to set the configuration for OEM payloads.

Table 49: Set user payload access command request and response data

Request data byte number	Data field
1	Channel number • [7:4] — Reserved • [3:0] — Channel number
2	[7:6] - Operation • 00b = Enable. Writing a “1b” to enable/disable bit enables the corresponding payload. Writing “0b” to bit causes no change to enabled/disabled state. • 01b = Disable. Writing a “1b” to bit disables the corresponding payload. Writing ”0b” to bit causes no change to enabled/disabled state. • 10b, 11b = Reserved [5:0] — User ID. 000000b = reserved.

Table Continued

3	Standard payload enables 1 • [7:2] — Reserved for standard payloads 2-7 enable/disable bits. • [1] — Standard payload 1 (SOL) enable/disable • [0] — Reserved. IPMI messaging is enabled/disabled for users via the `set user access` command.
4	Standard payload enables 2 - reserved
5	OEM payload enables 1 • [7] - OEM payload 7 enable/disable • [6] - OEM payload 6 enable/disable • [5] - OEM payload 5 enable/disable • [4] - OEM payload 4 enable/disable • [3] - OEM payload 3 enable/disable • [2] - OEM payload 2 enable/disable • [1] - OEM payload 1 enable/disable • [0] - OEM payload 0 enable/disable
6	OEM payload enables 2 - reserved

Response data byte number	**Data field**
1	Completion code. An implementation will not return an error completion code if the user access level is set higher than the privilege limit for a given channel. If it is desired to bring attention to this condition, it is up to software to check the channel privilege limits set using the `set channel access` command and provide notification of any mismatch.

Get user payload access command

This command is available to the MC.

The `get user payload access` command returns the user payload enable settings that were set using the `set user payload access` command.

Table 50: Get user payload access command request and response data

Request data byte number	Data field
1	Channel number • [7:4] — Reserved • [3:0] — Channel number
2	User ID • [7:6] — Reserved • [5:0] - User ID. 000000b = reserved
Response data byte number	**Data field**
1	Completion code
2	Standard payload enables 1 • [7:2] — Reserved for standard payloads 2-7 enabled/disabled state. • [1] ◦ 1b = Standard payload 1 enabled (SOL) ◦ 0b = Standard payload 1 disabled • [0] — Reserved.
3	Standard payload enables 2 - reserved
4	OEM payload enables 1. For each bit: 1b = payload enabled, 0b = payload disabled. • [7] - OEM payload 7 enabled/disabled • [6] - OEM payload 6 enabled/disabled • [5] - OEM payload 5 enabled/disabled • [4] - OEM payload 4 enabled/disabled • [3] - OEM payload 3 enabled/disabled • [2] - OEM payload 2 enabled/disabled • [1] - OEM payload 1 enabled/disabled • [0] - OEM payload 0 enabled/disabled
5	OEM payload enables 2 - reserved

Get channel payload support command

This command is available to the MC.

This command enables local and remote console software to determine what payloads are enabled on the given MC. The command returns a bitfield indicating which payload type numbers can be activated on the given channel.

Table 51: Get channel payload support command request and response data

Request data byte number	Data field
1	Channel number • [7:4] — Reserved • [3:0] — Channel number
Response data byte number	**Data field**
1	Completion code
2	• [7] = Standard payload type #7 supported • … • [0] = Standard payload type #0 supported
3	• [7] = Standard payload type #15 (0Fh) supported • … • [0] = Standard payload type #8 supported
4	• [7] = Session setup payload type #7 supported • … • [0] = Session setup payload type #0 supported
5	• [7] = Session setup payload type #15 (0Fh) supported • … • [0] = Session setup payload type #8 supported
6	• [7] = Payload type 27h (OEM7) used • … • [0] = Payload type 20h (OEM0) used
7	• [7] = Payload type 2Fh (OEM15) used • … • [0] = Payload type 28h (OEM8) used
8:9	Reserved. Return as 0000h

Get channel payload version command

This command is available to the MC.

This command returns version information for the given payload type. The version number has major and minor parts. The major part of the version should only increment when there are significant changes to the payload format, commands, or payload-specific protocols that break backward compatibility with earlier versions. The minor part of the version increments when there are extensions to the payload format that are significant but are backwards compatible with earlier versions under the same major version number. An example of a major change would be a change to the payload activation process that would prevent earlier applications from activating the given payload type. An example of a minor format version change would be the definition of commands for new functions that did not exist under the previous format, but if unused, do not interfere with the operation of older applications.

Table 52: Get channel payload version command request and response data

Request data byte number	Data field
1	Channel number • [7:4] — Reserved • [3:0] — Channel number
2	Payload type number/payload type handle - number of the standard payload type or OEM payload handle for which to retrieve status. See **RMCP+ support and payload commands**. .
Response data byte number	**Data field**
1	Completion code. Generic plus command-specific completion code: 80h — Payload type not available on given channel.
2	Format version • [7:4] - Major format version. BCD encoded (0 to 9). • [3:0] - Minor format version. BCD encoded (0 to 9). Software should present version data to the user in the format “major.minor”. For example, 10h —>1.0. The format version for the SOL payload implemented per this specification is 1.0 (10h).

IPMI LAN Device Commands

This section defines the configuration and control commands that are specific to LAN channels. None of the commands in the following table are required unless a LAN channel is implemented. See **iLO Command number assignments and privilege levels** for the specification of the Network Function and Command (CMD) values and privilege levels for these commands.

Set LAN configuration parameters command

This command is used for setting parameters such as the network addressing information required fro IPMI LAN-operation.

Table 53: Set LAN configuration parameters command request and response data

Request data byte number	Data field
1	Channel number • [7:4] — Reserved • [3:0] — Channel number
2	Parameter selector
3:N	Configuration parameter data, per the table.
Response data byte number	**Data field**
1	Completion code. • 80h = parameter not supported. • 81h = attempt to set the 'set in progress' value (in parameter #0) when not in the 'set complete' state. (This completion code provides a way to recognize that another party has already 'claimed' the parameters.) • 82h = attempt to write read-only parameter. • 83h = attempt to read write-only parameter.

Get LAN configuration parameters command

This command is used for retrieving the configuration parameters from the `set LAN configuration parameters` command.

Table 54: Get LAN configuration parameters command request and response data

Request data byte number	Data field
1	[7] • 0b = Get parameter • 1b = Get parameter revision only [6:4]—Reserved [3:0]—Channel number
2	Parameter selector
3	Set Selector. Selects a given set of parameters under a given Parameter selector value. 00h if parameter does not use a Set Selector.

Table Continued

4	Block Selector (00h if parameter does not require a block number)

Response data byte number	Data field
1	Completion Code. Generic codes, plus following command-specific completion code: 80h = parameter not supported.
2	[7:0]—Parameter revision. Format: MSN = Present revision. LSN = Oldest revision with which the parameter is backward compatible. 11h for parameters in this specification.
The following data bytes are not returned when the 'get parameter revision only' bit is 1b.	
3:N	Configuration parameter data, per the *LAN configuration parameters* table. If the rollback feature is implemented, the MC makes a copy of the existing parameters when the 'set in progress' state becomes asserted (see the Set In Progress parameter #0). While the 'set in progress' state is active, the MC returns data from this copy of the parameters, plus any uncommitted changes that were made to the data. Otherwise, the MC returns parameter data from nonvolatile storage.

Table 55: LAN configuration parameters

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
Set In Progress (volatile)	0	**data 1**—This parameter is used to indicate when any of the following parameters are being updated, and when the updates are complete. The bit is primarily provided to alert software that some other software or utility is in the process of changing the data. An implementation can also elect to provide a rollback feature that uses this information to decide whether to roll back to the previous configuration information, or to accept the configuration change. If used, the rollback feature restores all parameters to their previous state. Otherwise, the change takes effect when the write occurs. [7:2]—reserved [1:0] • 00b = set complete. If a system reset or transition to powered down state occurs while 'set in progress' is active, the BMC will go to the 'set complete' state. If a rollback is implemented, going directly to 'set complete' without first doing a 'commit write' causes pending write data to be discarded. • 01b = set in progress. This flag indicates that some utility or other software is presently doing writes to parameter data. It is a notification flag only, it is not a resource lock. The BMC does not provide an interlock mechanism to prevent other software from writing parameter data. • 10b = commit write (optional). This value is used if a rollback is implemented. The BMC saves data written since the last 'set in progress' state and then goes to the 'set in progress' state. If this option is not supported, an error completion code is returned. • 11b = reserved iLO Notes: • iLO does not support the commit write or rollback features. • iLO will not reset upon transition to the Set Complete state. If a change requires an iLO reset, the reset must be initiated by a separate command sequence. For more information, see parameter 224. • Using this parameter as a configuration semaphore or interlock is only valid for the IPMI interface. It is not integrated with other iLO user interfaces (UIs) that can change the network configuration. To guarantee that no iLO UI modified the network configuration during a certain time frame, use the configuration status counters defined in parameter 224.
Authentication Type Support (Read Only)	1	This read-only field returns the Authentication Types (algorithms) that can be enabled for the given channel. The following

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
		Authentication Type Enables parameter selects which Authentication Types are available when activating a session for a particular maximum privilege level. [7:6]—Reserved [5:0]—Authentication types enabled for this channel (bitfield): All bits: • 1b = Supported • 0b = Authentication type not available for use. [5]—OEM proprietary (per OEM identified by the IANA OEM ID in the RMCP Ping Response) [4]—Straight password / key [3]—Reserved [2]—MD5 [1]—MD2 [0]—none iLO Notes: iLO reports not available for use, all types.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
Authentication Type Enables	2	This field is used to configure which authentication types are available when a remote console activates an IPMI messaging connection to the MC for a requested maximum privilege level. Once the session has been activated, the accepted authentication type is the only one used for authenticated packets, regardless of the present operating privilege level, or the privilege level associated with the command. Depending on configuration of per-message and user-level authentication disables, unauthenticated packets (authentication type = none) may also be accepted. The BMC makes no attempt to check or ensure that stricter authentication types are associated with higher requested maximum privilege levels. For example, it is possible to configure the BMC so activating a session with a maximum privilege level of 'User' requires MD5 while 'Admin' requires 'none'. **NOTE:** An implementation that has fixed privilege and authentication type assignments, in which case this parameter can be implemented as Read Only. It is recommended that an implementation that implements a subset of the possible authentication types returns a CCh error completion code if an attempt is made to select an unsupported authentication type. • **byte 1** Authentication Types returned for maximum requested privilege = Callback level. ◦ [7:6]—Reserved ◦ [5:0]—Authentication types enabled for this channel (bitfield): ◦ All bits: – 1b = Authentication type enabled for use at given privilege level – 0b = Authentication type not available for use at given privilege level. ◦ [5]—OEM proprietary (per OEM identified by the IANA OEM ID in the RMCP Ping Response) ◦ [4]—straight password / key ◦ [3]—reserved ◦ [2]—MD5 ◦ [1]—MD2 ◦ [0]—none • **byte 2** Authentication Types for maximum privilege = User level

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
		(format follows byte 1) • **byte 3** Authentication Type (s) for maximum privilege = Operator level (format follows byte 1) • **byte 4** Authentication Type (s) for maximum privilege = Administrator level (format follows byte 1) • **byte 5** Authentication Type (s) for maximum privilege = OEM level (format follows byte 1) iLO Notes: • iLO responds with all 0s on read. • iLO treats this parameter as read-only.
IP Address	3	**data 1:4**—IP Address MS-byte first. iLO Notes: • If iLO is configured for DHCPv4 address assignment when this parameter is read, it reports the address currently assigned by the DHCPv4 server. • If iLO is configured for DHCPv4 address assignment when this parameter is written, iLO will transition automatically to static address assignment. Be sure to also configure iLO Subnet Mask (parameter 6) and iLO Default Gateway Address (parameter 12) if desired before setting iLO in this case. • Requires iLO reset after write to take effect.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
IP Address Source	4	**data 1** [7:4]—Reserved [3:0]—Address source • 0h = Unspecified • 1h = Static address (manually configured) • 2h =Address obtained by MC running DHCP • 3h = Address loaded by BIOS or system software • 4h = Address obtained by MC running other address assignment protocol iLO Notes: • iLO supports only sources 1h and 2h. • Transitioning iLO from DHCP to static will cause the currently assigned DHCPv4 address if any, to be set as a static address. This address may be then modified if desired by setting parameter 3 IP Address before resetting iLO. • Caution: If no DHCPv4 address has been assigned at the time iLO is transitioned from DHCP to static, iLO static address will be 0.0.0.0. If no static address is subsequently assigned using parameter 3 before iLO reset, iLO will be unreachable through IPv4 after reset. • Requires iLO reset after write to take effect.
MAC Address (can be Read Only)	5	**data 1:6**—MAC Address for messages transmitted from MC. MS-byte first. An implementation can either allow this parameter to be configurable, or it can be implemented as read-only. iLO Notes: This parameter is read-only on iLO.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
Subnet Mask	6	**data 1:4**—Subnet Mask. MS-byte first. iLO Notes: • iLO supports only CIDR subnet masks. For example, contiguous prefix bits only. • If iLO is configured for DHCPv4 address assignment when this parameter is read, it reports the subnet mask currently assigned by the DHCPv4 server. • If iLO is configured for DHCPv4 address assignment when this parameter is written, iLO will transition automatically to static address assignment. Be sure to also configure iLO Static IP Address (parameter 3) and iLO Default Gateway Address (parameter 12) if desired before resetting iLO in this case. • Requires iLO reset after write to take effect.

BMC-generated ARP control (optional)	10	**data 1**—BMC-generated ARP control.
		
		Note: the individual capabilities for BMC-generated ARP responses and BMC-generated Gratuitous ARPs are individually optional. The BMC should return an error completion code if an attempt is made to enable an unsupported capability.
		
		• [7:2]—Reserved
		  ◦ [1]
		  – 1b = enable BMC-generated ARP responses
		  – 0b = disable BMC-generated ARP responses
		  ◦ [0]
		  – 1b = enable BMC-generated Gratuitous ARPs
		  – 0b = disable BMC-generated Gratuitous ARPs
		
		iLO Notes:
		
		Only supported as read-only. Always responds BMC-generated ARP enabled, and Gratuitous ARPs disabled.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
Default Gateway Address	12	**data 1:4**—IP Address MS-byte first. This is the address of the gateway (router) used when the BMC sends a message or alert to a party on a different subnet than the one the BMC is on. iLO Notes: • If iLO is configured for DHCPv4 gateway address assignment when this parameter is read, it reports the address currently assigned by the DHCPv4 server. • If iLO is configured for DHCPv4 gateway address assignment when this parameter is written, it will be transitioned to a static gateway address assignment. • It is possible to configure a static default gateway address on iLO while IP Address and Subnet Mask are configured through DHCPv4. • Requires iLO reset after write to take effect.
Community String	16	**data 1:18**—Community String Default = 'public'. Used to fill in the 'Community String' field in a PET Trap. This string may optionally be used to hold a vendor-specific string that is used to provide the network name identity of the system that generated the event. Printable ASCII string-. If a full 18 non-null characters are provided, the last character does not need to be a null. 18 characters must be written when setting this parameter, and 18 will be returned when this parameter is read. The null character, and any following characters, will be ignored when the Community String parameter is placed into the PET. The BMC will return whatever characters were written. For example, it will not set bytes following the null to any particular value.
Number of Destinations (Read Only)	17	**data 1**—Number of LAN Alert Destinations supported on this channel. (Read Only). At least one set of nonvolatile destination information is required if LAN alerting is supported. Additional nonvolatile destination parameters can optionally be provided for supporting an alert 'call down' list policy. A maximum of 15 (1h to Fh) nonvolatile destinations are supported in this specification. Destination 0 is always present as a volatile destination that is used with the Alert Immediate command. [7:4]—Reserved. [3:0]—Number LAN Destinations. A count of 0h indicates that LAN Alerting is not supported. iLO Notes: iLO Supports 15 nonvolatile destinations.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
Destination Type (volatile / nonvolatile - see description)	18	Sets the type of LAN Alert associated with the given destination. This parameter is not present if the Number of Destinations parameter is 0. • **data 1**—Set Selector = Destination selector, 0 based. ◦ [7:4]—Reserved ◦ [3:0]—Destination selector. Destination 0 is always present as a volatile destination that is used with the Alert Immediate command. • **data 2**—Destination Type ◦ [7]—Alert Acknowledge. – 0b = Unacknowledged. Alert is assumed successful if transmission occurs without error. This value is also used with Callback numbers. – 1b = Acknowledged. Alert is assumed successful only if acknowledged is returned. Note, some alert types, such as Dial Page, do not support an acknowledge. ◦ [6:3]—Reserved ◦ [2:0]—Destination Type – 000b = PET Trap destination – 001b–101b = Reserved – 110b = OEM 1 – 111b = OEM 2 • **data 3**—Alert Acknowledge Timeout / Retry Interval, in seconds, 0-based (for example, minimum timeout = 1 second). This value sets the timeout waiting for an acknowledge, or the time between automatic retries depending on whether the alert is acknowledged or not. Recommended factory default = 3 seconds. Value is ignored if alert type does not support acknowledge, or if the Alert Acknowledge bit (above) is 0b. • **data 4**—Retries ◦ [7:4]—Reserved ◦ [3]—Reserved ◦ [2:0]—Number of times to retry alert to given destination. 0 = no retries (alert is only sent once). If the alert is acknowledged (Alert Acknowledge bit = 1b), the alert will only be retried if a timeout occurs waiting for the acknowledge. Otherwise, this value selects the number of times an unacknowledged alert will be sent out. The timeout interval or time between retries is

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
		set by the Alert Acknowledge Timeout / Retry Interval value (byte 3 of this parameter).

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
Destination Addresses	19	Sets/Gets the list of IP addresses that a LAN alert can be sent to. This parameter is not present if the Number of Destinations parameter is 0. • **data 1**—Set Selector = Destination Selector. ◦ [7:4]—Reserved ◦ [3:0]—Destination selector. Destination 0 is always present as a volatile destination that is used with the Alert Immediate command. • **data 2**—Address Format ◦ [7:4]—Address Format. 0h = IPv4 IP Address followed by DIX Ethernet/802.3 MAC Address 1h = IPv6 IP Address ◦ [3:0]—Reserved For Address Format = 0h: • **data 3**—Gateway selector ◦ [7:1]—Reserved ◦ [0] – 0b = Use default gateway Note: Older implementations (errata 4 or earlier) may only send to the default gateway. – 1b = Use backup gateway • **data 4:7**—Alerting IP Address (MS-byte first) • **data 8:13**—Alerting MAC Address (MS-byte first) For Address Format = 1h: **data 3:18**—Alerting IPv6 Address (MS-byte first) Router MAC Address is obtained through Neighbor Discovery or using the addressing specified using static router configuration in the LAN Configuration Parameters. iLO Notes: • Only IPv4 destinations are supported (Address Format = 0h). • Only IPv4 Default Gateway use is supported (Gateway selector = 0b).

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
The following parameters are introduced with IPMI v2.0 / RMCP+. VLAN configuration can be used with IPMI v1.5 and IPMI v2.0 sessions. Parameters labeled “RMCP+” are specific to IPMI v2.0 implementations that implement IPMI v2.0 / RMCP+ sessions.		
802.1q VLAN ID (12-bit)	20	• **data 1** [7:0]—Least significant 8-bits of the VLAN ID. 00h if VLAN ID not used. • **data 2** ◦ [7]—VLAN ID enable. – 0b = disabled – 1b = Enabled. If enabled, the BMC will only accept packets for this channel if they have 802.1q fields and their VLAN ID matches the VLAN ID value given in this parameter. ◦ [6:4]—Reserved ◦ [3:0]—Most significant four bits of the VLAN ID iLO Notes: • Only supported on iLO Shared Network Port (LOM or ALOM). • VLAN ID must be zero when VLAN ID enable is disabled (0b). • When enabled, VLAN ID must be between 1 and 4094. • Requires iLO reset to take effect.
802.1q VLAN Priority	21	**data 1** [7:3]—Reserved [2:0]—Value for Priority field of 802.1q fields. Ignored when VLAN ID enable is 0b (disabled). For more information, see 802.1q VLAN ID parameter, above. Setting is network dependent. By default, this should be set to 000b. iLO Notes: • Read only on iLO. • Only value 000b is supported.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
RMCP+ Messaging Cipher Suite Entry Support (Read Only)	22	This parameter provides a count of the number (16 max.) of Cipher Suites available to be enabled for use with IPMI Messaging on the given channel. Software can find out what security algorithms are associated with given Cipher Suite ID by using the Get Channel Cipher Suites command. In addition, there are Cipher Suite IDs assigned for standard Cipher Suites (see Table 22-19, Cipher Suite IDs) **data 1** [7:5]—reserved [4:0]—Cipher Suite Entry count. Number of Cipher Suite entries, 1-based, 10h max.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
Destination Address VLAN TAGs (can be READ ONLY, see description)	25	Sets/Gets the VLAN IDs (if any) addresses that a LAN alert can be sent to. This parameter is not present if the Number of Destinations parameter is 0, or if the implementation does not support the use of VLAN IDs for alerts. Otherwise, the number of VLAN TAG entries matches the number of Alert Destinations. An implementation may only be able to send alerts using the same VLAN TAG configuration as specified by parameters 20 and 21, in which case this parameter is allowed to be READ ONLY, where data 3-4 reflects the settings of parameters 20 and 21, and data 2 [7:4] indicates that VLAN TAGs are being used for alerts. If the implementation does support configurable VLAN TAGs for alert destinations, it must support configuring unique TAG information for all destinations on the given channel. • **data 1**—Set Selector = Destination Selector. ◦ [7:4]—Reserved ◦ [3:0]—Destination selector. Destination 0 is always present as a volatile destination that is used with the Alert Immediate command. • **data 2**—Address Format ◦ [7:4]—Address Format. – 0h = VLAN ID not used with this destination – 1h = 802.1q VLAN TAG ◦ [3:0]—Reserved For Address Format = 1h: • **data 3-4**—VLAN TAG ◦ [7:0]—VLAN ID, least-significant byte ◦ [11:8]—VLAN ID, most-significant nibble ◦ [12]—CFI (Canonical Format Indicator. Set to 0b) ◦ [15:13]—User priority (000b, typical) iLO Notes: • This parameter is read-only on iLO. • iLO only supports VLAN on its Shared Network Port. • Reports VLAN ID from parameter 20 when VLAN is enabled. • Reports VLAN ID not used when iLO Dedicated network port is selected.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
IPv6/IPv4 Support (read only)	50	This parameter is mandatory if IPv6 addressing is supported. This parameter and the following parameters up to and including IPv6 Neighbor Discovery/SLAAC Timing Configuration (parameter 79) should not be supported if the implementation does not support IPv6 addressing. **data 1** • [2] - 1b = Implementation supports IPv6 destination addresses for LAN alerting. • [1] - 1b = Implementation can be configured to use both IPv4 and IPv6 addresses simultaneously. • [0] - 1b = Implementation can be configured to use IPv6 addresses only.
IPv6/IPv4 Addressing enables	51	This parameter is mandatory if IPv6 addressing is supported. **data 1**—The following values can be set according to the capabilities specified in parameter 50. • 00h = IPv6 addressing disabled. • 01h = Enable IPv6 addressing only. IPv4 addressing is disabled. • 02h = Enable IPv6 and IPv4 addressing simultaneously. iLO Notes: iLO supports only 02h Enable IPv6 and IPv4 addressing simultaneously.
IPv6 Header Static Traffic Class	52	This parameter is mandatory if IPv6 addressing is supported. Recommended default = 0. **data 1**—Traffic Class This field determines the Traffic Class used by the BMC when transmitting Alert packets using IPv6 addressing. Otherwise, the BMC uses the traffic class value from the remote console. For more information, see [RFC2460] and [RFC2474]. iLO Notes: GET, returns fixed value of 0.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
IPv6 Header Static Hop Limit	53	This parameter is mandatory if IPv6 addressing is supported. Default = 64. **data 1**—Static Hop Limit This parameter is used in two situations: • If the router returns the value unspecified (00h) as the hop limit. • When a static router configuration is used, the BMC does not pay attention to Router Advertisement messages and uses this value for the Hop Limit. iLO Notes: GET, returns the fixed default value of 64.
IPv6 Header Flow Label [OPTIONAL]	54	**data 1:3**—Flow Label, 20-bits, right justified, MS Byte first. Default = 0. If this configuration parameter is not supported, the Flow Label should be set to 0 per [RFC2460]. Bits [23:20] = reserved—set to 0b. iLO Notes: Returns the fixed default value of 0x000000.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
IPv6 Status (read only)	55	Provides the number of IPv6 addresses that are supported and configurable for use by the BMC for IPMI. This parameter is mandatory if IPv6 addressing is supported. **data 1**—Static address max Maximum number of static IPv6 addresses for establishing connections to the BMC. Note: In some implementations, this might exceed the number of simultaneous sessions supported on the channel. 0 indicates that static address configuration is not available. **data 2**—Dynamic address max Maximum number of Dynamic (SLAAC/ DHCPv6) IPv6 addresses that can be obtained for establishing connections to the BMC. Note: In some implementations, this might exceed the number of simultaneous sessions supported on the channel. 0 = Dynamic addressing is not supported by the BMC. **data 3** • [7:2]—Reserved • [1] ◦ 1b = SLAAC addressing is supported by the BMC • [0] ◦ 1b = DHCPv6 addressing is supported by the BMC (optional)

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]

IPv6 Static Address	56	This parameter is mandatory if IPv6 addressing is supported.
		
		data 1—Set Selector = Address selector, 0 based.
		
		BMC shall provide at least one IPv6 Static Address entry if static address configuration is supported. For the case of 0 Static addresses, only selector 0 is allowed, data[2:19] are reserved, data 20 = “disabled”.
		
		data 2—Address source/type
		
		• [7]—Enable=1/disable=0
		• [6:4]—Reserved
		• [3:0]—Source/type
		  ◦ 0h = Static
		  ◦ All other = Reserved
		
		data 3–18—IPv6 Address, MS-byte first.
		
		data 19—Address Prefix Length
		
		data 20—Address Status (Read-only parameter). Writes to this location are ignored.
		
		• 00h = Active (in-use)
		• 01h = Disabled
		• 02h = Pending (currently undergoing DAD [duplicate address detection], optional)
		• 03h = Failed (duplicate address found, optional)
		• 04h = Deprecated (preferred timer has expired, optional)
		• 05h = Invalid (validity timer has expired, optional)
		• All other—Reserved
		
		iLO Notes:
		
		data 2
		
		[7] = 0 disable will cause iLO to delete the static IPv6 address indicated by the set selector (iLO does not support enable/disable of static addresses.)This is the only way to delete a static IPv6 address on iLO.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
IPv6 DHCPv6 Static DUID storage length (read only)	57	This parameter is mandatory if IPv6 addressing is supported. **data 1**—The maximum number of 16-byte blocks that can be used for storing each DUID through the IPv6 DHCPv6 Static DUIDs parameter. 1-based. Returns 0 if IPv6 Static Address configuration is not supported. Per [RFC3315], the DUID can be up to 128 octets long, though most forms are significantly shorter. DUID Type 2 (vendor-assigned unique ID based on Enterprise Number) is the format that has a vendor-specific length and could be lengthy. It is recommended that the implementation supports at least 3 blocks per DUID. iLO Notes: Returns 0, indicating storage of this parameter is not supported.[2]

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
IPv6 DHCPv6 Static DUIDs	58	DUIDs (DHCP Unique Identifiers). Per [RFC3315], each DHCP client and server has a DUID. This parameter provides storage for each DUID that identifies a particular IPv6 Interface Association (IA). This parameter is mandatory if IPv6 addressing is supported. **data 1**—Set Selector = DUID selector, 0 based. Each set selector matches with the corresponding Set Selector for the IPv6 Static Addresses parameter. This parameter is mandatory if IPv6 addressing is supported. If IPv6 Static Address configuration is not supported, only selector 0 and block 0 is allowed, and nothing is returned for data 3-18. **data 2**—Block Selector, 0-based Selects which 16-byte block of DUID data to write for the DUID storage from the given Set Selector. **data 3-18**—DUID data for given block. The first byte of block 0 is the overall length of the following DUID data (1-based). The remaining DUID data is formatted and stored in network byte order (MS-bytes first) per [RFC3315], starting with the DUID Type field. Notes: Per [RFC3315], a client must associate at least one distinct IA with each of its network interfaces for which it is to request the assignment of IPv6 addresses from a DHCP server. The client uses the IAs assigned to an interface to obtain configuration information from a server for that interface. Each IA must be associated with exactly one interface. Consequently, the Set Selector for the DUID effectively becomes bound to a particular IA. Since the Set Selector for the DUID and the Set Selector for the IPv6 Address correspond to one another, the Set Selector for the IPv6 address is also associated with the IA. In effect, the Set Selector value becomes the handle for a particular IA. Depending on DUID type, a given MAC address can have more than one associated DUID (and IPv6 Address). The Type 3 DUID uses the link layer address directly (MAC address) and therefore would not support more than one DUID for the MAC address. Other DUID types do not have that restriction. iLO Notes: Returns the fixed MAC-based DUID-LL used by the iLO DHCPv6 client. Only set=0, block=0 is supported.[2]

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
IPv6 Dynamic (SLAAC/DHCPv6) Address (read only)	59	This parameter is mandatory if IPv6 addressing is supported. **data 1**—Set Selector = Address selector, 0 based. BMC shall provide at least one entry in the array. For the case of 0 SLAAC and DHCPv6 addresses, only selector 0 is allowed, data[2:20] are reserved, data 21 = “disabled”. Mandatory if IPv6 addressing is supported. **data 2**—Address source/type [7:4]—Reserved [3:0]—Source/type • 0—Reserved • 1—SLAAC (Stateless Address Auto Configuration) • 2—DHCPv6 (optional) • Other—Reserved **data 3-18**—IPv6 Address, MS-byte first. **data 19**—Address Prefix Length. **data 20**—Address Status • 0—Active (in-use) • 1—Disabled • 2—Pending (currently undergoing DAD, optional) • 3—Failed (duplicate address found, optional) • 4—Deprecated (preferred timer has expired, optional) • 5—Invalid (validity timer has expired, optional) • Other—Reserved iLO Notes: • iLO supports only Active and Deprecated status for dynamic addresses • iLO does not support Disabled, Pending, Failed, or Invalid status for dynamic addresses • Failed addresses are never added to the iLO network stack • Invalid addresses are removed from the iLO network stack and will not be reported

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
IPv6 DHCPv6 Dynamic DUID storage length (read only)	60	This parameter is mandatory if IPv6 addressing is supported. **data 1**—The maximum number of 16-byte blocks that can be used for storing each DUID through the IPv6 DHCPv6 Static DUIDs parameter. 1-based. Returns 0 if IPv6 Static Address configuration is not supported. Per [RFC3315], the DUID can be up to 128 octets long, though most forms are significantly shorter. DUID Type 2 (vendor-assigned unique ID based on enterprise number) is the format which has a vendor-specific length and could be lengthy. It is recommended that the implementation supports at least 3 blocks per DUID. iLO Notes: Returns 0, indicating that storage of this parameter is not supported.[2]

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
IPv6 DHCPv6 Dynamic DUIDs	61	This parameter is mandatory if IPv6 addressing is supported. DUIDs (DHCP Unique Identifiers). Per [RFC3315], each DHCP client and server has a DUID. Although DHCPv6 is not used for address assignment when IPv6 static addresses are enabled, DHCPv6 can be used for discovery, configuration, or other purposes. Therefore, this configuration parameter provides a mechanism for setting and returning a DUID that is associated with each IPv6 Static Address that is supported. The Set Selector for the Dynamic DUIDs Type parameter matches the set selector for the corresponding IPv6 Static address. If IPv6 Dynamic Address configuration is not supported, only selector 0 and block 0 is allowed and nothing is returned for data 3-18. **data 1**—Set Selector = DUID selector, 0 based. Each set selector matches with the corresponding Set Selector for the IPv6 Dynamic Addresses parameter. This parameter is mandatory if IPv6 addressing is supported. If IPv6 Dynamic Address configuration is not supported, only selector 0 is allowed and the type is returned as 0 (not supported). **data 2**—Block Selector (0-based) Selects which 16-byte block of DUID data to write for the DUID storage from the given Set Selector. **data 3-18**—DUID data for given block. The first byte of block 0 is the overall length of the following DUID data (1-based). The remaining DUID data is formatted and stored in network byte order (MS-bytes first) per [RFC3315], starting with the DUID Type field. For more information, see the notes for the IPv6 DHCPv6 Static DUIDs parameter (parameter number 58). iLO Notes: Returns the fixed MAC-based DUID-LL used by the iLO DHCPv6 client. Only set=0, block=0 is supported.[2]

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
IPv6 DHCPv6 Timing Configuration Support (read only)	62	This parameter is mandatory if IPv6 addressing is supported. **data 1** [7:2]—Reserved [1:0] • 00b = DHCPv6 timing configuration per IPMI is not supported. • 01b = 'Global' - Timing configuration applies across all interfaces (IAs) that use dynamic addressing and have DHCPv6 enabled. • 10b = 'Per interface' - Timing is configurable for each interface and used when DHCPv6 is enabled for the given interface (IA). • 11b = Reserved iLO Notes: iLO does not support modification of DHCPv6 timing. 00b is always reported.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
IPv6 DHCPv6 Timing & Configuration (optional, recommended)	63	This parameter is mandatory if DHCPv6 timing configuration is supported as indicated by the IPv6 DHCPv6 Timing Configuration Support parameter (parameter 62). If DHCPv6 Dynamic Address configuration is not supported, this parameter should not be implemented. This parameter is used to configure the default timing values for DHCPv6. These values are used when the BMC is initially powered up or reinitialized, and after DHCPv6 address configuration becomes enabled or is re-enabled. Note that some of these parameters might be superseded by values received from the DHCP server. The DHCPv6 Timing configuration can be global, where a single set of timing parameters applies to all DHCPv6 transactions for all supported interfaces on the channel that are using DCHPv6 for IPv6 address assignment on the channel. It can also be per interface in which case only one set of timing parameters needs to be supported. The IPv6 DHCPv6 Timing Configuration Support parameter lets software know what is supported by the implementation. **data 1**—Set Selector = IPv6 interface (IA) selector, 0 based. If global, the Set Selector is always 0. If per interface, each set selector matches the corresponding Set Selector for the IPv6 Dynamic Addresses parameter. **data 2**—Block Selector (0-based) Selects which 16-byte block of timing parameter data is accessed for the given Set Selector. iLO Notes: Returns the fixed internal values used by the iLO DHCPv6 client. Only set=0, blocks 0 or 1 are supported.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
IPv6 Router Address Configuration Control	64	This parameter is mandatory if IPv6 addressing is supported. Router discovery is part of support for SLAAC and DHCPv6 addressing (dynamic addressing). This parameter controls whether automated router discovery occurs when static addresses are used for the BMC. It also enables the use of static router addresses. **data 1** [7:2]—Reserved [1] 1b = Enable dynamic router address configuration through router advertisement messages. Router solicitation messages are sent with timing and behavior as specified in [RFC4861]. The router solicitation timing values from the IPv6 Neighbor Discovery/SLAAC Timing Configuration parameter (below) are used if that parameter is implemented. [0] 1b = enable static router address. If static and dynamic router addressing are enabled, the BMC shall attempt to use the static router address and prefix first. iLO Notes: • iLO only supports having both static and dynamic router address sources enabled. Neither source can be disabled. • iLO does not follow the static router address first rule, router address in use is determined by the iLO network stack using the rules found in RFC4861.
IPv6 Static Router 1 IP Address	65	This parameter is mandatory if IPv6 addressing is supported. **data 1:16**—IPv6 Router IP Address (Used when static address is selected in the IPv6 IP Address Source configuration parameter). 16 bytes of IPv6 address, MS-byte first. iLO Notes: Returns the iLO IPv6 Static Route #1 Gateway address.[3]
IPv6 Static Router 1 MAC Address	66	This parameter is mandatory if IPv6 addressing is supported. **data 1:6**—Router MAC Address. MS-byte first. iLO Notes: Returns the fixed value 00:00:00:00:00:00.[4]

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
IPv6 Static Router 1 Prefix Length	67	This parameter is mandatory if IPv6 addressing is supported. **data 1**—Prefix length. Only used with static addressing. Used to determine whether an address is on link (can be accessed directly) or off link (must go to the corresponding static gateway address). The upper bits of the first address entry in the IPv6 Static Addresses parameter are used for the prefix value. Prefix length should be from 0 to 128 as per [RFC4861]. iLO Notes: Returns the iLO IPv6 Static Route #1 Destination prefix length.
IPv6 Static Router 1 Prefix Value	68	This parameter is mandatory if IPv6 addressing is supported. **data1**—set selector (0-based). **data 2:17**—Prefix value. MS-byte first. iLO Notes: Returns the iLO IPv6 Static Route #1 Destination address.
IPv6 Static Router 2 IP Address	69	This parameter is mandatory if IPv6 addressing is supported. **data 1:16**—IPv6 Gateway IP Address (used when static address is selected in the IPv6 IP Address Source configuration parameter). 16 bytes of valid IPv6 address, MS-byte first. **data 1-16**—Static Default Gateway address, MS-byte first. iLO Notes: Returns the iLO IPv6 Static Route #2 Gateway address.[3]
IPv6 Static Router 2 MAC Address	70	This parameter is mandatory if IPv6 addressing is supported. **data 1:6**—MAC Address. MS-byte first. iLO Notes: Returns the fixed value 00:00:00:00:00:00.[4]

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
IPv6 Static Router 2 Prefix Length	71	This parameter is mandatory if IPv6 addressing is supported. **data 1**—Prefix length. Only used with static addressing. Used to determine whether an address is on link (can be accessed directly) or off link (must go to the corresponding Static Gateway address). The upper bits of the first address entry in the IPv6 Static Addresses parameter are used for the prefix value. Prefix length should be from 0 to 128 as per [RFC4861]. iLO Notes: Returns the iLO IPv6 Static Route #2 Destination prefix length.
IPv6 Static Router 2 Prefix Value	72	This parameter is mandatory if IPv6 addressing is supported. **data1**—set selector (0-based) **data 2:17**—Prefix value. MS-byte first. iLO Notes: Returns the iLO IPv6 Static Route #2 Destination address.
IPv6 Dynamic Router Info IP Address (read-only)	73	This parameter is mandatory if the Number of Dynamic Router Info Sets is non-zero. **data1**—set selector (0-based) **data 2:17**—IPv6 Router IP Address (used when static address is selected in the IPv6 IP Address Source configuration parameter). 16 bytes of valid IPv6 address, MS-byte first. Set to 0000_0000h if the corresponding IPv6 Dynamic Router Info Prefix Length = FFh. iLO Notes: Returns 8 sets supported.
IPv6 Dynamic Router Info MAC Address (read-only)	74	This parameter is mandatory if the Number of Dynamic Router Info Sets is non-zero. **data 1**—set selector (0-based) **data 1:6**—MAC Address. MS-byte first. Set to 00_0000h if the corresponding IPv6 Dynamic Router Info Prefix Length = FFh. iLO Notes: Returns the IP address from which a given valid Router Advertisement (RA) message containing address prefix information was received.[5]

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
IPv6 Dynamic Router Info Prefix Length (read-only)	75	This parameter is mandatory if the Number of Dynamic Router Info Sets is non-zero. **data1**—set selector (0-based) **data 2**—Prefix length. Used with dynamic router address configuration. Used to determine whether an address is on link (can be accessed directly) or off link (must go to the corresponding Dynamic Router address). The upper bits of the first address entry in the IPv6 Static Addresses parameter are used for the prefix value. Prefix length should be from 0 to 128 as per [RFC4861]. FFh = Dynamic address is unspecified. The corresponding IPv6 Dynamic Router Info IP Address and MAC Address fields should be ignored. iLO Notes: Returns the source MAC address from which a given valid RA message containing address prefix information was received.
IPv6 Dynamic Router Info Prefix Value (read-only)	76	This parameter is mandatory if the Number of Dynamic Router Info Sets is non-zero. **data1**—set selector (0-based) **data 2:17**—Prefix value. Set to all 0s if the corresponding IPv6 Dynamic Router Info Prefix Length = FFh. iLO Notes: Returns the prefix length of an advertised address in a valid RA message.
IPv6 Dynamic Router Received Hop Limit [optional, recommended]	77	This parameter is optional (recommended). This value is obtained from the router as part of a Router Advertisement message. The BMC returns the most recently received information, regardless of the router that sent the advertisement. iLO Notes: Returns the prefix value of an advertised address in a valid RA message.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
IPv6 Dynamic Router Received Hop Limit[optional, recommended]	78	This parameter is optional (recommended). This value is obtained from the router as part of a Router Advertisement message. The BMC returns the most recently received information, regardless of the router that sent the advertisement. iLO Notes: Returns the hop limit received in the last received valid RA message, regardless of whether the message contained an advertised address prefix or not.
IPv6 Neighbor Discovery/SLAAC Timing Configuration Support (read only)	79	This parameter is mandatory if IPv6 static router address configuration is supported. **data 1** [7:2]—Reserved [1:0] • 00b = IPv6 Neighbor Discovery/SLAAC timing configuration per IPMI is not supported. • 01b = Global - Timing configuration applies across all interfaces (IAs) that have IPv6 dynamic addressing enabled. • 10b = Per interface - Timing is configurable for each interface and used when IPv6 dynamic addressing is enabled for the given interface (IA). • 11b = Reserved iLO Notes: iLO does not support changing these timing parameters, 00b is always returned.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
IPv6 Neighbor Discovery/SLAAC Timing Configuration	80	This parameter is Mandatory if Neighbor Discovery/SLAAC Timing configuration is supported. If it is not supported, this parameter is not implemented. This parameter is used to configure the default timing values for IPv6 Neighbor Discovery/ SLAAC. These values are used when the BMC is initially powered up or reinitialized, and after IPv6 dynamic addressing becomes enabled or is re-enabled for the interface(s). Note that some of these parameters may be superseded by values received from the router. The Neighbor Discovery/SLAAC Timing configuration can be global, where a single set of timing parameters applies to all Neighbor Discovery/SLAAC transactions for all supported interfaces on the channel that are using Neighbor Discovery/SLAAC for IPv6 address assignment on the channel or per interface in which case only one set of timing parameters needs to be supported. The IPv6 Neighbor Discovery/SLAAC Timing Configuration Support parameter lets software know what is supported by the implementation. **data 1**—Set Selector = IPv6 interface (IA) selector, 0 based. If global the Set Selector is always 0. If per interface, each set selector matches with the corresponding Set Selector for the IPv6 Dynamic Addresses parameter. **data 2**—Block Selector (0-based) Selects which 16-byte block of timing parameter data is accessed for the given Set Selector. iLO Notes: Returns the fixed internal values use by the iLO network stack.
iLO IPv4 Options	192	iLO IPv4 Option Controls **data 1** [7:3]—Reserved [2]—1b = Enable WINS registration. [1]—1b = Enable DDS registration for IPv4 Addresses. [0]—1b = Enable Gateway Ping on Startup. **data 2** [7:0]—Reserved iLO Notes: Changes require iLO reset to take effect.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
iLO DHCPv4 Options	193	iLO DHCPv4 Client Option Controls **data 1** [7:1]—Reserved [0]—1b = Enable DHCPv4 address and subnet mask assignment **data 2** [7:6]—Reserved [5]—1b = Enable DHCPv4 configuration of WINS servers [4]—1b = Enable DHCPv4 configuration of NTP servers [3]—1b = Enable DHCPv4 configuration of DNS servers [2]—1b = Enable DHCPv4 configuration of iLO Domain Name [1]—1b = Enable DHCPv4 configuration of IPv4 Static Routes [0]—1b = Enable DHCPv4 configuration of IPv4 Default Gateway Address iLO Notes: • Enable DHCPv4 in data 1 must also be set in order to set any options in data 2 • Clearing the enable DHCPv4 bit in data 1 will clear all data 2 options • Changes require iLO reset to take effect
iLO IPv4 DNS Server Addresses	194	iLO Static IPv4 DNS Server Addresses **data 1**—Set selector (0-primary, 1-secondary, 2-tertiary) **data 2**—Maximum number of IPv4 DNS addresses that can be set/read (write is ignored, 3 for iLO currently) **data 3:6**—DNS Address, MSB first iLO Notes: • When iLO IPv4 DNS server address is configured through DHCPv4, this parameter is read-only and will report the addresses provided by the DHCPv4 server. • Changes require iLO reset to take effect.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
iLO IPv4 WINS Server Addresses	195	iLO Static WINS Server Addresses **data 1**—Set selector (0–primary, 1–secondary) **data 2**—Maximum number of WINS server addresses that can be set/read (writes to this byte are ignored, 2 for iLO currently) **data 3:6**—WINS Server Address, MSB first iLO Notes: • When iLO WINS server addresses are configured by DHCPv4, this parameter is read-only and will report the addresses provided by the DHCPv4 server. • Changes require iLO reset to take effect.
iLO IPv4 Static Routes	196	iLO Static IPv4 Route Table Entries **data 1**—Set selector (0–2 for iLO currently) **data 2**—Maximum number of IPv4 static routes that can be set/read (writes to this byte are ignored, 3 for iLO currently) **data 3:6**—Destination address for this route table entry, MSB first **data 7:10**—Destination address subnet mask for this route table entry, MSB first **data 11:14**—Gateway address for this route table entry, MSB first iLO Notes: • When iLO IPv4 Static Routes are set to be configured by DHCPv4, this parameter is read-only, and will report the addresses provided by the DHCPv4 server. • Changes require iLO reset to take effect.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
iLO Shared NIC Selection and Shared NIC Port Number	197	iLO Shared NIC and Port Number Selection **data 1** • Shared NIC Selection • 01h = LOM • 02h = ALOM • All others = reserved **data 2** • Shared NIC Port Number • 01h = Use NIC port 1 • 02h = Use NIC port 2 • All others = reserved **data 3**—LOM/ALOM support status (writes to this byte are ignored) • [7:2]—Reserved • [1]—1b = Platform supports ALOM shared NIC • [0]—1b = Platform supports LOM shared NIC iLO Notes: • for Dedicated/Shared NIC selection • Shared NIC port number selection is not supported by all iLO Shared NICs, and will revert to port number 1 after iLO reset when unsupported. • Changes require iLO reset to take effect.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
iLO Ethernet Link Speed and Duplex	198	iLO Dedicated NIC Ethernet Link Speed and Duplex (set selector is ignored) **data 1** • [7:6]—Reserved • [5]—Autonegotiation Mode read-only status. ◦ 0b—Autonegotiation mode is configurable. ◦ 1b—Autonegotiation mode is read-only (blades and shared NIC) • [4]—Autonegotiation mode validity. ◦ 0b—Autonegotiation mode status is unknown (shared NIC) ◦ 1b—Autonegotiation mode as reported is valid. • [3]—Link status validity (link-state, link-duplex, link-speed) ◦ 0b = Link status data reported is not valid (inactive NIC) ◦ 1b = Link status data is valid. • [2]—Link-duplex. ◦ 0b = Link inactive or link-state unknown ◦ 1b = Link active. • [1]—Link-duplex. ◦ Half duplex or duplex state unknown • [0]—Autonegotiation Mode ◦ 0b = Using manual settings or autonegotiation mode unknown ◦ 1b = Use autonegotiation mode. **data 2**—Link speed. • 00h = 10 Mb/sec • 01h = 100 Mb/sec • 02h = 1000 Mb/sec • FFh = Link-speed is unknown • All others = reserved iLO Notes:

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
		• Only data2 and data1s bits [0] and [1] are configurable, if at all. • data1 [0] Autonegotiation mode is only configurable when data1 [5] is reported as 0b. • Speed and duplex on writes are ignored when set to autonegotiation mode. • Speed and duplex will report unknown for autonegotiation mode when the link-state indicator (data1 [2]) shows link inactive or unknown. • When in autonegotiation mode, link-speed, duplex, and state are only valid when data1 [3] is 1b. • When in manual settings mode, link-speed and duplex indicate configuration regardless of the state of data1 [3]. • Autonegotiation mode is forced on for blades and cannot be configured off. • Autonegotiation mode is unknown for shared NICs (it is controlled by the host NIC configuration and is not visible to iLO.) • Changes to this parameter require iLO reset to take effect.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]

iLO Network Configuration Status and NIC Selection	224	iLO Network Configuration Update Counters.
		
		iLO Dedicated/Shared NIC Selection.
		
		iLO Network Configuration Status.
		
		data 1
		
		IPv6 Configuration Counter. Unsigned 8–bit counter, updated each time IPv6 configuration is modified.
		
		data 2
		
		IPv4 Configuration Counter. Unsigned 8–bit counter, updated each time IPv4 configuration is modified.
		
		data 3
		
		• Selected iLO NIC.
		  ◦ 0h = iLO Dedicated NIC is selected.
		  ◦ 1h = iLO Shared NIC is selected.
		  ◦ All others = reserved
		
		data 4—iLO Network Configuration Status.
		
		[7:1]—Reserved
		
		[0]
		
		• 0b = All settings are in use currently.
		• 1b = iLO reset needed for some settings changes that have been made.
		
		iLO Notes:
		
		• Reading full iLO IP configuration through IPMI requires many individual transactions. To guarantee that no other iLO UI made network configuration changes during that process the configuration counters for IPv4 and IPv6 can be read at the beginning and at the end of reading iLO configuration. If the start and end counter values match, then no configuration changes occurred during the reading process.
		• The IPv6 configuration counter does not record changes to IPv6 address status.
		• To switch to another iLO NIC:
		
		Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
		◦ Write this (and possibly parameter 197) to the desired NIC selection. ◦ Configure all other relevant network parameters. ◦ Reset iLO. The selected NIC will be in use after the reset. • When writing changes to data 3, NIC selection: ◦ data 1 must be AAh ◦ data 2 must be 55h ◦ data 4 must be FFh
iLO IPv6 Options	225	iLO IPv6 Option Enables **data 1** [7:3]—Reserved [2]—1b = Enable SLAAC Address Creation from Router Advertisements. [1]—1b = Enable DDNS registration of IPv6 addresses. [0]—1b = iLO client applications prefer IPv6 protocol first over IPv4. **data 2**—Reserved iLO Notes: • bit [2] of data 1 only disables SLAAC address creation. Router Advertisement messages will still continue to be processed for other information, even when it is set to 0b. • Changes to SLAAC and DDNS options require an iLO reset take effect.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
iLO DHCPv6 Options	226	iLO DHCPv6 Configuration Control Enables **data 1** [7:3]—Reserved [2]—1b = Enable DHCPv6 Rapid Commit Mode [1]—1b = Enable Stateless DHCPv6 Mode [0]—1b = Enable Stateful DHCPv6 Mode **data 2** [7:3]—Reserved [2]—1b = Enable DHCPv6 configuration of iLO Domain Name. [1]—1b = Enable DHCPv6 configuration of IPv6 DNS server addresses. [0]—1b = Enable DHCPv6 configuration of IPv6 NTP server addresses. iLO Notes: • When both DHCPv4 and DHCPv6 are enabled to configure iLO Domain Name and the two servers provide different names, the DHCPv6 provided name will generally take precedence. • When DHCPv6 Stateful mode is enabled, Stateless mode is enabled by default (Stateless mode cannot be disabled when Stateful mode is enabled.) • DHCPv6 Rapid Commit mode should not be enabled in network environments where more than one DHCPv6 server may be enabled to provide address and configuration data. • Changes to DHCPv6 enabled/disabled status require iLO reset to take effect.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
iLO IPv6 DNS Server Addresses	227	IPv6 Based DNS Server Addresses for iLO **data 1**—Set selector • 0—primary • 1—secondary, • 2—tertiary **data 2**—Maximum number of IPv6 DNS addresses that are supported (writes ignored, 3 for iLO currently) **data [3:18]**—DNS Address, MSB first iLO Notes: When DHCPv6 is configuring the IPv6 DNS address list, this parameter is read-only and returns the values provided by the DHCPv6 server.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
iLO IPv6 Static Route Destination	228	Destination Address for an IPv6 Static Route Table Entry **data 1**—Set selector (0–2 for iLO currently) **data 2** [7]—1b = Route table entry is enabled [6] • 1b = Route table entry failed • 0b = Entry is active [5:0]—Reserved **data 3**—Maximum number of routes that are supported (writes ignored, 3 for iLO currently) **data [4:19]**—IPv6 Destination address (network prefix) for this route table entry, MSB first. **data 20**—Network prefix length for this destination address. iLO Notes: • This parameter works together with parameter 229 below to specify/status a complete iLO IPv6 static route table entry. • To set a static route table entry, fist write parameter 229 with the corresponding IPv6 gateway address, and then next write this parameter with the destination and network prefix length for the entry. • To read a static route table entry, first read this parameter, and then read the corresponding gateway address using parameter 229. • To delete an IPv6 static route table entry, set its enable bit in data 2 to disabled (parameter 229 does not need to be specified first in this case.) • Suggest using the Set In Progress indicator when reading and writing IPv6 static routes to prevent interaction between simultaneous IPMI sessions.

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
iLO IPv6 Static Route Gateway	229	Gateway address for an IPv6 Static Route Table Entry **data 1**—Set selector (0–2 for iLO currently.) **data [2:17]**—IPv6 Gateway address. MSB first. iLO Notes: • This parameter works in conjunction with parameter 228 above. See the notes for parameter 228 usage. • When clearing/disabling an IPv6 static route table entry, this parameter does not need to be specified. • IPv6 gateway addresses are typically link-local. Non-link local addresses used as a gateway address can cause the route table entry to fail when a no route to the specified gateway address condition is encountered (this will never happen when link-local gateway addresses are used.)

Table Continued

Parameter	#	Parameter Data (nonvolatile unless otherwise noted)[1]
iLO IPv4 Static IPv6 Default Gateway	230	Static Default Gateway IPv6 Address **data 1**—Set selector (0 only for iLO currently.) **data 2**—Maximum number of static default gateway addresses supported (writes ignored, currently 1 for iLO.) **data [3:18]**—IPv6 Gateway address, MSB first. iLO Notes: • IPv6 gateway addresses are typically link-local. Non-link local addresses used as a gateway address can cause the route table entry to fail when a no route to the specified gateway address condition is encountered (this will never happen when link-local gateway addresses are used.) • This entry is simply added to the gateway address table that is maintained by the iLO network stack as a possible gateway. There is no prioritization among the addresses in the gateway address table, and the network stack switches among them using the rules specified in RFC4861. • This parameter can be used to specify a default IPv6 gateway address in a network environment where no Router Advertisement messages are available.
iLO Current IPv6 Default Gateway (read-only)	231	Read only status indicates the IPv6 default gateway in use by iLO. **data [1:16]**—Current default IPv6 gateway address in use. iLO Notes: • Only valid for the selected iLO NIC. • Indicates the default gateway address currently being used for forwarding messages to off-link destinations. The address could be either the static default gateway address (if specified) or a gateway address that was learned through router advertisement messages received by iLO.

1 Choice of system manufacturing defaults is left to the system manufacturer unless otherwise specified.

2 The iLO DHCPv6 client uses the same fixed MAC-based DUID-LL for Stateless and Stateful operation modes (IPMI misnames these modes Static and Dynamic).

3 iLO supports up to three IPv6 Static Routes through read-write OEM command parameters. These read-only values are provided for convenience. They represent only the first two of the three supported routes.

4 iLO cannot configure IPv6 static routes by MAC address, and it does not have access to the gateway MAC address for a given static route.

5 Due to the way that IPMI defines these parameters, RA messages that have no address prefix data are not represented. Only valid RA messages that contain address prefix data are listed.

SOL commands

Set SOL configuration parameters command

This command is available to the MC.

This command is used for setting parameters such as the network addressing information required for SOL payload operation. Parameters can be volatile or non-volatile.

Table 56: Set SOL configuration parameters command request and response data

Request data byte number	Data field
1	• [7:4] — Reserved • [3:0] — Channel number
2	Parameter selector
3:N	Configuration parameter data. See **Get SOL configuration parameters command**.
Response data byte number	**Data field**
1	Completion code. • 80h = Parameter not supported. • 81h = Attempt to set the set in progress value (in parameter #0) when not in the set complete state. (This completion code provides a way to recognize that another party has already "claimed" the parameters). • 82h = Attempt to write read-only parameter. • 83h = Attempt to read write-only parameter.

Get SOL configuration parameters command

This command is available to the MC.

This command is used for retrieving the configuration parameters from the `set sol configuration parameters` command.

Table 57: Get SOL configuration parameters command request and response data

Request data byte number	Data field
1	• [7] ◦ 0b = Get parameter ◦ 1b = Get parameter revision only • [6:4] — Reserved • [3:0] — Channel number
2	Parameter selector

Table Continued

3	Set selector. Selects a given set of parameters under a given parameter selector value. 00h if parameter does not use a set selector.
4	Block selector (00h if parameter does not require a block number).
Response data byte number	**Data field**
1	Completion code. Generic codes, plus command-specific completion code: 80h = parameter not supported.
	[7:0] - Parameter revision. Format: MSN = present revision. LSN = oldest revision parameter in which it is backward compatible. 11h for parameters in this specification.
The following data byte is not returned when the get parameter revision only bit is 1b.	
3:N	Configuration parameter data, per the following *SOL configuration parameters* table. If the rollback feature is implemented, the MC makes a copy of the existing parameters when the set in progress state becomes asserted. (See the set in progress parameter #0). While the set in progress state is active, the MC returns data from this copy of the parameters, plus any uncommitted changes that were made to the data. Otherwise, the MC returns parameter data from non-volatile storage.

Table 58: SOL configuration parameters

Parameter	#	Parameter Data (non-volatile unless otherwise noted)[1]		
Set In Progress (volatile)	0	Data 1 - This parameter is used to indicate when any of the following parameters are being updated, and when the updates are completed. The bit is primarily provided to alert software that some other software or utility is in the process of making changes to the data. An implementation can also elect to provide a rollback feature that uses this information to decide whether to roll back to the previous configuration information, or to accept the configuration change. If used, the roll back restores all parameters to their previous state. Otherwise, the change takes effect when the write occurs.		
		[7:2]	Reserved	
		[1:0]	00b =	Set complete. If a system reset or transition to powered down state occurs while set in progress is active, the MC goes to the set complete state. If rollback is implemented, going directly to set complete without first doing a commit write causes any pending write data to be discarded.
			01b =	Set in progress. This flag indicates that some utility or other software is presently doing writes to parameter data. It is a notification flag only, it is not a resource lock. The MC does not provide any interlock mechanism that would prevent other software from writing parameter data.

Table Continued

Parameter	#	Parameter Data (non-volatile unless otherwise noted)[1]		
			10b =	Commit write (optional). This is only used if a rollback is implemented. The MC saves the data that has been written since the last time it was set in progress and then goes to the set in progress state. An error completion code will be returned if this option is not supported.
			11b =	Reserved
SOL enable	1	Byte 1:		
		[7:1]	Reserved	
			[0]	SOL enable. This controls whether the SOL payload type can be activated. Whether an SOL stream can be established is also dependent on the access mode and authentication settings for the corresponding LAN channel. The enabled/disabled state and access mode settings for the serial/modem channel have no effect on SOL.
			1b =	Enable SOL payload
			0b =	Disable SOL payload
SOL authentication	2	Byte 1: SOL authentication enable		
		[7]	Force SOL payload encryption	
			1b:	Force encryption. If the cipher suite for the session supports encryption, this setting forces the use of encryption for all SOL payload data.
			0b:	Encryption controlled by remote console. Whether SOL packets are encrypted or not is selected by the remote console at the time the payload is activated (using the `activate payload` command) and can be changed during operation via the `suspend/resume payload encryption` command.
		[6]	Force SOL payload authentication	
			1b:	Force authentication. If the cipher suite for the session supports authentication, this setting forces the use of authentication on all SOL payload data.
			0b:	Authentication controlled by remote software. For the standard cipher suites, if encryption is used then authentication must also be used. Therefore, while encryption is being used, software is not be able to select using unauthenticated payloads.
		[5:4]	Reserved	

Table Continued

Parameter	#	Parameter Data (non-volatile unless otherwise noted)[1]
		[3:0] SOL privilege level. Sets the minimum operating privilege level that is required to be able to activate SOL using the `activate payload` command.

0h:	Reserved
1h:	Reserved
2h:	User level
3h:	Operator level
4h:	Administrator level
5h:	OEM proprietary level
All other :	Reserved

Character accumulate interval & character send threshold	3	• Byte 1: Character accumulate interval in 5 ms increments. 1-based. This sets the typical amount of time that the MC waits before transmitting a partial SOL character data packet. (Where a partial packet is defined as a packet that has fewer characters to transmit than the number of characters specified by the Send threshold parameter (see below). A packet is not sent. 00h = reserved • Byte 2: Character send threshold. 1-based. The MC automatically sends an SOL character data packet containing this number of characters as soon as this number of characters (or greater) has been accepted from the baseboard serial controller into the MC. This provides a mechanism to tune the buffer to reduce latency to when the first characters are received after an idle interval. In the degenerate case, setting this value to 1 would cause the MC to send a packet as soon as the first character was received. This can be useful if the character accumulate interval is large. If the MC is waiting for an acknowledge from the previous packet, it ignores this threshold and continues to collect data until it has a full packet's worth.
SOL retry	4	• Byte 1: Retry count ◦ [7:3] - Reserved ◦ [2:0] - Retry count. 1-based. 0 = no retries after packet is transmitted. Packet is dropped if no ACK/NACK received by the time retries expire. • Byte 2: Retry interval. 1-based. Retry interval in 10 ms increments. Sets the time that the MC waits before the first retry and the time between retries when sending SOL packets to the remote console. ◦ 00h: Retries sent back-to-back

Table Continued

Parameter	#	Parameter Data (non-volatile unless otherwise noted)[1]
SOL non-volatile bit rate (non-volatile)	5	This configuration parameter is not supported if the implementation does not have a MC serial controller that can be potentially configured. Serial communication with the MC when SOL is activated always occurs using 8 bits/character, no parity, 1 stop bit, and RTS/CTS (hardware) flow control. **NOTE:** If SOL is enabled for multiple LAN channels, the MC uses the serial communication settings for the channel over which the `activate sol` command was initially received. The settings for other channels are ignored. Data 1 [7:4] Reserved [3:0] Bit rate. 1-5h = reserved. Support for bit rates other than 19.2 kbps is optional. The MC must return an error completion if a requested bit rate is not supported. It is recommended that the parameter out-of-range (C9h) code be used for this situation. 0h = Use setting presently configured for IPMI over serial channel. The setting is used even if the access mode for the serial channel is set to disabled. IPMI specification can allow more than one serial channel. If serial port sharing is not implemented, this value is reserved. 6h: 9600 bps. 7h = 19.2 kbps 8h = 38.4 kbps 9h = 57.6 kbps Ah = 115.2 kbps All other = Reserved
SOL volatile bit rate (volatile)	6	Set volatile version of SOL serial settings. Data follows that for the SOL non-volatile bit rate parameter.
SOL payload channel (optional, read only)	7	This parameter indicates which IPMI channel is being used for the communication parameters (such as IP address, MAC address) for the SOL payload. Typically, these parameters come from the same channel that the `activate payload` command for SOL was accepted.
SOL payload port number (read only or read/write)	8	This parameter is read/write when the implementation allows the port number over which the SOL payload can be activated to be configurable. Otherwise, it is a read only parameter. Data 1:2 - Primary RMCP port number, LS byte first.
OEM parameters	192:255	This range is available for special OEM configuration parameters. The OEM is identified according to the manufacturer ID field returned by the `get device id` command.

[1] Choice of system manufacturing defaults is left to the system manufacturer unless otherwise specified.

MC watchdog timer commands

The MC implements a standardized watchdog timer that can be used for a number of system timeout functions by system management software or by the BIOS. Setting a timeout value of 0 allows the selected timeout action to occur immediately. This provides a standardized means for devices on the IPMB, such as remote management cards, to perform emergency recovery actions.

Watchdog timer actions

The following actions are available on expiration of the watchdog timer:

- System reset
- System power off
- System power cycle
- Pre-timeout interrupt (optional)

The system reset on timeout, system power off on timeout, and system power cycle on timeout action selections are mutually exclusive. The watchdog timer is stopped whenever the system is powered-down. A command must be sent to start the timer after the system powers up.

Watchdog timer use field and expiration flags

The watchdog timer provides a timer use field that indicates the current use assigned to the watchdog timer. The watchdog timer provides a corresponding set of timer use expiration flags that are used to track the type of timeout that had occurred.

The timeout use expiration flags retain their state across system resets and power cycles, as long as the MC remains powered. The flags are normally cleared solely by the `set watchdog timer` command; with the exception of the don't log flag, which is cleared after every system hard reset or timer timeout.

The timer use fields indicate:

Timer use field	Description
BIOS FRB2 timeout	An FRB-2 (fault-resilient booting, level 2) timeout has occurred indicating that the last system reset or power cycle was due to the system timeout during POST, presumed to be caused by a failure or hang related to the bootstrap processor.[1]
BIOS POST timeout	In this mode, the timeout occurred while the watchdog timer was being used by the BIOS for some purpose other than FRB-2 or OS load watchdog.
OS load timeout	The last reset or power cycle was caused by the timer being used to watchdog the interval from boot to OS up and running. This mode requires system management software, or OS support. BIOS should clear this flag if it starts this timer during POST.

Table Continued

Timer use field	Description
SMS OS watchdog timeout	Indicates that the timer was being used by SMS. During run-time, SMS starts the timer, then periodically resets it to keep it from expiring. This periodic action serves as a heartbeat that indicates that the OS (or at least the SMS task) is still functioning. If SMS hangs, the timer expires and the MC generates a system reset. When SMS enables the timer, it should make sure the SMS bit is set to indicate that the timer is being used in its OS watchdog role.
OEM	Indicates that the timer was being used for an OEM-specific function.

[1] In a multiprocessor system, the bootstrap processor is defined as the processor that, on system power-up or hard reset, is allowed to run and execute system initialization (BIOS POST) while the remaining processors are held in an idle state awaiting startup by the multiprocessing OS.

Using the timer use field and expiration flags

The software that sets the timer use field is responsible for managing the associated timer use expiration flag. For example, if SMS sets the timer use to SMS/OS watchdog, then that same SMS is responsible for acting on and clearing the associated timer use expiration flag.

In addition, software should only interpret or manage the expiration flags for watchdog timer uses that it set. For example, BIOS should not report watchdog timer expirations or clear the expiration flags for non-BIOS uses of the timer. This is to allow the software that did set the timer use to see that a matching expiration occurred.

Watchdog timer event logging

By default, the MC automatically logs the corresponding sensor-specific watchdog sensor event when a timer expiration occurs. A don't log bit is provided to temporarily disable the automatic logging. The don't log bit is automatically cleared (logging re-enabled) whenever a timer expiration occurs.

Pre-timeout interrupt

The watchdog timer offers a pre-timeout interrupt option. This option is enabled whenever the interrupt on timeout option is selected along with any of the other watchdog timer actions. If this option is enabled, the MC generates the selected interrupt a fixed interval before the timer expires. This feature can be used to allow an interrupt handler to intercept the timeout event before it actually occurs. The default pre-timeout interrupt interval is one (1) second.

The watchdog timeout action and the pre-timeout interrupt functions are individually enabled. Thus, the watchdog timer can be configured so that when it times out it provides just an interrupt, just the selected action, both an interrupt and selected action, or none.

If the pre-timeout interval is set to zero, the pre-timeout action occurs concurrently with the timeout action. If a power or reset action is selected with a pre-timeout interval of zero, there is no guarantee that a pre-timeout interrupt handler would have time to execute, or to run to completion.

Pre-timeout interrupt support detection

An application that wishes to use a particular pre-timeout interrupt can check for its support by issuing a `set watchdog timer` command with the desired pre-timeout interrupt selection. If the controller does not return an error completion code, then a `get watchdog timer` command should be issued to verify that the interrupt selection was accepted.

While it can be assumed that a controller that accepts a given interrupt selection supports the associated interrupt, it is recommended that, if possible, an application also generate a test interrupt and verify that the interrupt occurs and the handler executes correctly.

BIOS support for watchdog timer

If a system `warm reset` occurs, the watchdog timer may still be running while BIOS executes POST. Therefore, BIOS should take steps to stop or restart the watchdog timer early in POST. Otherwise, the timer may expire later during POST or after the OS has booted.

Reset watchdog timer command

This command is used for starting and restarting the watchdog timer from the initial countdown value that was specified in the `set watchdog timer` command. If a pre-timeout interrupt has been configured, the `reset watchdog timer` command is not restart the timer once the pre-timeout interrupt interval has been reached. The only way to stop the timer once it has reached this point is via the `set watchdog timer` command.

Table 59: Reset watchdog timer command response data

Response data byte number	Data field
1	Completion code. Generic plus command-specific completion code: 80h — Attempt to start un-initialized watchdog. It is recommended that a MC implementation return this error completion code to indicate to software that a `set watchdog timer` command has not been issued to initialize the timer since the last system power on, reset, or MC reset. Since many systems may initialize the watchdog timer during BIOS operation, this condition may only be seen by software if a MC gets re-initialized during system operation (as might be the case if a firmware update occurred, for example).

Set watchdog timer command

This command is used for initializing and configuring the watchdog timer. The command is also used for stopping the timer.

If the timer is already running, the `set watchdog timer` command stops the timer (unless the don't stop bit is set) and clears the watchdog pre-timeout interrupt flag. MC hard resets, system hard resets, and the `cold reset` command also stop the timer and clear the flag.

- Byte 1 is used for selecting the timer use and configuring whether an event will be logged on expiration.
- Byte 2 is used for selecting the timeout action and pre-timeout interrupt type.
- Byte 3 sets the pre-timeout interval. If the interval is set to zero, the pre-timeout action occurs concurrently with the timeout action.
- Byte 4 is used for clearing the timer use expiration flags. A bit set in byte 4 of this command clears the corresponding bit in byte 5 of the `get watchdog timer` command.
- Bytes 5 and 6 hold the least significant and most significant bytes, respectively, of the countdown value. The watchdog timer decrement is one count/100 ms. The counter expires when the count reaches zero. If the counter is loaded with zero and the `reset watchdog` command is issued to start the timer, the associated timer events occur immediately.

Table 60: Set watchdog timer command request and response data

Request data byte number	Data field		
1	Timer use		
	[7]	1b =	Don't log
	[6]	1b =	Don't stop the timer on `set watchdog timer` command. New parameters take effect immediately. If the timer is already running, the countdown value gets set to the given value and the countdown continues from that point. If the timer is already stopped, it remains stopped. If the pre-timeout interrupt bit is set, it is cleared. [1]
		0b =	Timer stops automatically when the `set watchdog timer command` is received.
	[5:3]	Reserved	
	[2:0]	Timer use (logged on expiration when don't log bit = 0b)	
		000b =	Reserved
		001b =	BIOS FRB2
		010b =	BIOS/POST
		011b =	OS Load
		100b =	SMS/OS
		101b =	OEM
		110b -111b =	Reserved
2	Timer actions		
	[7]	Reserved	
	[6:4]	Pre-timeout interrupt (logged on expiration when don't log bit = 0b)	
		000b =	None
		001b =	SMI (optional)
		010b =	NMI / diagnostic interrupt (optional)

Table Continued

		011b =	Messaging interrupt (this is the same interrupt as allocated to the messaging interface, if communications interrupts are supported for the system interface).
		100b,111b =	Reserved
	[3]	Reserved	
	[2:0]	Timeout action	
		000b =	No action
		001b =	Hard reset
		010b =	Power down
		011b =	Power dycle
		100b,111b =	Reserved
3	Pre-timeout interval in seconds. 1 based.		
4	Timer use expiration flags clear. (0b = leave alone, 1b = clear timer use expiration bit).		
	[7]	Reserved	
	[6]	Reserved	
	[5]	OEM	
	[4]	SMS/OS	
	[3]	OS load	
	[2]	BIOS/POST	
	[1]	BIOS FRB2	
	[0]	Reserved	
5	Initial countdown value, LS byte (100 ms/count)		
6	Initial countdown value, MS byte		

[1] Potential race conditions exist with implementations of this option. If the `set watchdog timer` command is sent just before a pre-timeout interrupt or timeout is set to occur, the timeout could occur before the command is executed. To avoid this condition, it is recommended that software set this value no closer than 3 counts before the pre-timeout or timeout value is reached.

Response data byte number	Data field
1	Completion code

[1] Potential race conditions exist with implementations of this option. If the `set watchdog timer` command is sent just before a pre-timeout interrupt or timeout is set to occur, the timeout could occur before the command is executed. To avoid this condition, it is recommended that software set this value no closer than 3 counts before the pre-timeout or timeout value is reached.

Get watchdog timer command

This command retrieves the current settings and present countdown of the watchdog timer. The timer use expiration flags in byte 5 retain their states across system resets and system power cycles. With the exception of bit 6 in the timer use byte, the timer use expiration flags are cleared using the `set watchdog timer` command. They may also become cleared because of a loss of MC power, firmware update, or other cause of MC hard reset. Bit 6 of the timer use byte is automatically cleared to 0b whenever the timer times out, is stopped when the system is powered down, enters a sleep state, or is reset.

Table 61: Get watchdog timer command response data

Response data byte number	Data field		
1	Completion code		
2	Timer use		
	[7]	1b =	Don't log
	[6]	1b =	Timer is started (running)
		0b =	Timer is stopped
	[5:3]	Reserved	
	[2:0]	Timer use (logged on expiration when don't log bit = 0b)	
		000b =	Reserved
		001b =	BIOS FRB2
		010b =	BIOS/POST
		011b =	OS Load
		100b =	SMS/OS
		101b =	OEM

Table Continued

		110b -111b =	Reserved
3	Timer actions		
	[7]	Reserved	
	[6:4]	Pre-timeout interrupt (logged on expiration when don't log bit = 0b)	
		000b =	None
		001b =	SMI (if implemented)
		010b =	NMI / diagnostic interrupt (if implemented)
		011b =	Messaging interrupt (this is the same interrupt as allocated to the messaging interface).
		100b,111b =	Reserved
	[3]	Reserved	
	[2:0]	Timeout action	
		000b =	No action
		001b =	Hard reset
		010b =	Power down
		011b =	Power dycle
		100b,111b =	Reserved
4	Pre-timeout interval in seconds. 1 based.		
5	Timer use expiration flags clear. (1b = timer expired while associated use was selected.)		
	[7]	Reserved	
	[6]	Reserved	
	[5]	OEM	
	[4]	SMS/OS	

Table Continued

	[3]	OS load
	[2]	BIOS/POST
	[1]	BIOS FRB2
	[0]	Reserved
6	Initial countdown value, LS byte (100 ms/count)	
7	Initial countdown value, MS byte	
8	Present countdown value, LS byte. The initial countdown value and present countdown values should match immediately after the countdown is initialized via a `set watchdog timer` command and after a `reset watchdog timer` has been executed. Internal delays in the MC may require software to delay up to 100 ms before seeing the countdown value change and be reflected in the `get watchdog timer` command.	
9	Present countdown value, MS byte	

Chassis commands

The following chassis commands are specified for IPMI v1.5. These commands are primarily to provide standardized chassis status and control functions for remote management cards and remote consoles that access the MC. They can also be used for emergency management control functions by system management software.

Get chassis capabilities command

This command returns information about which main chassis management functions are present on the IPMB (or virtual IPMB) and what addresses are used to access those functions. This command is used to find the devices that provide functions such as SEL, SDR, and ICMB bridging so that they can be accessed via commands delivered via a physical or logical IPMB. The command does not include a channel number for the individual functions, therefore all reported functions must be located on the primary IPMB.

The chassis capabilities information is non-volatile. There is no requirement that the information be configurable. The chassis device function in a peripheral chassis may be hardcoded with this information. For example, a system that implements the ICMB as an add-on bridge to an MC is typically able to have the well known address for the MC (20h) hardcoded as the address for the chassis SDR, SEL, and SM devices, while the chassis FRU info device address could be set with the chassis devices own address.

An add-in device that serves as a bridge device that could be used in different vendors systems may want to provide a way for this information to be configured. The `set chassis capabilities` command is one option for providing this.

Table 62: Get chassis capabilities command response data

Response data byte number	Data field		
1	Completion code		
2	Capabilities flags		
	[7:4]	Reserved	
	[3]	1b =	Provides power interlock (IPMI 1.5)
	[2]	1b =	Provides diagnostic interrupt, FP NMI. (IPMI 1.5)
	[1]	1b =	Provides front panel lockout which indicates that the chassis has capabilities to lock out external power control and reset button or front panel interfaces and/or detect tampering with those interfaces.
	[0]	1b =	Chassis provides intrusion (physical security) sensor
3	Chassis FRU info device address. All IPMB addresses used in this command have the 7-bit I^2C slave address as the most-significant 7-bits and the least significant bit set to 0b. 00h = unspecified.		
4	Chassis SDR device address		
5	Chassis SEL device address		
6	Chassis system management device address		
(7)	Chassis bridge device address. Reports location of the ICMB bridge function. If this field is not provided, the address is assumed to be the MC address (20h). Implementing this field is required when the `get chassis capabilities` command is implemented by an MC, and whenever the chassis bridge function is implemented at an address other than 20h.		

Get chassis status command

This command is available to ChMC and MC.

This command returns information regarding the high-level status of the system chassis and main power subsystem.

Table 63: Get chassis status command response data

Response data byte number	Data field
1	Completion code
2	Current power state

Table Continued

Response data byte number	Data field		
	[7]	Reserved	
	[6:5]	Power restore policy[1]	
		00b =	Chassis stays powered off after AC/mains returns.
		01b =	After AC returns, power is restored to the state that was in effect when AC/mains was lost.
		10b =	Chassis always powers up after AC/mains returns.
		11b =	Unknown
	[4]	Power control fault	
		1b =	Controller attempted to turn system power on or off, but system did not enter desired state.
	[3]	Power fault	
		1b =	Fault detected in main power subsystem.
	[2]	1b =	Interlock (chassis is presently shut down because a chassis panel interlock switch is active). (IPMI 1.5).
	[1]	Power overload	
		1b =	System shutdown because of power overload condition.
	[0]	Power is on.	
		1b =	System power is on.
		0b =	System power is off (soft-off S4/S5 or mechanical off).
3	Last power event		
	[7:5]	Reserved	
	[4]	1b =	Last Power is on state was entered through an IPMI command.
	[3]	1b =	Last power down caused by power fault.
	[2]	1b =	Last power down caused by a power interlock being activated.

Table Continued

Response data byte number	Data field		
	[1]	1b =	Last power down caused by a power overload.
	[0]	1b =	AC failed
4	Miscellaneous chassis state		
	[7]	Reserved	
	[6]	1b =	Chassis identify command and state information supported (optional).
		0b =	Chassis identify command support unspecified via this command. (The `get command support` command, if implemented, would still indicate support for the `chassis identify` command).
	[5:4]	Chassis identify state. Mandatory when bit [6] = 1b, reserved (return as 00b) otherwise. Returns the present chassis identify state. See **Chassis identify command** .	
	[3]	1b =	Cooling/fan fault detected
	[2]	1b =	Drive fault
	[1]	1b =	Front panel lockout active (power off and reset through chassis push-buttons disabled.)
	[0]	1b =	Chassis intrusion active
(5)	Front panel button capabilities and disable/enable status (optional). Button actually refers to the ability for the local user to be able to perform the specified functions through a pushbutton, switch, or other front panel control built into the system chassis.		
	[7]	1b =	Standby (sleep) button disable allowed
	[6]	1b =	Diagnostic Interrupt button disable allowed.
	[5]	1b =	Reset button disable allowed.
	[4]	1b =	Power off button disable allowed (in the case there is a single combined power/standby (sleep) button, disabling power off also disables sleep requests through that button).
	[3]	1b =	Standby (sleep) button disabled
	[2]	1b =	Diagnostic Interrupt button disabled

Table Continued

Response data byte number	Data field		
	[1]	1b =	Reset button disabled
	[0]	1b =	Power off button disabled (in the case there is a single combined power/ standby (sleep) button, then this indicates that sleep requests through that button are also disabled).

[1] In some installations, the chassis' main power feed might be DC based. For example, -48V. In this case, the power restore policy for AC/ mains refers to the loss and restoration of the DC main power feed.

Chassis control command

This command is available to the MC.

This command provides a mechanism for providing power up, power down, and reset control.

Table 64: Chassis control command request data

Request data byte number	Data field		
1	[7:4]	Reserved	
	[3:0]	Chassis control[1]	
		0h =	Power down. Force system into soft off (S4/S45) state. This is for emergency management power down actions. The command does not initiate a clean shut-down of the operating system before powering down the system.
		1h =	Power up.
		2h =	Power cycle (optional). This command provides a power off interval of at least 1 second following the de-assertion of the system's power good status from the main power subsystem. It is recommended that no action occur if system power is off (S4/S5) when this action is selected, and that a D5h `Request parameter(s) not supported in present state.` error completion code be returned. Some implementations may cause a system power up if a power cycle operation is selected when system power is down. For consistency of operation, it is recommended that SMS first check the system power state before issuing a power cycle, and only issue the command if the system power is on or in a lower sleep state than S4/S5.
		3h =	Hard reset. In some implementations, the MC may not know whether a reset causes any particular effect and pulses the system reset signal regardless of power state. If the implementation can tell that no action occurs if a reset is delivered in a given power state, then it is recommended (but still optional) that a D5h `Request parameter(s) not supported in present state.` error completion code be returned.

Table Continued

Request data byte number	Data field	
	4h =	Pulse diagnostic interrupt (optional). Pulse a version of a diagnostic interrupt that goes directly to the processor(s). This is typically used to cause the operating system to do a diagnostic dump (OS dependent). The interrupt is commonly an NMI on IA-32 systems and an INIT on Intel Itanium processor based systems.
	5h =	Initiate a soft-shutdown of OS via ACPI by emulating a fatal overtemperature (optional).
	All other =	Reserved

[1] The command can also be used for compute blades or compute partition applications where the blades or partitions entities are emulating independent computer systems that implement IPMI. In these applications, the chassis power control aspects of the command are not required to be supported. Individual blades or computer partitions can elect to either not support the power on/off functions, can use them for power control of the blade/partition independent of the containing chassis, or may map them into a power control scheme for the overall chassis. For example, a scheme where chassis power will go off only after all blades within a chassis have been commanded into the power off state.

Table 65: Chassis control command response data

Response data byte number	Data field
1	Completion code[1]

[1] The implementation is allowed to return the completion code before performing the selected control action if necessary.

Chassis identify command

This command is available to the MC.

This command causes the chassis to physically identify itself by a mechanism chosen by the system implementation; such as turning on blinking user-visible lights or emitting beeps via a speaker, LCD panel, etc. Unless the optional force identify on capability is supported and used, the `chassis identify` command automatically times out and deasserts the indication after a configurable time-out. Software must periodically resend the command to keep the identify condition asserted. This restarts the timeout.

Table 66: Chassis identify command request and response data

Request data byte number	Data field	
1[1]	[7:0]	Identify interval in seconds (optional). 1-based. Timing accuracy = -0/+20%. If this byte is not provided, the default timeout shall be 15 seconds -0/+20%. This byte can be overridden by optional byte 2. 00h = Turn off identify
2[2]	Force identify on (optional). This field enables software to command the identify to be on indefinitely. The MC implementation should return an error completion code if this byte is not supported.	

Table Continued

Request data byte number	Data field		
	[7:1]	Reserved	
	[0]	1b =	Turn on identify indefinitely. This overrides the values in byte 1.
		0b =	Identify state driven according to byte 1.

[1] This parameter byte is optionally present. If not provided, the dhassis identify can be used to turn on the identify indication for the default timeout interval, but cannot be used to turn the indication off.

[2] This parameter byte is optionally present. If provided, it is highly recommended that the chassis provides a local manual mechanism that enables a user or service personnel to turn off Identify. If a local manual mechanism is not provided, AC removal (MC reset) should remove the indication.

Response data byte number	Data field
1	Completion code

Set power restore policy command

This command is available to the MC.

This command can be used to configure the power restore policy. This configuration parameter is kept in non-volatile storage. The power restore policy determines how the system or chassis behaves when AC power returns after an AC power loss. The `get chassis status` command returns the power restore policy setting.

Table 67: Set power restore policy command request and response data

Request data byte number	Data field		
1	[7:3]	Reserved	
	[2:0]	Power restore policy[1]	
		011b =	No change (just get present policy support)
		010b =	Chassis always powers up after AC/mains is applied or returns
		001b =	After AC/mains is applied or returns, power is restored to the state that was in effect when AC/mains was removed or lost
		000b =	Chassis always stays powered off after AC/mains is applied, power push button or command required to power on system
		All other =	Reserved

[1] In some installations, the chassis' main power feed may be DC based. For example, -48V. In this case, the power restore policy for AC/mains refers to the loss and restoration of the DC main power feed.

Response data byte number	Data field		
1	Completion code. A non-zero completion code should be returned if an attempt is made to set a policy option that is not supported.		
2	Power restore policy support (bitfield)		
	[7:3]	Reserved	
	[2]	1b =	Chassis supports always powering up after AC/mains returns
	[1]	1b =	Chassis supports restoring power to the state that was in effect when AC/mains was lost
	[0]	1b =	Chassis supports staying powered off after AC/mains returns

[1] In some installations, the chassis' main power feed may be DC based. For example, -48V. In this case, the power restore policy for AC/mains refers to the loss and restoration of the DC main power feed.

Get system restart cause command

This command returns information about what action last caused the system to restart. BIOS can use this command in conjunction with the System Boot Options as additional information in determining whether to perform the requested boot operation.

Table 68: Get system restart cause command request and response data

Request data byte number	Data field		
-	-	-	

Response data byte number	Data field		
1	Completion code.		
2	Restart cause (bitfield)		
	[7:4]	Reserved	
	[3:0]	0h	Unknown (system start/restart detected, but cause unknown)
		1h	Chassis Control command
		2h	Reset through pushbutton
		3h	Power up through power pushbutton

Table Continued

Response data byte number	Data field	
	4h	Watchdog expiration (see watchdog flags)
	5h	OEM
	6h	Automatic power-up on AC being applied due to always restore power restore policy.
	7h	Automatic power-up on AC being applied due to restore previous power state power restore policy.
	8h	Reset through PEF
	9h	Power-cycle through PEF
	Ah	Soft reset (for example, CTRL-ALT-DEL)
	Bh	Power up through RTC (system real time clock) wakeup
3		

Set system boot options command

This command is available to the MC.

This command is used to set parameters that direct the system boot following a system power up or reset. The boot flags only apply for one system restart. It is the responsibility of the system BIOS to read these settings from the MC and then clear the boot flags.

It is possible that a remote console application could set the boot option flags and then be terminated either accidentally or intentionally. In this circumstance, it is possible that a user-initiated system restart could occur hours or even days later. If the boot options were used without examining the reset cause, this could cause an unexpected boot sequence. Thus, the MC automatically clears a boot flags valid bit if a system restart is not initiated by a `chassis control` command within 60 seconds +/- 10% of the valid flag being set. The MC also clears the bit on any system resets or power cycles that are not triggered by a `system control` command. This default behavior can be temporarily overridden using the MC boot flag valid bit clearing parameter.

Table 69: Set system boot options command request data

Request data byte number	Data field		
1	Parameter valid		
	[7]	1b =	Mark parameter invalid/locked
		0b =	Mark parameter valid/unlocked

Table Continued

Request data byte number	Data field	
	[6:0]	Boot option parameter selector
(2:N)	Boot option parameter data. Passing 0-bytes of parameter data allows the parameter valid bit to be changed without affecting the present parameter setting.	

Table 70: Set system boot options command response data

Response data byte number	Data field	
1	Completion code. Generic plus the command-specific completion codes:	
	80h =	Parameter not supported.
	81h =	Attempt to set the set in progress value (in parameter #0) when not in the set complete state. This completion code provides a way to recognize that another party has already claimed the parameters.
	82h =	Attempt to write read-only parameter.

Get system boot options command

This command is available to the MC.

This command is used to retrieve the boot options set by the `set system boot options` command.

Table 71: Get system boot options command request data

Request data byte number	Data field	
1	Parameter selector	
	[7]	Reserved
	[6:0]	Boot option parameter selector
2	[7:0] - Set selector. Selects a particular block or set of parameters under the given parameter selector. Write as 00h if parameter does not use a set selector.	
3	[7:0] - Block selector. Selects a particular block within a set of parameters. Write as 00h if parameter does not use a block selector. **NOTE:** There are no IPMI-specified boot options parameters that use the block selector. However, this field is provided for consistency with other configuration commands and as a placeholder for future extension of the IPMI specification.	

Table 72: Get system boot options command response data

Response data byte number	Data field		
1	Completion code. Generic plus the command-specific completion code: 80h = parameter not supported.		
2	[7:4]	Reserved	
	[3:0]	Parameter version. 1h for this specification unless otherwise specified.	
3	Parameter valid		
	[7]	1b =	Parameter marked invalid / locked
		0b =	Parameter marked valid / unlocked
	[6:0]	Boot option parameter selector	
4:N	Configuration parameter data, per **Table 73: Boot option parameters**. If the rollback feature is implemented, the MC makes a copy of the existing parameters when the set in progress state becomes asserted (see the set in progress parameter #0). While the set in progress state is active, the MC returns data from this copy of the parameters, plus any uncommitted changes that were made to the data. Otherwise, the MC returns parameter data from non-volatile storage.		

Table 73: Boot option parameters

Parameter	#	Parameter data (non-volatile unless otherwise noted)	
Set in progress (volatile)	0	Data 1 - This parameter is used to indicate when any of the following parameters are being updated, and when the updates are completed. The bit is primarily provided to alert software that some other software or utility is in the process of making changes to the data. An implementation can also elect to provide a rollback feature that uses this information to decide whether to roll back to the previous configuration information, or to accept the configuration change. If used, the roll back restores all parameters to their previous state. Otherwise, the change takes effect when the write occurs.	
		[7:2]	Reserved

Table Continued

Parameter	#	Parameter data (non-volatile unless otherwise noted)		
		[1:0]	00b =	Set complete. If a system reset or transition to powered down state occurs while set in progress is active, the MC goes to the set complete state. If rollback is implemented, going directly to set complete without first doing a commit write causes any pending write data to be discarded.
			01b =	Set in progress. This flag indicates that some utility or other software is presently doing writes to parameter data. It is a notification flag only, it is not a resource lock. The MC does not provide any interlock mechanism that would prevent other software from writing parameter data.
			11b =	Reserved
MC boot flag valid bit clearing (semi-volatile)	3	Data 1 — MC boot flag valid bit clearing. Default = 00000b.		
		[7:5]	Reserved	
		[4]	1b =	Do not clear valid bit on reset/power cycle caused by PEF.
		[3]	1b =	Do not automatically clear boot flag valid bit if `chassis control` command is not received within 60-second timeout (countdown restarts when a `chassis control` command is received).
		[2]	1b =	Do not clear valid bit on reset/power cycle caused by watchdog timeout.
		[1]	1b =	Do not clear valid bit on push button reset/soft-reset (such as “Ctrl-Alt-Del”).
		[0]	1b =	Do not clear valid bit on power up via power push button or wake event.
Boot info acknowledge (semi-volatile)	4	These flags are used to allow individual parties to track whether they have already seen and handled the boot information. Applications that deal with boot information should check the boot information and clear their corresponding bit after consuming the boot options data.		
		Data 1: Write mask (write-only). This field is returned as 00h when read. This is to eliminate the need for the MC to provide storage for the write mask field.		
		[7]	1b =	Enable write to bit 7 of data field

Table Continued

Parameter	#	Parameter data (non-volatile unless otherwise noted)		
		[6]	1b =	Enable write to bit 6 of data field
		[5]	1b =	Enable write to bit 5 of data field
		[4]	1b =	Enable write to bit 4 of data field
		[3]	1b =	Enable write to bit 3 of data field
		[2]	1b =	Enable write to bit 2 of data field
		[1]	1b =	Enable write to bit 1 of data field
		[0]	1b =	Enable write to bit 0 of data field
		Data 2: Boot initiator acknowledge data. The boot initiator should typically write FFh to this parameter before initiating the boot. The boot initiator may write 0's if it wants to intentionally direct a given party to ignore the boot info. This field is automatically initialized to 00h when the management controller is first powered up or reset.		
		[7]	Reserved. Write as 1b. Ignore on read.	
		[6]	Reserved. Write as 1b. Ignore on read.	
		[5]	Reserved. Write as 1b. Ignore on read.	
		[4]	0b =	OEM has handled boot information.
		[3]	0b =	SMS has handled boot information.
		[2]	0b =	OS/service partition has handled boot information.
		[1]	0b =	OS loader has handled boot information.
		[0]	0b =	BIOS/POST has handled boot information.
Boot flags (semi-volatile)	5	Data 1		
		[7]	1b =	Boot flags valid. The bit should be set to indicate that valid flag data is present. This bit may be automatically cleared based on the boot flag valid bit clearing parameter.
		[6]	0b =	Options apply to next boot only.

Table Continued

Parameter	#	Parameter data (non-volatile unless otherwise noted)	
		1b =	Options requested to be persistent for all future boots (such as requests for BIOS to change its boot settings). **NOTE:** In order to set this bit remotely (over a session), the user must execute the `set system boot options` command at Admin privilege level. In order to retain backward compatibility, this bit is automatically cleared by the MC whenever the boot flags valid bit is clear (0b). This is to avoid the possibility that this bit would already be set when an older application changes other options. Thus, this bit and the boot flags valid bit must be set simultaneously.
	[5]	BIOS boot type (for BIOS that support both legacy and EFI boots).	
		0b =	PC compatible boot (legacy).
		1b =	Extensible firmware interface boot (EFI).
	[4:0]	Reserved	
		BIOS support for the following flags is optional. If a given flag is supported, it must cause the specified function to occur in order for the implementation to be considered conformant with this specification. The following parameters represent temporary overrides of the BIOS default settings when data1[6] has value 0b (one-boot), and represent requests to persistently change the BIOS boot behavior when data1[6] has value 1b (persistent). BIOS should only use the following flags when the boot flags valid bit (data1[7]) is set (1b). If data[6] = 0b (one-boot) a value of 0 for a given data2 parameter indicates that BIOS should use its default configuration for the given option (no override) - a non-zero value requests BIOS to enter the requested state. If data[6] = 1b (persistent) BIOS is requested to change its setting according to the flag. This only applies to parameters labeled 
	Data 2		
	[7]	1b =	CMOS clear.
	[6]	1b =	Lock keyboard.

Table Continued

Parameter	#	Parameter data (non-volatile unless otherwise noted)		
		[5:2]	Boot device selector 	
			0000b =	No override
			0001b =	Force PXE
			0010b =	Force boot from default hard-drive
			1100-1110b	Reserved
		[1]	1b =	Screen blank.
		[0]	1b =	
OEM parameters (optional). Non-volatile or volatile as specified by OEM.	96:127	This range is available for special OEM configuration parameters. The OEM is identified according to the manufacturer ID field returned by the `get device ID` command.		

NOTE: Semi-volatile means that the parameter is kept across system power cycles, resets, system power on/off, and sleep state changes, but is not preserved if the management controller loses standby power or is cold reset. Parameters designated as semi-volatile are initialized to 0's upon controller power up or hard reset, unless otherwise specified.

Get POH counter command

This command is available to the MC.

IPMI provides a specification for an optional, POH counter. The management controller automatically increments non-volatile storage at the specified rate whenever the system is powered up. It is recommended that this command be implemented in the MC to provide a standardized location for this function.

The definition of powered up used in this document indicates that the power-on hours accumulate whenever the system is in the operational (S0) state. An implementation may elect to increment power-on hours in the S1 and S2 states as well.

Clear or set commands are not specified for this counter because the counter is most typically used for warranty tracking or replacement purposes.

Table 74: Get POH counter command response data

Response data byte number	Data field
1	Completion code
2	Minutes per count
3:6	Counter reading. LS byte first.

When the system is powered down between counts, the counter either picks up incrementing at the offset at which the power down occurred, or starts counting at 0 minutes from the last counter reading, depending on the choice of the implementer. In any case, the time does not get rounded up to the next count as a result of powering down between counts.

Event commands

The sensor/event network function is used for device functionality related to the transmission, reception, and handling of event messages and platform sensors. An event message is actually a sensor/event message with a command byte of '02h'. The request is also referred to as an event request message, while the corresponding response is referred to as an event response message.

Set event receiver command

This command tells a controller where to send event messages. The slave address and LUN of the event receiver must be provided. A value FFh for the event receiver slave address disables event message generation entirely. This command is only applicable to management controllers that act as IPMB event generators.

A device that receives a `set event receiver` command re-arms event generation for all its internal sensors. This means internally re-scanning for the event condition and updating the event status based on the result. This causes devices that have any pre-existing event conditions to transmit new event messages for those events.

A reading/state unavailable (formerly initial update in progress) bit is provided with the `get sensor reading` and `get sensor event status` commands to help software avoid getting incorrect event status due to a re-arm. For example, a controller only scans for an event condition once every four seconds. Software that accessed the event status using the `get sensor reading` command could see the wrong status for up to four seconds before the event status would be correctly updated. A controller that has slow updates must implement the reading/state unavailable bit, and should not generate event messages until the update has completed. Software should ignore the event status bits while the reading/state unavailable bit is set.

Table 75: Set event receiver command request and response data

Request data byte number	Data field
1	Event receiver slave address. 0FFh disables event message generation. Otherwise: • [7:1] - IPMB (I^2C) slave address • [0] - always 0b when [7:1] holds I^2C slave address
2	• [7:2] - reserved • [1:0] - event receiver LUN

Response data byte number	Data field
1	Completion code

Get event receiver command

This command is used to retrieve the present setting for the event receiver slave address and LUN. This command is only applicable to management controllers that act as IPMB event generators.

Table 76: Get event receiver command response data

Response data byte number	Data field
1	Completion code
2	Event receiver slave address. 0FFh indicates event message generation has been disabled. Otherwise: • [7:1] - IPMB (I^2C) slave address • [0] - always 0b when [7:1] holds I^2C slave address
3	• [7:2] - reserved • [1:0] - event receiver LUN

Platform event message command

This command is a request for the MC to process the event data that the command contains. Typically, the data is logged to the SEL. Depending on the implementation, the data may also go to the event message buffer and processed by PEF.

Table 77: Platform event message command request and response data

IPMB messaging (IPMB, LAN, Serial/Modem, PCI Mgmt. Bus)		System interface	
Request data byte number	**Data field**	**Request data byte number**	**Data field**
—	Generator ID (RqSA, RqLUN)	1	Generator ID
1	EvMRev	2	EvMRev
2	Sensor type	3	Sensor type
3	Sensor #	4	Sensor #
4	Event Dir | Event type	5	Event Dir | Event type
5	Event data 1	6	Event data 1
6	Event data 2	7	Event data 2
7	Event data 3	8	Event data 3

IPMB messaging (IPMB, LAN, Serial/Modem, PCI Mgmt. Bus)		System interface	
Response data byte number		**Response data byte number**	
1	Completion code	1	Completion code

The generator ID field is a required element of an event request message. For IPMB messages, this field is equated to the requester's slave address and LUN fields. Thus, the generator ID information is not carried in the data field of an IPMB request message.

For system side interfaces, do not overlay the generator ID field with the message source address information. It should be specified as being carried in the data field of the request.

PEF and Alerting commands

The commands in this section are related to Platform Event Filtering (PEF) and Alerting.

Get PEF Capabilities command

This command returns information about the implementation of PEF on the BMC.

Table 78: Get PEF Capabilities command response data

Response data byte number	Data field
1	Completion code
2	PEF Version (BCD encoded, LSN first, 51h for this specification.)
3	**[7]** 1b=OEM event record filtering supported **[6]** 1b=Reserved **[5]** 1b=Diagnostic interrupt **[4]** 1b=OEM action **[3]** 1b=Power cycle **[2]** 1b=Reset **[1]** 1b=Power down **[0]** 1b=Alert
4	Number of event filter table entries (1 based)

Arm PEF Postpone Timer command

This command is used by software to enable and arm the PEF Postpone Timer. The command can also be used by software to disable PEF indefinitely during run-time. Once enabled, the timer automatically starts counting down whenever the last software-processed event Record ID is for a record that is not equal to the most recent (last) SEL record. The countdown will begin immediately if the Record IDs are already different when the timer is armed.

In order to keep the PEF Postpone Timer from expiring, software must use the Set Last Processed Event ID command to update the last software-processed Record ID to match the value for the last SEL record. This will cause the BMC to stop the timer and rearm it to start counting down from the value that was passed in the Arm PEF Postpone Timer command.

The Get Last Processed Event ID command can be used to retrieve the present value for the last SEL record's

Record ID, the last BMC-processed Record ID, and the last software-processed Record ID.

Table 79: Arm PEF Postpone Timer command request data

Request data byte number	Data field
1	**[7:0]** PEF Postpone Timeout, in seconds. 01h is 1 second. **00h** Disable Postpone Timer (PEF will immediately handle events, if enabled). The BMC automatically disables the timer whenever the system enters a sleep state, is powered down, or reset. **01h—FDh** Arm timer. Timer will automatically start counting down from given value when the last-processed event record ID is not equal to the last received event's record ID. **FEh** Temporary PEF disable. The PEF Postpone timer does not countdown from the value. The BMC automatically re-enables PEF (if enabled in the PEF configuration parameters) and sets the PEF Postpone timeout to 00h whenever the system enters a sleep state, is powered down, or reset. Software can cancel this disable by setting this parameter to 00h or 01h–FDh. **FFh** Get present countdown value.

Table 80: Arm PEF Postpone Timer command response data

Response data byte number	Data field
1	Completion code
2	Present timer countdown value

Set PEF Configuration Parameters Command

This command is used for setting parameters such as PEF enable/disable and for entering the configuration of the Event Filter table and the Alert Strings.

Table 81: Set PEF Configuration command request data

Request data byte number	Data field
1	Parameter selector **[7]** Reserved **[6:0]** Parameter selector
2:N	Configuration parameter data, per **Get PEF Configuration Parameters Command**.

Table 82: Set PEF Configuration command response data

Response data byte number	Data field
1	Completion Code. Generic plus the following command-specific completion codes: **80h** Parameter not supported. **81h** Attempt to set the 'set in progress' value (in parameter #0) when not in the 'set complete' state. (This completion code provides a way to recognize that another party has already 'claimed' the parameters) **82h** Attempt to write read-only parameter **83h** Attempt to read write-only parameter

Get PEF Configuration Parameters Command

This command is used for retrieving the configuration parameters from the Set PEF Configuration command.

Table 83: Get PEF Configuration Parameters command request data

Request data byte number	Data field
1	Parameter selector **[7]** • 1b=Get parameter revision only. • 0b=Get parameter **[6:0]** Parameter selector
2	Set selector (00h if parameter does not require a set selector)
3	Block selector (00h if parameter does not require a block number)

Table 84: Get PEF Configuration Parameters command response data

Response data byte number	Data field
1	Completion Code. Generic plus the following command-specific completion codes: **80h** Parameter not supported.
2	**[7:0]** Parameter revision. Format: MSN = present revision. LSN = oldest revision with which parameter is backward compatible.
3:N	Configuration parameter data, per the following *PEF configuration parameters* table. If the rollback feature is implemented, the BMC makes a copy of the existing parameters when the 'set in progress' state becomes asserted (See the Set in Progress parameter #0). While the 'set in progress' state is active, the BMC will return data from this copy of the parameters, plus any uncommitted changes that were made to the data. Otherwise, the BMC returns parameter data from non-volatile storage. **NOTE:** Data bytes 3:N are not returned when the 'get parameter revision only' bit is 1b.

Table 85: PEF configuration parameters

Parameter	#	Parameter Data
Set In Progress (volatile)	0	**Data 1** This parameter is used to indicate when any of the following parameters are being updated, and when the updates are completed. The bit is primarily provided to alert software than some other software or utility is in the process of making changes to the data. An implementation can also elect to provide a 'rollback' feature that uses this information to decide whether to 'roll back' to the previous configuration information, or to accept the configuration change. If used, the roll back shall restore all parameters to their previous state. Otherwise, the change shall take effect when the write occurs. **[7:2]** Reserved **[1:0]** **00b** Set complete. If a system reset or transition to powered down state occurs while 'set in progress' is active, the BMC will go to the 'set complete' state. If rollback is implemented, going directly to 'set complete' without first doing a 'commit write' will cause any pending write data to be discarded. **01b** Set in progress. This flag indicates that some utility or other software is presently doing writes to parameter data. It is a notification flag only, it is not a resource lock. The BMC does not provide any interlock mechanism that would prevent other software from writing parameter data while. **10b** Commit write (optional). This is only used if a rollback is implemented. The BMC will save the data that has been written since the last time the 'set in progress' and then go to the 'set in progress' state. An error completion code will be returned if this option is not supported. **11b** Reserved
PEF control (non-volatile)	1	**Data 1** **[7:4]** Reserved **[3]** PEF Alert Startup Delay disable. (optional) • 1b = Enable PEF Alert Startup delay • 0b = Disable PEF startup delay.

Table Continued

Parameter	#	Parameter Data
		[2] PEF Startup Delay disable. (optional) An implementation that supports this bit should also provide a mechanism that allows the user to Disable PEF in case the filter entries are programmed to cause an 'infinite loop' of PEF actions (such as system resets or power cycles) when the PEF startup delay is disabled. If this bit is not implemented the PEF startup delay must always be enabled. • 1b = Enable PEF startup delay on manual (pushbutton) system power-ups (from S4/S5) and system resets (including system resets initiated by PEF). • 0b = Disable PEF startup delay. **[1]** **1b** Enable event messages for PEF actions. If this bit is set, each action triggered by a filter will generate an event message for the action. These allow the occurrence of PEF- triggered actions to be logged (if event logging is enabled). The events are logged as System Event Sensor 12h, offset 04h. See Table 2, Sensor Type Codes.) These event messages are also subject to PEF. **0b** Disable event messages for PEF actions. **[0]** **1b** Enable PEF. **0b** Disable PEF.

Table Continued

Parameter	#	Parameter Data
PEF Action global control (non-volatile)	2	**Data 1** **[7:6]** Reserved **[5]** 1b = Enable diagnostic interrupt **[4]** 1b = Enable OEM action **[3]** 1b = Enable power cycle action (No effect if power is already off) **[2]** 1b = Enable reset action **[1]** 1b = Enable power down action **[0]** 1b = Enable Alert action
PEF Startup Delay (optional, non-volatile)	3	**Data 1** Time to delay PEF after a system power-ups (from S4/S5) and resets. Default = 60 seconds. If this parameter is not provided, the default PEF Startup Delay must be implemented. Enable/ disable of the delay is configured using the PEF Control parameter, above. If this parameter is supported, a 00h value can also be used to disable the delay if necessary. **NOTE:** An implementation that supports this parameter should also provide a mechanism that allows the user to Disable PEF in case the filter entries are programmed to cause an 'infinite loop' of PEF actions under the situation where this parameter is set to too short an interval to allow a user to locally disable PEF. An implementation is allowed to force this parameter to a minimum, non-zero value. PEF Startup Delay [7:0] - PEF Startup Delay in seconds, +/- 10%. 1-based. 00h = no delay.

Table Continued

Parameter	#	Parameter Data
PEF Alert Startup Delay (optional, non-volatile)	4	**Data 1** Time to delay Alerts after system power-ups (from S4/S5) and resets. Default = platform-specific. 60-seconds typical, though may be longer on systems that require more startup time before user can take action to disable PEF. If this parameter is not provided, a default PEF Startup Delay, appropriate for the platform, must be implemented. Enable/disable of the delay can also be optionally configured using the PEF Control parameter, above. An implementation can separately implement this parameter and/or the enable/disable bit. PEF Alert Delay [7:0] - PEF Alert Startup Delay in seconds, +/- 10%. 1-based. 00h = No delay.
Number of Event Filters (READ ONLY)	5	Number of event filters supported. 1-based. This parameter does not need to be supported if Alerting is not supported. [7:0] - Number of event filter entries. 0 = Alerting not supported.
Event Filter Table (non-volatile)	6	**Data 1** - Set Selector = Filter number. [7] - Filter number. 1-based. 00h = Reserved. **Data 2:21** - Filter data
Event Filter Table Data 1 (non-volatile)	7	This parameter provides an aliased access to the first byte of the event filter data. This is provided to simplify the act of enabling and disabling individual filters by avoiding the need to do a read-modify-write of the entire filter data. **Data 1** - Set Selector = Filter number [7:0] - Filter number. 1-based. 00h = Reserved. **Data 2** - Data byte 1 of event filter data
Number of Alert Policy Entries (READ ONLY)	8	Number of alert policy entries supported. 1-based. This parameter does not need to be supported if Alerting is not supported. [7] - Reserved [6:0] - Number of alert policy entries. 0 = alerting not supported.

Table Continued

Parameter	#	Parameter Data
Alert Policy Table (non-volatile)	9	**Data 1** - Set Selector = entry number [7] - Reserved [6:0] - Alert policy entry number. 1-based. **Data 2:4** - Entry data
System GUID (non-volatile)	10	**Data 1** Used to fill in the GUID field in a PET Trap. **[7:1]** Reserved **[0]** **1b** BMC uses following value in PET Trap. **0b** BMC ignores following value and uses value returned from Get System GUID command instead. **2:17** - System GUID
Number of Alert Strings (READ ONLY)	11	Number of alert strings supported in addition to Alert String 0. 1-based. This parameter does not need to be supported if Alerting is not supported. [7] - Reserved [6:0] - Number of alert strings.

Table Continued

Parameter	#	Parameter Data
Alert String Keys (volatile & non-volatile, see description)	12	Sets the keys used to look up Alert String data in PEF. This parameter does not need to be supported if Alerting is not supported. **Data 1** - Set Selector = Alert string selector. [7] - Reserved. [6:0] - String selector. • 0 = Selects volatile string parameters • 01h-7Fh = Non-volatile string selectors PEF uses the following Event Filter Number and the Alert String Key fields to look up the string associated with a particular event. String 0 is a special, volatile string reserved for use by the Alert Immediate command. The following two fields are used by PEF to look up a particular Alert String based on information obtained from the alert policy entry. The fields should typically be set to 0's (unspecified) for string selector 0. PEF will scan the values for string 0 when doing a look up, so the string 0 values can be set to non-zero values for PEF testing/debug purposes in order to avoid writes to non-volatile storage. **Data 2** - Event Filter Number [7] - Reserved. [6:0] - Filter number. 1-based. 00h = unspecified. **data 3** - Alert String Set [7] - Reserved [6:0] -Set number for string. 1-based. 00h = unspecified.

Table Continued

Parameter	#	Parameter Data
Alert Strings (volatile & non-volatile)	13	Sets the Alert String data. The string data that should be used is dependent on the Channel and Alert Type. This parameter does not need to be supported if Alerting is not supported. For Dial paging, the BMC automatically follows the string with a <CR> (carriage return) character when sending it to the modem. For TAP paging the string corresponds to 'Field 2', the Pager Message. Note that while the string accepts 8-bit ASCII data, the TAP implementation only supports 7-bit ASCII. The BMC shall automatically zero the 8th bit when transmitting the string during TAP paging. String 0 is a special, volatile string reserved for use by the Alert Immediate command. **Data 1** - Set Selector = string selector. [7] - Reserved. [6:0] - String selector. 0 = Selects volatile string 01h-7Fh = Non-volatile string selectors **data 2** - Block Selector = string block number to set, 1 based. Blocks are 16 bytes. **data 3:N -** String data. Null terminated 8-bit ASCII string. 16-bytes max. per block.
Number of Group Control Table entries (READ ONLY)	14	**Data 1** - Number of group control table entries. 1-based (4 min, 8 max) **NOTE:** Number of Group Control Table entries is optional. Present if BMC supports automatic ICMB Group Power Control.

Table Continued

Parameter	#	Parameter Data
Group Control Table	15	**data 1** - Set Selector = group control table entry selector. [7] - Reserved. [6:0] - Group control table entry selector. **data 2** - **[7:6]** Reserved **[5]** Request/Force **0b** Request control operation. A requested operation will only complete once the same operation has been requested for all control groups and all enabled control members for the given chassis. **1b** Force control operation. A forced operation will occur regardless of whether the same operation has been requested for all control groups and all enabled control membership for the given chassis. **[4]** Immediate/Delayed. Selects whether the BMC requests an immediate or delayed control operation. Note: whether this operation is initiated at the time the command is received is dependent on the request/force bit, see above. **0b** immediate control. BMC sends command that requests an immediate control operation. **1b** delayed control. BMC sends control command to request a delayed control operation. This is conditioned by the request/force bit. **[3:0]** Channel Number (channel number for ICMB that group control operation is to be delivered over) **Data 3:** Group ID 0 (1-based) **00h** unspecified **FFh** all groups **Data 4:**

Table Continued

Parameter	#	Parameter Data

Member ID 0 (0-based)

[7:5]

Reserved

[4]

0b

Enable member ID check.

1b

Disable member ID check.

NOTE: The enable/disable member ID check bit controls whether a control request for the group is checked against the enabled members or not. If Member ID Check is disabled, then a control request to the groups will automatically be 'logged' for that group. Note, however, that the requested control state must match for all enabled groups in order for it to take effect.

[3:0]

member ID. ID of this chassis within specified group. (value is ignored if Group ID 0 = FFh)

data 5:

Group ID 1 (1-based)

00h

unspecified

FFh

all groups

Data 6:

Member ID 1 (0-based)

[7:5]

Reserved

[4]

0b

Enable member ID check.

1b

Disable member ID check. See note below.

[3:0]

Member ID. ID of this chassis within specified group. (value is ignored if Group ID 0 = FFh)

Data 7:

Group ID 2 (1-based)

Table Continued

Parameter	#	Parameter Data

00h

Unspecified

FFh

All groups

data 8:

Member ID 2 (0-based)

[7:5]

Reserved

[4]

0b

Enable member ID check.

1b

Disable member ID check. See note below.

[3:0]

Member ID. ID of this chassis within specified group. (value is ignored if Group ID 0 = FFh)

Data 9:

Group ID 3 (1-based)

00h

Unspecified

FFh

All groups

Data 10:

Member ID 3 (0-based)

[7:5]

Reserved

[4]

0b

Enable member ID check.

1b

Disable member ID check. See note below.

[3:0]

Member ID. ID of this chassis within specified group. (value is ignored if Group ID 0 = FFh)

Data 11: -

Retries and Operation

Table Continued

Parameter	#	Parameter Data
		[7] Reserved **[6:4]** Number of times to retry sending the command to perform the group operation [For ICMB, the BMC broadcasts a Group Chassis Control command] (1-based) **[3:0]** Operation **0h** Power down. Force system into soft off (S4/S45) state. This is for 'emergency' management power down actions. The command does not initiate a clean shut-down of the operating system prior to powering down the system. **1h** Power up. **2h** Power cycle (optional). This command provides a power off interval of at least 1 second. **3h** Hard reset. Some systems may accept this option even if the system is in a state (e.g. powered down) where resets are unavailable. **4h** Pulse Diagnostic Interrupt. (optional) Pulse a version of a diagnostic interrupt that goes directly to the processor(s). This is typically used to cause the operating system to do a diagnostic dump (OS dependent). The interrupt is commonly an NMI on IA-32 systems and an INIT on Intel Itanium processor based systems. **5h** Initiate a soft-shutdown of OS via ACPI by emulating a fatal overtemperature. (optional) **NOTE:** • Group Control Table is optional, non-volatile. Present if BMC supports automatic ICMB Group Power Control. • The enable/disable member ID check bit controls whether a control request for the group is checked against the enabled members or not. If Member ID Check is disabled, then a control request to the groups will automatically be 'logged' for that group. Note, however, that the requested control state must match for all enabled groups in order for it to take effect.
OEM Parameters (optional. Non-volatile	96:127	This range is available for special OEM configuration parameters. The OEM is identified according to the Manufacturer ID field returned by the Get Device ID command.

Parameter	#	Parameter Data
or volatile as specified by OEM)		

Set Last Processed Event ID command

This command is used to set the Record ID for the last event that was processed by system software. For test and debug purposes, it can also be used to set the Record ID for the last event processed by the BMC. The Last Processed Event ID value is automatically set to FFFFh whenever the SEL is cleared using the Clear SEL command. If the Delete SEL Entry command is used to either clear the SEL or delete the last event, software must set the Last Processed event manually by using the Set Last Processed Event ID command.

Of the two Record IDs (software-processed or BMC-processed) PEF uses the Record ID for the most recent event that was added to the SEL as the indicator of events that have yet to be processed. Both the last BMC-processed and last software-processed IDs are kept in NV storage.

Table 86: Set Last Processed Event ID command request data

Request data byte number	Data field
1	**[7:1]** Reserved **[0]** **0b** Set Record ID for last record processed by software. **1b** Set Record ID for last record processed by BMC.
2:3	Record ID. LS-byte first.

Table 87: Set Last Processed Event ID command response data

Response data byte number	Data field
1	Completion Code. 81h = Cannot execute command, SEL erase in progress.

Get Last Processed Event ID command

This command is used to retrieve the Record ID for the last event that was processed by system software and the BMC.

Table 88: Get Last Processed Event ID command

Response data byte number	Data field
1	Completion code. 81h = Cannot execute command, SEL erase in progress.
2:5	Most recent addition timestamp. LS byte first.
6:7	Record ID for last record in SEL. Returns FFFFh if SEL is empty.
8:9	Last SW Processed Event Record ID.
10:11	Last BMC Processed Event Record ID. Returns 0000h when event has been processed but could not be logged because the SEL is full or logging has been disabled.

Alert Immediate command

This command is used to send an alert to the destination specified by the destination selector. The kind of alert that will be sent is determined by Destination Type associated with the destination. Alerts that are initiated via this command are never logged as events. This command is to support utilities or BIOS setup options that allow the user to test their alerting configuration for a given destination. The command can also be used by system software as a run-time mechanism to trigger the delivery of an alert.

These alerts are not subject to the Page Blackout intervals, although an alert must complete before the next Alert Immediate command will be accepted. Alert Immediate commands are also rejected with an error completion code if an IPMI messaging session or automatic page is already in progress.

Table 89: Alert Immediate command request data

Request data byte number	Data field
1	Channel number. (This value is required to select which configuration parameters are to be used to send the Alert.) [7:4] - reserved [3:0] - Channel number. Note: BMC stores the 'Alert immediate status' for each channel that can send alert.
2	Destination Selector/ Operation [7:6] - Operation 00b = Initiate alert 01b = Get Alert Immediate status 10b = Clear Alert Immediate status (sets status to 00h) 11b = reserved [5:4] - Reserved [3:0] - destination selector. Selects which alert destination should go to. 0h = use volatile destination info. 1h-Fh = non-volatile destination. **Note: If Operation is 'Get Alert Immediate status' or 'Clear Alert Immediate Status' bits [3:0] are reserved.**
3	Alert String Selector Selects which Alert String, if any, to use with the alert. [7] - 0b = don't send an Alert String 1b = send Alert String identified by following string selector. [6:0] - string selector. 000_0000b = use volatile Alert String. 01h-7Fh = non-volatile string selector.
The following "Platform Event Parameters" (bytes 4:11) can be used to fill in the corresponding event data fields of a Platform Event Trap. When supported, all bytes (4:11) must be supplied. Implementation of this capability is OPTIONAL but highly recommended for IPMI v2.0 implementations.	
4	Generator ID
5	EvMRev
6	Sensor Type
7	Sensor #

Table Continued

Request data byte number	Data field
8	Event Dir | Event Type
9	Event Data 1
10	Event Data 2
11	Event Data 3

Table 90: Alert Immediate command response data

Response data byte number	Data field
1	Completion Code. Generic codes, plus following command-specific completion codes: • 81h = Alert Immediate rejected due to alert already in progress. • 82h = Alert Immediate rejected due to IPMI messaging session active on this channel. • 83h = Platform Event Parameters (4:11) not supported.
Following byte is only returned when Operation in request is set to “Get Alert Immediate status”	
2	Alert Immediate Status SMS can poll this status to determine present state of the immediate alert. 00h = No status. Note: A BMC implementation is allowed (but not required) to abort the Alert Immediate command due to a channel parameter configuration, power, or reset state changes that occur while the Alert Immediate command is being processed. In which case the BMC will return the ‘no status’ state. 01h = Alert was Normal End. This will also be returned if one or more attempts failed, but the last attempt was successful. 02h = “Call Retry” (Dial connection) retries failed. 03h = Alert failed due to timeouts waiting for acknowledge on all retries. FFh = Alert by this command is in progress. Status pending.

PET Acknowledge command

This message is used to acknowledge a Platform Event Trap (PET) alert. PET alerts are SNMP Traps that are delivered by LAN or PPP alerting, see [PET] for more info. The PET Acknowledge message is an IPMI Request Message that is sent by the remote console that has received the trap.

NOTE:

The PET Acknowledge command does not require that an IPMI Messaging session be established with the BMC. It is in the same class as the Get Channel Authentication Capabilities command. In addition, if Alerting is enabled and the configuration parameters for the Alert Destination require the PET Alert to be acknowledged, the BMC will wait for and accept the PET Acknowledge command until the selected retry interval has expired, even if IPMI Messaging is not available according to the present Access Mode for the channel. For systems using Serial Port Sharing, the BMC will stay switched to the serial connector while waiting for the PET Acknowledge.

Table 91: PET Acknowledge command request data

Request data byte number	Data field
1:2	Sequence Number. Value from the Sequence Number field of the PET. LS- byte first .
3:6	Local Timestamp. Value from the Local Timestamp field of the PET. LS-byte first [1].
7	Event Source type. From corresponding field in the PET.
8	Sensor Device. From corresponding field in the PET.
9	Sensor Number. From corresponding field in the PET.
10:12	Event Data 1:3. From corresponding field in the PET.
1	Completion Code.

[1] The sequence number and local timestamp fields in the actual PET on the network are in network byte order, therefore filling in these values may require software to re-order the bytes as they get them from the trap.

Table 92: PET Acknowledge command response data

Response data byte number	Data field
1	Completion Code.

SEL commands

The SEL is a non-volatile repository for system events and certain system configuration information. The device that fields these commands is referred to as the SEL device. Event message information is normally written into the SEL after being received by the event receiver functionality in the event receiver device.

The SEL device commands are structured in such a way that the device can be separated from the event receiver device. In this instance, the event receiver device must send the appropriate `Add sel entry` message directly to the SEL device or pass the equivalent request through an intermediary.

SEL entries have a unique record id field, used for retrieving log entries from the SEL. SEL reading is done in a random access manner, that is, SEL entries are read in any order as long as the record id is known.

NOTE:

SEL record id's 0000h and FFFFh are reserved for functional use and are not legal id values. Record ids are handles and are not required to be sequential or consecutive. Applications should not assume that the SEL record id will follow any particular numeric order.

SEL records are stored as ordered lists. Appending and deleting individual entries does not change the access order.

SEL device commands

The SEL device can be implemented as a separate device from the event receiver and event generator devices. If this is done, it is up to the implementer to create the method by which even messages are passed from the even receiver device to the SEL device.

Get SEL info command

This command return the number of entries in the SEL, SEL command version and the timestamp for the most recent entry and delete/clear. The most recent addition timestamp field returns the timestamp for the last add or log operation while the most recent erase field returns the timestamp for the last delete or clear operation.

These timestamps are independent of timestamps that may be returned by other commands, such as those returned by the `Get sdr repository info` command. The timestamp reflects when the most recent SEL add or erase occurred, and not when the last add or erase occurred on the physical storage device.

For example, the SEL `Info most recent addition` timestamp would reflect the last time a new event was added to the SEL. This would be independent of the `Info most recent addition` time for an SDR even if the implementation elected to implement the SEL and SDR repository in the same storage device.

Table 93: Get SEL info command request and response data

Request data byte number	Data field
1	Completion code • 81h = cannot execute command • SEL erase in progress
2	SEL version — version number of the SEL command set for this SEL Device. • 51h • BCD encoded with bits 7:4 holding the least significant digit of the revision and bits 3:0 holding the most significant bits.
3	Entries LS byte — number of log entries in SEL, LS byte
4	Entries MS byte — number of log entries in SEL, MS byte
5:6	Free space in bytes, LS byte first. FFFFh indicates 65535 or more bytes of free space are available.

Table Continued

Request data byte number	Data field
7:10	Most recent addition timestamp. • LS byte first. • Returns FFFF_FFFFh if no SEL entries have ever been made or if a component update or error caused the retained value to be lost.
11:14	Most recent erase timestamp. Last time that one or more entries were deleted from the log. LS byte first.
15	Operation support • [7] — Overflow flag. 1=events have been dropped due to lack of space in the SEL. • [6:4] — reserved. Write as 000 • [3] — 1b = `Delete SEL` command supported • [2] — 1b = Partial `Add SEL Entry` command supported • [1] — 1b = `Reserve SEL` command supported • [0] — 1b = `Get SEL Allocation` information command supported

Reserve SEL command

This command sets the present owner of the SEL as identified by the software id or by the requesters slave address from the command. The reservation process provides a limited amount of protection on repository access from the IPMB when records are deleted or incrementally read.

The `reserve sel` command provides helps prevent the wrong record from being deleted. It includes a mechanism that prevents the SEL from being cleared when a new event is received in addition to preventing receipt of incorrect data during incremental reads.

The `reserve sel` does not guarantee access to the SEL. Essentially, this command prevents requesters from causing deadlocking.

A reservation id value is returned in response to this command. This value is required in other requests, such as the `clear sel` command. This commands will not execute unless the correct reservation id value is provided.

The reservation id is used in the following manner. Suppose an application wishes to clear the SEL. The application would first reserve the repository by issuing a `reserve sel` command. The application would then check that all SEL entries have been handled prior to issuing the `clear sel` command.

If a new event is placed in the SEL after records were checked, but before the `clear sel` command, it is possible for the event to be lost. However, the addition of a new event to the SEL causes the present reservation id to be cancelled. This would prevent the `clear sel` command from executing. If this occurred, the application would repeat the reserve check clear process until successful.

Table 94: Reserve SEL command request and respond data

Request data byte number	Data field
1	Completion code. 81h = cannot execute command, SEL erase in progress
2	Reservation id, LS byte 0000h reserved
3	Reservation id, MS byte

Reservation restricted commands

A requester must issue a `reserve sel` command prior to issuing any of the following SEL commands. The `reserve sel` command is reissued if a reservation is canceled. These commands are rejected if the requesters reservation is cancelled.

- `delete sel entry` command
- `clear sel` command
- `get sel entry` command, if `get` is from an offset other than 00h.
- partial `add sel entry` command

If the given reservation is canceled, a `reservation canceled` completion code is returned in response to the above commands.

The record id associated with a given record can change between successive offset 0 `gets` to the record id. That is, the first SEL entry changes if the SEL is cleared and a new event comes in. The device accessing the SEL verifies that the retrieved record information matches the id information (timestamp, slave address, LUN, sensor id) of the event record.

Reservation cancellation

The SEL device automatically cancels the present SEL reservation after any of the following events occur:

- A SEL entry is added.
- A SEL entry is deleted, and other record ids change. That is, an implementation is allowed to cancel the reservation on any SEL entry deletion.
- The SEL is cleared.
- The SEL device is reset by hardware or cold reset command.
- A new `reserve sel` command is received.

Get SEL entry command

Use this command to retrieve entries from the SEL. The record data field in the response returns the 16 bytes of data from the SEL event record.

Table 95: Get SEL entry request data

Request data byte number	Data field
1:2	Reservation id, LS byte first. Only required for partial `get`. Use 0000h otherwise. [1]
3:4	SEL record id, LS byte first. 0000h = get first entry FFFFh — get last entry
5	Offset into record
6	Bytes to read. FFh means read entire record.

[1] The reservation id should be set to 0000h for implementations that don't implement the `reserve sel` command.

Table 96: Get SEL entry response data

Response data byte number	Data field
1	Completion code. Return an error completion code if the SEL is empty. 81h = cannot execute command, SEL erase in progress.
2:3	Next SEL record id, LS byte first (return FFFFh if the record returned is the last record.) FFFFh is not a valid record id.
4:N	Record data, 16 bytes for entire record.

Add SEL entry command

This command enables the BIOS to add records to the system event log. Normally, the SEL device and the event receiver device are incorporated into the same management controller. In this case, the BIOS or the system SMI handler adds its own events to the SEL by formatting an event message and transmitting it to the SEL device instead of using this command.

Records are added after the last record in the SEL. The SEL device adds the timestamp according to the SEL record type when it creates the record. In some cases, the timestamp bytes in the record data are ignored, there are still dummy timestamp bytes present in the data.

The record data field is passed in the request consists of all bytes of the SEL event record. The record id field that is passed in the request is just a placeholder. The record id field that was passed in the request is overwritten with a record id value that the SEL device generates before the record is stored. Depending on the record type, the entry may also be automatically timestamped. If the entry is automatically timestamped, the SEL device also overwrites the four bytes of the records timestamp field.

NOTE:

The normal mechanism for adding entries to the SEL is by an event request message to the event receiver device.

Table 97: Add SEL entry request data

Request data byte number	Data field
1:16	Record data, 16 bytes

Table 98: Add SEL entry response data

Request data byte number	Data field
1	Completion code. Generic, plus following command specific: 80h = operation not supported for this record type 81h = cannot execute command, SEL erase in progress
2:3	Record id for added record, LS byte first.

Clear SEL

This command erases all contents of the System Event Log. Since this process may take several seconds, based on the type of storage device, the command also provides a means for obtaining the status of the erasure.

Table 99: Clear SEL entry request data

Request data byte number	Data field
1:2	Reservation ID, LS Byte first.[1]
3	'C' (43h)
4	'L' (4Ch)
5	'R' (52h)
6	AAh = initiate erase. 00h = get erasure status.

[1] The reservation ID should be set to 0000h for implementations that don't implement the Reserve SEL command.

Table 100: Clear SEL entry response data

Request data byte number	Data field
1	Completion Code
2	Erasure progress. [7:4] - reserved [3:0] - erasure progress • 0h = erasure in progress. • 1h = erase completed.

SEL record type ranges

Table 101: SEL record type ranges

Record ranges	Description
00h — BFh	This range is reserved for standard SEL record types. Records are automatically timestamped unless otherwise indicated.
C0h — DFh	This range is reserved for timestamped OEM SEL records. These records are automatically timestamped by the SEL device.
E0h — FFh	This range is reserved for non-timestamped OEM SEL records. The SEL device does not automatically timestamp these records. The four bytes passed in the byte locations for the timestamp are directly entered into the SEL.

Get SEL time command

This command returns the time from the SEL device. This time is used by the SEL device for event timestampting.

Table 102: Get SEL time command request and respond data

Response data byte number	Data field
1	Completion code
2:5	Present timestamp clock reading. LS byte first.
Request data byte number	**Data field**
1:4	Time in four byte format. LS byte first.

Set SEL time command

This command initializes the time in the SEL device. This time is used by the SEL device for event timestamping.

Table 103: Set SEL time command response data

Response data byte number	Data field
1	Completion code.

SDR repository device commands

The following sections describe the commands that an SDR repository device provides for accessing the SDR repository. The commands are designed to simplify the SDR repository device's implementation by pushing back intelligence to higher-level software where possible. The SDR repository device is not intended to be a database engine. Thus, the SDR access commands do not include automatic search functions. It is recommended that an application read the SDR repository into a RAM buffer and work from that copy (keeping track of the SDR timestamp to check for possible changes to the SDR repository). The general procedure for reading SDRs from the SDR repository is described under the `get sdr` command.

As with event messages, the commands are designed so that the SDR repository device is isolated and does not need to know the content and format of the SDR records themselves.

SDR record IDs

In order to generalize SDR access, sensor data records are accessed using a record ID number. There are a fixed number of possible record IDs for a given implementation of the SDR repository. The most common implementation of record IDs is as a value that translates directly to an index or offset into the SDR repository. However, it is also possible for an implementation to provide a level of indirection, and implement record IDs as handles to the SDRs.

Record ID values may be recycled so that the record ID of a previously deleted SDR can be used as the record ID for a new SDR. The requirement is that, at any given time, the record IDs are unique for all SDRs in the repository.

Record IDs can be reassigned by the SDR repository device as needed when records are added or deleted. An application that uses a record ID to directly access a record should always verify that the retrieved record information matches up with the ID information (slave address, LUN, sensor ID, and so on) of the desired sensor. An application that finds that the SDR at a given record ID has moved needs to re-enumerate the SDRs by listing them out using a series of `get sdr` commands. It is not necessary to read out the full record data to see if the record ID for a particular record has changed. Software can determine whether a given record has been given a different record ID by examining just the SDR's header and record key bytes.

Get SDR repository info command

This command is available to the MC.

At the zone level, remember to issue the `SDR repository` version of the command. At any other zone, use the `device SDR` version of the command.

This command returns the SDR command version for the SDR repository. It also returns a timestamp for when the last add, delete, or clear occurred. The most recent addition timestamp field returns the timestamp for the last addition operation, while the most recent erase field returns the timestamp for the last delete or clear operation.

These timestamps are independent of timestamps that may be returned by other commands, such as those returned by the `get SEL info` command. The timestamp reflects when the most recent SDR repository add or erase occurred, not when the last add or erase occurred on the physical storage device.

For example, the SDR repository info most recent addition timestamp would reflect the last time a new record was added to the SDR repository. The SDR repository's most recent addition timestamp is always independent of the most recent addition time for the SEL - even if the SEL and SDR repository are implemented in the same physical storage device.

Table 104: Get SDR repository info command response data

Response data byte number	Data field		
1	Completion code		
2	SDR version - version number of the SDR command set for the SDR device. 51h for this specification. (BCD encoded with bits 7:4 holding the least significant digit of the revision and bits 3:0 holding the most significant bits.)		
3	Record count LS byte - number of records in the SDR repository.		
4	Record count MS Byte - number of records in the SDR repository.		
5:6	Free space in bytes, LS Byte first. 0000h indicates full, FFFEh indicates 64KB-2 or more available. FFFFh indicates unspecified.		
7:10	Most recent addition timestamp. LS byte first.		
11:14	Most recent erase (delete or clear) timestamp. LS byte first.		
15	Operation support		
	[7]	Overflow flag. 1=SDR could not be written due to lack of space in the SDR repository.	
	[6:5]	00b =	Modal/non-modal SDR repository update operation unspecified.
		01b =	Non-modal SDR repository update operation supported.
		10b =	Modal SDR repository update operation supported.
		11b =	Both modal and non-modal SDR repository update supported.
	[4]	Reserved. Write as 0b.	
	[3]	1b=	`Delete SDR` command supported.
	[2]	1b=	Partial Add SDR command supported.

Table Continued

Response data byte number	Data field		
	[1]	1b=	`Reserve SDR repository` command supported.
	[0]	1b=	`Get SDR repository allocation information` command supported.

Get SDR repository allocation info command

This command is available to the MC.

This command returns the number of possible allocation units, the amount of usable free space (in allocation units), the allocation unit size (in bytes), and the size of the largest contiguous free region (in allocation units). The allocation unit size is the number of bytes in which storage is allocated. For example, if a 20-byte record is to be added, and the SDR repository has a 16-byte allocation unit size, then the record would take up 32-bytes of storage.

The SDR repository implementation, at a minimum, provides an allocation unit size of 16 bytes or more and a maximum record size supporting a record of 64 bytes or more. Software should assume an allocation unit size of 16 bytes if this command is not implemented.

Table 105: Get SDR repository allocation info command response data

Response data byte number	Data field
1	Completion code
2	Number of possible allocation units, LS byte.
3	Number of possible allocation units, MS byte. This number indicates whether the total number of possible allocation units is equal to or less than the log size divided by the allocation unit size. 0000h indicates unspecified.
4	Allocation unit size in bytes. 0000h indicates unspecified.
5	
6	Number of free allocation units, LS byte.
7	Number of free allocation units, MS byte.
8	Largest free block in allocation units, LS byte.
9	Largest free block in allocation units, MS byte.
10	Maximum record size in allocation units.

Reserve SDR repository command

This command is available to the MC.

This command is used to set the present owner of the repository, as identified by the software ID or by the requester's slave address from the command. The reservation process provides a limited amount of protection on repository access from the IPMB when records are being deleted or incrementally read.

The `reserve SDR repository` command is provided to help prevent deleting the wrong record when doing deletes, and to prevent receiving incorrect data when doing incremental reads. It does not guarantee access to the SDR repository so that a pair of requesters could vie for access to the SDR that they alternately cancel the reservation that is held by the other - effectively deadlocking each other.

A reservation ID value is returned in response to this command which is required in other requests, such as the `delete SDR` command. These commands do not execute unless the correct reservation ID value is provided.

The reservation ID is used in the following manner. Suppose an application wishes to delete a particular record. The application would first reserve the repository by issuing a `reserve SDR repository` command. The application reads the header and key information from the record to verify that it has the correct record ID for the record. Assuming this is correct, the application issues a `delete SDR` command using the reservation ID and record ID as parameters.

If an event had occurred that changed the record IDs after the header and key information was read but before the `delete SDR` command, the `delete SDR` command could be issued with the record ID for the wrong record. However, events that change record IDs for any existing records cause the present reservation ID to be canceled. This prevents software from using an out-of-date record ID to access a record. For example, it would prevent the `delete SDR` command from executing and deleting the wrong record in case a given record ID was reassigned to a different record.

Table 106: Reserve SDR repository command response data

Response data byte number	Data field
1	Completion code
2	Reservation ID, LS byte.
3	Reservation ID, MS byte.

Reservation restricted commands

A requester must issue a `reserve SDR repository` command before issuing any of the following SDR repository commands. This command only needs to be reissued if the reservation is canceled. The following commands are rejected if the requester's reservation has been canceled.

- Delete SDR command
- Clear SDR Repository command
- Get SDR command (if a partial read)

If the given reservation has been canceled, a reservation canceled completion code is returned in response to the above commands. See **Reservation cancellation**

Since record IDs could change between offset 0 "gets" of a given record, it is the responsibility of the device accessing the repository to verify that the retrieved record information matches up with the ID information (slave address, LUN, sensor ID, and so on) of the desired sensor.

Reservation cancellation

The SDR repository device automatically cancels the present SDR repository reservation after any of the following events occur:

- An SDR record is added using the `add SDR` command so that other record IDs change. As a simplification, an implementation is allowed to cancel the reservation on any SDR record add.
- An SDR record is deleted so that other record IDs change. As a simplification, an implementation is allowed to cancel the reservation on any SDR record deletion.
- The SDR repository is cleared.
- The SDR repository device is reset (via hardware or `cold reset` command).
- A new `reserve SDR repository` command is received.

An error completion code is returned if an attempt is made to execute a command that requires a reservation ID, but the reservation ID used is not valid or current.

Get SDR command

This command is available to the MC.

This command returns the sensor record specified by record ID. The command also accepts a byte range specification that allows just a selected portion of the record to be retrieved (incremental read). The requester must first reserve the SDR repository using the `reserve SDR repository` command in order for an incremental read to an offset other than 0000h to be accepted. (It is also recommended that an application use the `get SDR repository info` command to verify the version of the SDR repository before it sends any other SDR repository commands. This is important since the SDR repository command format and operation can change between versions).

If the record ID is specified as 0000h, this command returns the record header for the first SDR in the repository. FFFFh specifies that the last SDR in the repository should be listed. If the record ID is non-zero, the command returns the information from the matching record, and the record ID for the next SDR in the repository.

An application that wishes to retrieve the full set of SDR records must first issue the `get SDR` command starting with 0000h as the record ID to get the first record. The next record ID is extracted from the response and this is then used as the record ID in a `get SDR` request to get the next record. This is repeated until the last record ID value (FFFFh) is returned in the next record ID field of the response.

A partial read from offset 0000h into the record can be used to extract the header and associated key fields for the specified sensor data record in the SDR repository. An application can use the command in this manner to get a list of what records are in the SDR and to identify the instances of each type. It can also be used to search for a particular sensor record.

NOTE: To support future extensions, applications should check the SDR version byte before interpreting any of the data that follows.

The application issuing `get SDR` commands with a non-zero value for the offset into the record field must first reserve the SDR repository by issuing a `reserve SDR repository` command.

If you issue a `get SDR` command (storage 23h) with a bytes to read size of FFh (meaning read entire record) this will cause an error in most cases, since SDRs are bigger than the buffer sizes for the typical system interface implementation. The controller therefore returns an error completion code if the number of record bytes exceeds the maximum transfer length for the interface. The completion code CAh indicates that the number of requested bytes cannot be returned. Returning this code is recommended, although a controller could also return an FFh completion code. In either case, the algorithm for handling this situation is to default to using partial reads if the read entire record operation fails (that is, if you get a non-zero completion code).

Table 107: Get SDR command request and response data

Request data byte number	Data field
1	Reservation ID, LS byte. Only required for partial reads with a non-zero offset into record field. Use 0000h for the reservation ID otherwise.
2	Reservation ID, MS byte.
3	Record ID of record being requested, LS byte.
4	Record ID of record being requested, MS byte.
5	Offset into record.
6	Bytes to read. FFh means read entire record.

Response data byte number	Data field
1	Completion code.
2	Record ID for next record, LS byte.
3	Record ID for next record, MS byte.
4:3+N	Record data.

Add SDR command

This command is available to the MC.

This command adds the specified sensor record to the SDR repository and returns its record ID. The data passed in the request must contain the SDR data in its entirety.

Table 108: Add SDR command request and response data

Request data byte number	Data field
1:N	SDR data

Response data byte number	Data field
1	Completion code

Table Continued

2	Record ID for added record, LS byte
3	Record ID for added record, MS byte

Delete SDR command

This command is available to the MC.

This command deletes the sensor record specified by record ID. The requester's ID and the reservation ID must also match the present owner of the SDR repository.

Table 109: Delete SDR command request and response data

Request data byte number	Data field
1	Reservation ID, LS byte
2	Reservation ID, MS byte
3	Record ID of record to delete, LS byte
4	Record ID of record to delete, MS byte

Response data byte number	Data field
1	Completion code
2	Record ID for deleted record, LS byte
3	Record ID for deleted record, MS byte

Clear SDR repository command

This command is available to the MC.

This command clears all records from the SDR repository and re-initializes the SDR repository subsystem. Mainly a development and production aid, use of this command should be avoided in utilities and system management software. The requester's ID and reservation ID information must also match the present owner of the SDR repository.

Table 110: Clear SDR repository command request and response data

Request data byte number	Data field
1	Reservation ID, LS byte
2	Reservation ID, MS byte
3	C (43h)

Table Continued

4	L (4Ch)
5	R (52h)
6	• AAH = initiate erase • 00h = get erasure status

Response data byte number	Data field
1	Completion code
2	Erasure progress • [7:4] - reserved • [3:0] - erasure in progress ◦ 0h = erasure in progress ◦ 1h = erase completed

Get SDR repository time command

This command returns the time from the SDR Repository Device. This time is used by the SDR Repository Device for tracking when changes to the SDR Repository have been made. The time keeping format is specified in **Timestamp format**.

Table 111: Get SDR repository command response data

Response data byte number	Data field
1	Completion code
2:5	Time is four-byte format. LS byte first.

Run initialization agent command

This command is available to the MC.

This command can be used to cause the initialization agent to run. The command can be used to check the status of the initialization agent as well.

Table 112: Run initialization agent command request and response data

Request data byte number	Data field
1	• [7:1] - reserved • [0] ◦ 1b = run initialization agent ◦ 0b = get status of initialization agent process

Response data byte number	Data field
1	Completion code
2	• [7:1] - reserved • [0] ◦ 1b = initialization completed ◦ 0b = initialization in progress

FRU inventory device commands

The FRU inventory data contains information such as the serial number, part number, asset tag, and a short descriptive string for the FRU. The contents of a FRU inventory record are specified in the Platform Management FRU Information Storage Definition.

The FRU inventory device is a logical device and is not necessarily implemented as a separate physical device. The device that contains the SDR repository device also typically holds FRU inventory information for the main system board and chassis, there may also be a separate FRU inventory device that provides access to the FRU information for a replaceable module such as a memory codule.

Get FRU inventory area info command

This command returns the overall size of the FRU inventory area for a device in bytes.

Table 113: Get FRU inventory area info command request and response data

Request data byte number	Data field
1	FRU device id. FFh = reserved

Response data byte number	Data field
1	Completion code.

Table Continued

2	FRU inventory area size in bytes, LS byte.
3	FRU inventory area size in bytes, MS byte.
4	[7:1] — reserved [0] — 0b = device is accessed by bytes, 1b = device is accessed by words

Read FRU data command

This command returns the specified data from the FRU inventory info area. This is effectively a low level direct interface to a non-volatile storage area. This means that the interface does not interpret or check any semantics or formatting for the data being accessed. The offset used in this command is a logical offset that may correspond to the physical address used in the device that provides the non-volatile storage. For example, FRU information kept in flash at physical address 1234h, however the offset 0000h would be used with this command to access the start of the FRU information. IPMI FRU device data (devices formatted per FRU) as well as process and DIMM FRU data always starts from offset 0000h unless otherwise noted.

NOTE:

While offset values are 16–bit allowing FRU devices up to 64K words, the `count to read`, `count returned` and `count written` fields are 8–bits. This is in recognition of the limitations on the size of messages. Currently IPMB messages are limited to 32–bytes total.

Table 114: Read FRU data command request and response data

Request data byte number	Data field
1	FRU device id. FFh = reserved.
2	FRU inventory offset to read, LS byte.
3	FRU inventory offset to read, MS byte. Offset is in bytes or words per device access type returned in the `get fru inventory area info` command.
4	Count to read – count is 1 based.
Response data byte number	**Data field**

Table Continued

1	Completion code. Generic, plus following command specific: • 81h = FRU device busy. The requested cannot be completed because the implementation of the logical FRU device is in a state where the FRU information is temporarily unavailable. This could be due to a condition such as a loss of arbitration if the FRU is implemented as a device on a shared bus. • Software can elect to retry the operation after at least 30 milliseconds if this code is returned. **NOTE:** It is highly recommended that management controllers incorporate built-in retry mechanisms. Generic IPMI software cannot be relied upon to take advantage of this completion code
2	Count returned – count is 1 based
3:2+N	Requested data.

Write FRU data command

This command writes the specified byte or word to the FRU inventory info area. This is a low level direct interface to a non-volatile storage area. This means that the interface does not interpret or check any semantics or formatting for the data being written. The offset used in this command is a logical offset that may correspond to the physical address used in device that provides the non-volatile storage. For example, FRU information could be kept in flash at physical address 1234h, however offset 0000h would still be used with this command to access the start of the FRU information. IPMI FRU device data (devices that are formatted per FRU) as well as processor and DIMM FRU data always starts from offset 0000h unless otherwise noted. Updating the FRU inventory data is presumed to be a system level, privileged operation. There is no requirement for devices implementing this command to provide mechanisms for rolling back the FRU inventory area in the case of incomplete or incorrect writes.

Table 115: Write FRU data command request and response data

Request data byte number	Data field
1	FRU device id. FFH = reserved
2	FRU inventory offset to write, LS byte
3	FRU inventory offset to write, MS byte
4:3+N	Data to write
Response data byte number	**Data field**
1	Completion code. Generic, plus following command specific: • 80h = write-protected offset. Cannot complete write because one or more bytes of FRU data are to a write-protected offset in the FRU device. An implementation may have allowed a partial write of the data to occur. • 81h = FRU device busy. Refer to **Read FRU data command**. for the description of this completion code.
2	Count written – count is 1 based

Sensor Device Commands

Get device SDR info command

This command returns general information about the collection of sensors in a dynamic sensor device.

NOTE:

Issuing this command without a parameter, returns LUN based device sensor information.

Regarding LUN based device sensor information, a device could implement four sensors under one LUN and twelve under another. SDR info does not return the aggregate of the sensor information, instead you must issue a `Get Device SDR Info` command for each LUN.

Table 116: Get device SDR info command request and response data

Request data byte number	Data field
1	Operation (optional) [7:1] — reserved [0] — 1b = `Get SDR count`, returns the total number of SDRs in the device. 0b = `Get sensor count`, returns the number of sensors implemented on the LUN addressed.
Response data byte number	**Data field**
1	Completion code
2	For operation = `Get sensor count` (or if byte 1 not present in the request): Number of sensors in device for the LUN this command was addressed. For operation = `Get SDR count`, the total number of SDRs in the device.

Table Continued

3	Flags: Dynamic population [7] 0b = static sensor population. The number of sensors handled by this device is fixed and a query shall return records for all sensors. 1b = dynamic sensor population. This device may have its sensor population vary during run time (any time other than during installation). Reserved [6:4] — reserved Device LUNs [3] — 1b = LUN 3 has sensors [2] — 1b = LUN 2 has sensors [1] — 1b = LUN 1 has sensors [0] — 1b = LUN 0 has sensors
4:7	Sensor population change indicator. LS byte first. Four byte timestamp, or counter. Updated or incremented each time the sensor population changes. This field is not provided if the flags indicate a static sensor population.

Get device SDR command

The `Get Device SDR` command allows SDR information for sensors and is typically implemented in a satellite management controller. It also returns SDR types in addition to 01h and 02h. This is an optional command for static sensor devices, and mandatory for dynamic sensor devices. The format action is similar to the `get sdr` command for repository devices.

NOTE: A sensor device uses consistent sensor numbers for particular sensor.

The `Get Device SDR` command includes a `reservation id` that notifies the requestor that a record may have changed during a multi-part read.

Table 117: Get device SDR command request and response data

Request data byte number	Data field
1	Reservation ID. LS Byte. Only required for partial reads with a non-zero offset into record field. Use 0000h for reservation id.
2	Reservation ID. MS Byte.
3	Record id of record to `get`, LS byte. 0000h returns the first record.

Table Continued

4	Record id of record to `get`, MS byte.
5	Offset into record.
6	Bytes to read. FFh means read entire record.

Response data byte number	Data field
1	Completion code. Generic, plus command specific: 80h = record changed. This status is returned if any of the record contents were altered since the last time the requestor issued the request with 00h for the `offset into SDR` field.
2	Record id for next record, LS byte.
3	Record id for next record, MS byte.
4:3+N	Requested bytes from record.

Reserve device SDR repository command

This command is used to obtain a `reservation id` that is part of the mechanism used to notify the requestor of record changes during a multi-part read.

Table 118: Reserve device SDR repository command request and response data

Response data byte number	Data field
1	Completion code
2	Reservation id, LS byte 0000h reserved.
3	Reservation id, MS byte
Request data byte number	**Data field**
1	Sensor number (FFh = reserved)

Get sensor thresholds command

This command retrieves the threshold for a given sensor.

Table 119: Get sensor thresholds command response data

Response data byte number	Data field
1	Completion code
2	[7:6] — reserved. Returns as 00b. Readable thresholds: This bit mask indicates which thresholds are readable [5] — 1b = upper non-recoverable threshold [4] — 1b = upper critical threshold [3] — 1b = upper non-critical threshold [2] — 1b = lower non-recoverable threshold [1] — 1b = lower critical threshold [0] — 1b = lower non-critical threshold
3	lower non-critical threshold, if present, ignore on read otherwise
4	lower critical threshold, if present, ignore on read otherwise
5	lower non-recoverable threshold, if present, ignore on read otherwise
6	upper non-critical threshold, if present, ignore on read otherwise
7	upper critical, if present, ignore on read otherwise
8	upper non-recoverable, if present, ignore on read otherwise

Set sensor event enable command

This command provides the ability to disable or enable Event Message Generation for individual sensor events. The command is also used to enable or disable sensors in their entirety using the disable scanning bit.

A typical sensor will come up with Event Messages (EvM) enabled for all thresholds/states. Sensors are not required to have individual or per-event Event Message enables. The type of enable/disable support that a sensor provides can be obtained from the Sensor Data Record for the sensor.

Note that internal event flags and scanning will continue even though Event Message generation is disabled, unless sensor scanning is disabled.

Table 120: Set sensor event enable command request and response data

Request data byte number	Data field
1	Sensor number (FFh = reserved)

Table Continued

2	[7]	0b	Disable all Event Messages from this sensor [does not impact individual enable/ disable status]
	[6]	0b	Disable scanning on this sensor
	[5:4]	00b	do not change individual enables
		01b	Enable selected event messages
		10b	Disable selected event messages
		11b	Reserved
	[3:0]	Reserved	
3	For sensors with threshold based events:		
	[7]	1b	Assertion event for upper non-critical going high
	[6]	1b	Assertion event for upper non-critical going low
	[5]	1b	Assertion event for lower non-recoverable going high
	[4]	1b	Assertion event for lower non-recoverable going low
	[3]	1b	Assertion event for lower critical going high
	[2]	1b	Assertion event for lower critical going low
	[1]	1b	Assertion event for lower non-critical going high
	[0]	1b	Assertion event for lower non-critical going low
	For sensors with discrete events		
	[7]	1b	Assertion event for state 7
	[6]	1b	Assertion event for state 6
	[5]	1b	Assertion event for state 5
	[4]	1b	Assertion event for state 4
	[3]	1b	Assertion event for state 3
	[2]	1b	Assertion event for state 2

Table Continued

	[1]	1b	Assertion event for state 1
	[0]	1b	Assertion event for state 0
4	For sensors with threshold based events		
	[7:4]	Reserved. Written as 0000b.	
	[3]	1b	Assertion event for upper non-recoverable going high
	[2]	1b	Assertion event for upper non-recoverable going low
	[1]	1b	Assertion event for upper critical going high
	[0]	1b	Assertion event for upper critical going low
	For sensors with discrete events (00h otherwise)		
	[7]	1b	Reserved. Written as 0b.
	[6]	1b	Assertion event for state bit 14
	[5]	1b	Assertion event for state bit 13
	[4]	1b	Assertion event for state bit 12
	[3]	1b	Assertion event for state bit 11
	[2]	1b	Assertion event for state bit 10
	[1]	1b	Assertion event for state bit 9
	[0]	1b	Assertion event for state bit 8
5	For sensors with threshold based events		
	[7]	1b	Deassertion event for upper non-critical going high
	[6]	1b	Deassertion event for upper non-critical going low
	[5]	1b	Deassertion event for lower non-recoverable going high
	[4]	1b	Deassertion event for lower non-recoverable going low
	[3]	1b	Deassertion event for lower critical going high

Table Continued

	[2]	1b	Deassertion event for lower critical going low
	[1]	1b	Deassertion event for lower non-critical going high
	[0]	1b	Deassertion event for lower non-critical going low
	For sensors with discrete events (00h otherwise)		
	[7]	1b	Deassertion event for state bit 7
	[6]	1b	Deassertion event for state bit 6
	[5]	1b	Deassertion event for state bit 5
	[4]	1b	Deassertion event for state bit 4
	[3]	1b	Deassertion event for state bit 3
	[2]	1b	Deassertion event for state bit 2
	[1]	1b	Deassertion event for state bit 1
	[0]	1b	Deassertion event for state bit 0
6	For sensors with threshold based events		
	[7:4]	Reserved. Written as 0000b.	
	[3]	1b	Deassertion event for upper non-recoverable going high
	[2]	1b	Deassertion event for upper non-recoverable going low
	[1]	1b	Deassertion event for upper critical going high
	[0]	1b	Deassertion event for upper critical going low
	For sensors with discrete events (00h otherwise)		
	[7]	1b	Reserved. Written as 0b.
	[6]	1b	Deassertion event for state bit 14
	[5]	1b	Deassertion event for state bit 13
	[4]	1b	Deassertion event for state bit 12

Table Continued

	[3]	1b	Deassertion event for state bit 11
	[2]	1b	Deassertion event for state bit 10
	[1]	1b	Deassertion event for state bit 9
	[0]	1b	Deassertion event for state bit 8

Response data byte number	Data field
1	Completion code

Get sensor event enable command

This command returns the enabled/disabled state for Event Message Generation from the selected sensor. The command also returns the enabled/disabled state for scanning on the sensor.

A typical sensor will come up with Event Messages (EvM) enabled for all thresholds. Sensors are not required to have individual or per-event Event Message enables. The type of enable/disable support that a sensor provides can be obtained from the Sensor Data Record for the sensor.

Table 121: Get sensor event enable command request and response data

Request data byte number	Data field		
1	Sensor number (FFh = reserved)		
Response data byte number	**Data field**		
1	Completion code		
2	[7]	0b	Disable all Event Messages from this sensor [does not impact individual enable/disable status]
	[6]	0b	Disable scanning on this sensor
	[5:0]	Reserved. Ignore on read.	
3	For sensors with threshold based events:		
	[7]	1b	Assertion event for upper non-critical going high
	[6]	1b	Assertion event for upper non-critical going low
	[5]	1b	Assertion event for lower non-recoverable going high

Table Continued

	[4]	1b	Assertion event for lower non-recoverable going low
	[3]	1b	Assertion event for lower critical going high
	[2]	1b	Assertion event for lower critical going low
	[1]	1b	Assertion event for lower non-critical going high
	[0]	1b	Assertion event for lower non-critical going low
	For sensors with discrete events		
	[7]	1b	Assertion event for state 7
	[6]	1b	Assertion event for state 6
	[5]	1b	Assertion event for state 5
	[4]	1b	Assertion event for state 4
	[3]	1b	Assertion event for state 3
	[2]	1b	Assertion event for state 2
	[1]	1b	Assertion event for state 1
	[0]	1b	Assertion event for state 0
4	For sensors with threshold based events		
	[7:4]	Reserved. Written as 0000b.	
	[3]	1b	Assertion event for upper non-recoverable going high
	[2]	1b	Assertion event for upper non-recoverable going low
	[1]	1b	Assertion event for upper critical going high
	[0]	1b	Assertion event for upper critical going low
	For sensors with discrete events (00h otherwise)		
	[7]	1b	Reserved.
	[6]	1b	Assertion event for state bit 14

Table Continued

	[5]	1b	Assertion event for state bit 13
	[4]	1b	Assertion event for state bit 12
	[3]	1b	Assertion event for state bit 11
	[2]	1b	Assertion event for state bit 10
	[1]	1b	Assertion event for state bit 9
	[0]	1b	Assertion event for state bit 8
5	For sensors with threshold based events		
	[7]	1b	Deassertion event for upper non-critical going high
	[6]	1b	Deassertion event for upper non-critical going low
	[5]	1b	Deassertion event for lower non-recoverable going high
	[4]	1b	Deassertion event for lower non-recoverable going low
	[3]	1b	Deassertion event for lower critical going high
	[2]	1b	Deassertion event for lower critical going low
	[1]	1b	Deassertion event for lower non-critical going high
	[0]	1b	Deassertion event for lower non-critical going low
	For sensors with discrete events (00h otherwise)		
	[7]	1b	Deassertion event for state bit 7
	[6]	1b	Deassertion event for state bit 6
	[5]	1b	Deassertion event for state bit 5
	[4]	1b	Deassertion event for state bit 4
	[3]	1b	Deassertion event for state bit 3
	[2]	1b	Deassertion event for state bit 2
	[1]	1b	Deassertion event for state bit 1

Table Continued

	[0]	1b	Deassertion event for state bit 0
6	For sensors with threshold based events		
	[7:4]	Reserved. Written as 0000b.	
	[3]	1b	Deassertion event for upper non-recoverable going high
	[2]	1b	Deassertion event for upper non-recoverable going low
	[1]	1b	Deassertion event for upper critical going high
	[0]	1b	Deassertion event for upper critical going low
	For sensors with discrete events (00h otherwise)		
	[7]	1b	Reserved. Written as 0b.
	[6]	1b	Deassertion event for state bit 14
	[5]	1b	Deassertion event for state bit 13
	[4]	1b	Deassertion event for state bit 12
	[3]	1b	Deassertion event for state bit 11
	[2]	1b	Deassertion event for state bit 10
	[1]	1b	Deassertion event for state bit 9
	[0]	1b	Deassertion event for state bit 8

Get sensor reading command

This command returns the present reading for sensor. The sensor device may return a stored version of a periodically updated reading, or the sensor device may scan to obtain the reading after receiving the request.

The event/reading type code from the SDR determines the state bits returned by discrete sensors.

Table 122: Get sensor reading command request data

Request data byte number	Data field
1	Sensor number (FFh = reserved)

Response data byte number	Data field
1	Completion code.
2	Sensor reading Byte 1: byte of reading. Ignore on read if the sensor does not return a numeric (analog) value.
3	[7] — 0b = All event messages disabled from this sensor. [6] — 0b = sensor scanning disabled. [5] — 1b = reading/state unavailable (formerly: initial update in progress). This bit is set to indicate a re-arm or `set event receiver` command has been used to request an update of the sensor status, and that update has not occurred yet. Software should use this bit to avoid getting an incorrect status while the first sensor update is in progress. This bit is only required if it is possible for the controller to receive and process a `get sensor reading` or `get sensor event status` command for the sensor before the update completes. This is most likely the case for sensors, such as fan RPM sensors that may require seconds to accumulate the first reading after a re-arm. The bit may also indicate when a reading/state is unavailable because the management controller cannot obtain a valid reading or state for the monitored entity, typically because the entity is not present. [4:0] — reserved. Ignore on read.

Table Continued

(4)	For threshold-based sensors: Present threshold comparison status [7:6] — reserved. Returned as 1b. Ignore on read. [5] — 1b = at or above (≥) upper non-recoverable threshold [4] — 1b = at or above (≥) upper critical threshold [3] — 1b = at or above (≥) upper non-critical threshold [2] — 1b = at or below (≤) lower non-recoverable threshold [1] — 1b = at or below (≤) lower critical threshold [0] — 1b = at or below (≤) lower non-critical threshold For discrete reading sensors: [7] = 1b = state 7 asserted [6] = 1b = state 6 asserted [5] = 1b = state 5 asserted [4] = 1b = state 4 asserted [3] = 1b = state 3 asserted [2] = 1b = state 2 asserted [1] = 1b = state 1 asserted [0] = 1b = state 0 asserted
(5)	For discrete reading sensors only. (Optional) (00h Otherwise) [7] = Reserved. Returned as 1b. Ignore on read. [6] = 1b = state 14 asserted [5] = 1b = state 13 asserted [4] = 1b = state 12 asserted [3] = 1b = state 11 asserted [2] = 1b = state 10 asserted [1] = 1b = state 9 asserted [0] = 1b = state 8 asserted

DCMI specific commands

This section includes commands specific to the Data Center Manageability Interface implementation in iLO. The following *DCMI completion codes* table shows the completion codes and definitions found in data byte 1 of DCMI specific command responses.

Table 123: DCMI completion codes

Code	Definition
00h	Command completed normally.
C0h	Node busy. Command could not be processed because command processing resources are temporarily unavailable.
C1h	Invalid Command. Used to indicate an unrecognized or unsupported command.
C2h	Command invalid for given LUN.
C3h	Timeout while processing command. Response unavailable.
C4h	Out of space. Command could not be completed because of a lack of storage space required to execute the given command operation.
C5h	Reservation cancelled or invalid reservation ID.
C6h	Request data truncated.
C7h	Request data length invalid.
C8h	Request data field length limit exceeded.
C9h	Parameter out of range. One or more parameters in the data field of the request are out of range. This is different from 'Invalid data field' (CCh) code in that it indicates that the erroneous fields have a contiguous range of possible values.
CAh	Cannot return number of requested data bytes.
CBh	Requested sensor, data, or record not present.
CCh	Invalid data field in request.
CDh	Command illegal for specified sensor or record type.
CEh	Command response could not be provided.
CFh	Cannot execute duplicated request. This completion code is for devices which cannot return the response that was returned for the original instance of the request. Such devices should provide separate commands that allow the completion status of the original request to be determined. An event receiver does not use this completion code, but returns the 00h completion code in the response to (valid) duplicated requests.
D0h	Command response could not be provided. SDR repository in update mode.

Table Continued

Code	Definition
D1h	Command response could not be provided. Device in firmware update mode.
D2h	Command response could not be provided. MC initialization or initialization agent in progress.
D3h	Destination unavailable. Cannot deliver request to selected destination. For example, this code can be returned if a request message is targeted to SMS, but receive message queue reception is disabled for the particular channel.
D4h	Cannot execute command due to insufficient privilege level or other security-based restriction.
D5h	Cannot execute command. Command, or request parameters, not supported in present state.
D6h	Cannot execute command. Parameter is illegal because command sub-function has been disabled or is unavailable.
FFh	Unspecified error.

Get DCMI capability info command

This command provides version information for DCMI and information about the mandatory and optional DCMI capabilities that are available on the particular platform. The command is session-less, and is a bare-metal provisioning command. The availability of features does not imply that the features are configured.

Table 124: Get DCMI capability info command request and response data

Request data byte number	Data field
1	Group extension identification = DCh
2	Parameter selector
Response data byte number	**Data field**
1	Completion code. See **DCMI specific commands**.
2	Group extension identification = DCh
3:4	DCMI specification conformance: • Byte 1—Major version (01h) • Byte 2—Minor version (01h)
5	Parameter revision = 02h
6:N	Parameter data

Table 125: DCMI Capabilities Parameters

Parameter	#	Parameter data (nonvolatile unless noted)
Supported DCMI capabilities	1	This field returns the supported capabilities available in the server in conformance to the DCMI specification for both Platform and Manageability access. All reserved bits shall be set to 0b. • Byte 1—Reserved • Byte 2—Platform capabilities ◦ All bits: 0b=Not present 1b=Available ◦ [7:1]—Reserved ◦ [0]—Power management/monitoring support (Defined as support for either power monitoring or power monitoring plus power limiting.) • Byte 3—Manageability access capabilities ◦ All bits: 0b=Not present 1b=Available ◦ [7:3]—Reserved ◦ [2]—Out-of-band secondary (second) LAN channel available (optional) ◦ [1]—Serial TMODE available (TMODE on serial port to management controller) (optional) ◦ [0]—Power management/monitoring support (Defined as support for either power monitoring or power monitoring plus power limiting.)
Mandatory platform attributes	2	This field returns the platform attributes for the platform capabilities. All reserved bits shall be set to 0b. • Byte 1:2—SEL attributes ◦ [15]–SEL automatic rollover enabled (SEL overwrite) 0b=Not present, 1b=available ◦ [14] Entire SEL flush upon rollover (valid if rollover is enabled) 0b=Not present, 1b=available ◦ [13] Record level SEL flush upon rollover (valid if rollover is enabled) 0b=Not present, 1b=available

Table Continued

Parameter	#	Parameter data (nonvolatile unless noted)
		◦ [12] Reserved ◦ [11:0] Number of SEL entries (Maximum of 4096. The number of entries supported must be 64 or greater to be in conformance) • Byte 3-4—Reserved • Byte 5–Sample frequency for temperature monitoring in units of 1 second.
Optional platform attributes	3	This field returns the attributes required for the recommended platform capabilities. • Byte 1—Power management device slave address ◦ [7:1] 7-bit I^2C slave address on IPMB. ◦ [0] Reserved. Write as 0b. ◦ [20h=MC, XXh=Satellite/external controller] • Byte 2—Power management controller channel number ◦ [7:4] The management controller channel number. Use 0h for the primary MC. ◦ [3:0] Device revision (used for providing the revision control for power management capability)

Table Continued

Parameter	#	Parameter data (nonvolatile unless noted)
Manageability access attributes	4	This field returns the attributes of the manageability access. • Byte 1–Mandatory primary LAN OOB support (RMCP+ support only) [7-0] Channel number (0xFFh==Not supported) • Byte 2–Optional secondary LAN OOB support (RMCP+ support only) [7:0] Channel number (0xFFh==Not supported) • Byte 3–Optional serial out-of-band TMODE capability [7:0] Channel number (0xFFh==Not supported)
Enhanced System Power Statistics attributes (Optional)	5	This field returns list of Enhanced System Power Statistic capabilities. This parameter has a direct relationship with the Get Power Reading command. For more information, see **Table 126: Get power reading command request and response data**. • Byte 1—The number of supported rolling average time periods. The maximum number of supported rolling average time periods reported by the platform management subsystem is limited by the DCMI transport response length. • Bytes 2:*n*—Rolling average time periods (where *n* is (value of byte 1 + 1). [7:6]: Time duration units ◦ 00b: Seconds ◦ 01b: Minutes ◦ 10b: Hours ◦ 11b: Days [5:0]: Time duration Zero time duration is acceptable and means “NOW” or current reading. iLO supports 10-second, 5-minute, and 24-hour power statistics. iLO 5 2.33 and later also supports 7-day power statistics.

Get power reading command

The `get power reading` command returns system power statistics.

Table 126: Get power reading command request and response data

Request data byte number	Data field
1	Group Extension Identification = DCh
2	Mode • 01h – System Power Statistics • 02h – Enhanced System Power Statistics
3	Mode based attributes • For Mode 01h System Power Statistics Attributes: Reserved for future use 00h • For Mode 02h Enhanced System Power Statistics Attributes: Rolling Average Time periods, only the time periods specified in Parameter 5 of Get DCMI Capabilities Info Command are supported.
4	Reserved

Response data byte number	Data field
1	Completion Code. Refer to **DCMI specific commands**.
2	Group Extension Identification = DCh
3:4	Current Power in watts.
5:6	Minimum Power over sampling duration in watts Note: Sampling duration depends on Mode selection.
7:8	Maximum Power over sampling duration in watts Note: Sampling duration depends on Mode selection.
9:10	Average Power over sampling duration in watts Note: Sampling duration depends on Mode selection.
11:14	IPMI Specification based Time Stamp. For Mode 02h: The time stamp specifies the end of the averaging window

Table Continued

Response data byte number	Data field
15:18	Statistics reporting time period • For Mode 01h: Timeframe in milliseconds, over which the controller collects statistics • For Mode 02h: Timeframe reflects the Averaging Time period in units.
19	Power Reading State • [0:5] Reserved • [6] 1b – Power Measurement active • 0b – No Power Measurement is available. • [7] Reserved

Get power limit command

The Get power limit command returns the present settings for the power limit that has been set using the Set power limit command.

Table 127: Get power limit command request and response data

Request data byte number	Data field
1	Group Extension Identification = DCh
2:3	Reserved for future use, use 0000h

Response Data byte number	Data field
1	Completion Code. Refer to **DCMI specific commands**. 00h = Power Limit Active 80h = No Active Set Power Limit OEM can also provide Completion Codes.
2	Group Extension Identification = DCh
3:4	Reserved for future use

Table Continued

Response Data byte number	Data field
5	Exception Actions, taken if the Power Limit is exceeded and cannot be controlled within the Correction Time Limit • 01h Hard Power Off system and log event to SEL 02h – 10h OEM defined actions • 02h — 10h OEM defined actions • 11h Log event to SEL only • 12h-FFh Reserved
6:7	Power Limit Requested in Watts
8:11	Correction Time Limit in milliseconds See description of corresponding parameter in **Set power limit command**.
12:13	Reserved for future use
14:15	Management application Statistics Sampling period in seconds

Set power limit command

The Set Power Limit command sets the power limit parameters on the system. The power limit defines a threshold which, if exceeded for a configurable amount of time, will trigger a system power off and/or event logging action. This enables the Power Limit to be used as a form of “circuit breaker” for protecting data center power delivery from systems that have abnormal, prolonged power excursions outside their normal operating range.

It is recommended to do a Get Power Limit or check the Get Power Reading command before attempting to set and activate or re-activate the power limit. If the limit is already active, the Set Power Limit command may immediately change the limit that is in effect. However, software should always explicitly activate the limit using the Activate/Deactivate power limit command to ensure the setting takes effect.

It should be noted that in the current context, this command shall be used to set a static upper limit of system power usage and not used as a command interface for dynamic or frequently changing power limit. The power limit set should be persistent across AC and DC cycles.

Table 128: Set power limit command request and response data

Request data byte number	Data field
1	Group Extension Identification = DCh
2:4	Reserved for future use

Table Continued

Request data byte number	Data field
5	Exception Actions, taken if the Power Limit is exceeded and cannot be controlled within the Correction time limit 00h – No Action 01h – Hard Power Off system and log events to SEL. 02h – 10h OEM defined actions 11h – Log event to SEL only 12h-FFh Reserved
6:7	Power Limit Requested in Watts
8:11	Correction Time Limit in milliseconds Maximum time taken to limit the power after the platform power has reached the power limit before the Exception Action will be taken. The Exception Action shall be taken if the system power usage constantly exceeds the specified power limit for more than the Correction Time Limit interval. The Correction Time Limit timeout automatically restarts if the system power meets or drops below the Power Limit.
12:13	Reserved for future use
14:15	Management application Statistics Sampling period in seconds

Response Data byte number	Data field
1	Completion Code. Refer to **DCMI specific commands**. =00h – Success =84h – Power Limit out of range =85h – Correction Time out of range =89h – Statistics Reporting Period out of range OEM can also provide Completion Codes.
2	Group Extension Identification = DCh

Activate/Deactivate power limit command

The command is used to activate or deactivate the power limit set. This command should succeed a successful Set Power limit command.

Table 129: Activate/Deactivate power limit command request and response data

Request data byte number	Data field
1	Group Extension Identification = DCh
2	Power Limit Activation 00h – Deactivate Power Limit 01h – Activate Power Limit
3:4	Reserved
Response Data byte number	**Data field**
1	Completion Code. Refer to **DCMI specific commands**.
2	Group Extension Identification = DCh

Get asset tag command

This command enables management consoles or local software to get asset tag data. UTF-8 encoding is identified when the first three bytes (offsets 0, 1, and 2) of the returned asset tag data are set to the UTF-8 byte order mark (BOM) pattern, EFh, BBh, BFh, respectively.

Table 130: Get asset tag command request and response data

Request data byte number	Data field
1	Group extension identification = DCh
2	Offset to read
3	Number of bytes to read (16 bytes maximum) **NOTE:** If the number of bytes to read starting from the given **offset to read** exceeds the number of remaining asset tag data bytes, the command will complete normally (completion code = 00h) but will only return the remaining bytes (provided the offset to read and bytes to read are within their correct ranges.) For example, if the asset tag length is presently 20 bytes, submitting an offset to read of 16 and a bytes to read of 16 will be accepted, but only the asset tag data bytes at offsets 16–19 will be returned.
Response data byte number	**Data field**

Table Continued

1	Completion code. C9h shall be returned if offset >62, offset to read+bytes to read >63, or bytes to read is >16. The following applies to implementations that keep the DCMI asset tag and IPMI FRU asset tag information synchronized: If the encoding indicated by the Type/Length byte in the IPMI FRU is not set to ASCII+Latin1 or the language code for the IPMI FRU product info area is not set to English (0 or 25), the command shall return the requested data bytes, but shall also return a command-specific completion code based on the detected encoding type, as follows: • 80h=Encoding type in FRU is binary/unspecified • 81h=Encoding type in FRU is BCD Plus • 82h=Encoding type is FRU is 6–bit ASCII Packed • 83h=Encoding type is set to ASCII+Latin1 but language code is not set to English (indicating data is 2–byte Unicode). The management controller does not check, nor require, a BOM in the asset tag data; asset tag data can be stored and retrieved as ASCII+Latin1 without receiving an error completion code.
2	Group extension identification = DCh
3	Total asset tag length (must be less than or equal to 64 bytes)
4:N	Asset tag data (starting from the offset to read)

Get DCMI sensor info command

This DCMI command returns information about a DCMI-specified sensor. A particular sensor is identified by the combination of its entity ID and entity instance numbers.

Table 131: DCMI Entity ID extension

Entity ID description	Entity ID	Entity instance	Sensor type
Inlet temperature	40h	0x01…n	Temp (01h)
CPU temperature (based on number of processors or cores)	41h	0x01…n	Temp (01h)
Baseboard temperature	42h	0x01…n	Temp (01h)

Table 132: Get DCMI sensor info command request and response data

Request data byte number	Data field
1	Group extension identification = DCh
2	Sensor type
3	Entity ID
4	Entity instance • 00h = Retrieve information about all instances associate with entity ID • 01h-FFh = Retrieve only the information about particular instance
5	Entity instance start, used with entity instance 00h for number of instance exceeding one IPMI response.
Response data byte number	**Data field**
1	Completion code. See **DCMI specific commands**.
2	Group extension identification = DCh
3	Total number of available instances for the entity ID.
4	Number of record IDs in this response (maximum of 8 per response) 01h for entity instance not equal to 00h.
5:6 + N	SDR record ID corresponding to the entity IDs. • Byte 1: Record ID LS byte, used for retrieving SDR records. • Byte 2: Record ID MS byte, used for retrieving SDR records. **NOTE:** The management controller can include SDR record IDs corresponding to entities IDs compatible IPMI 2.0 specified entity IDs such as: • Request for inlet temp (40h) can also include air inlet temp info (37h) • Request for CPU temp (41h) can also include CPU temp (03h) • Request for baseboard temp (42h) can also include system board (07h)

Set asset tag command

This command enables remote consoles or local software to set the asset tag data. UTF-8 encoding is identified when the first three bytes (offsets 0, 1, and 2) of the returned asset tag data are set to the UTF-8 byte order mark (BOM) pattern, EFh, BBh, BFh, respectively. Otherwise the data encoding is assumed to be the ASCII+Latin1 subset. Note that the management controller simply stores all eight bits of each of the given asset tag data bytes. It does not check the encoding of the asset tag data bytes, nor does it check for a BOM in the data.

Implementations that keep the asset tag in synch with the IPMI FRU data shall write the given characters to the asset tag field in the product info area of the IPMI FRU device and set the encoding of the corresponding type/length byte field to ASCII+Latin1.

Table 133: Set asset tag command request and response data

Request data byte number	Data field
1	Group extension identification = DCh
2	Offset to write (0 to 62). The offset is relative to the first character of the asset tag data.
3	Number of bytes to write (16 bytes maximum) **NOTE:** The command must set the overall length of the asset tag (in bytes) to the value (offset to write + bytes to write). Any pre-existing asset tag bytes at offsets past that length are automatically deleted.
4–N	Asset tag data
Response data byte number	**Data field**
1	Completion code. C9h shall be returned if offset >62, offset+bytes to write >63, or bytes to writer >16. A C9h completion code shall also be returned if an attempt is mde to writer to an offset that is more than one great than the length of the presently stored asset tag data. Set operations for asset tags must be contiguous. For example, if the asset tag is presently seven bytes long an attempt to writer starting at offset 10 will be rejected and a C9h completion code returned.
2	Group extension identification = DCh
3	Total asset tag length. This is the length in bytes of the stored asset tag after set operation has completed. The asset tag length shall be set to the sum of the offset to write plus bytes to write. For example, if offset to write is 32 and bytes to write is 4, the total asset tag length returned will be 36.

Management controller ID string

The management controller ID string is provided in order to accommodate the requirement for the management controllers to identify themselves during discovery phases. `Set/get management controller identifier string` commands are provided to provision the controller with the unique identification. The management controller must maintain the management controller identifier string as non-volatile data.

The management controller ID string is used to override the default OEM provided ID for DHCP discovery. If the management controller ID string is not provisioned, then the default controller ID shall be “DCMI<*mac-address*>”. The maximum length of the identifier string shall be 64 bytes including a NULL terminator.

Get controller ID string command

Table 134: Get controller ID string command request and response data

Request data byte number	Data field
1	Group extension identification = DCh
2	Offset to read
3	Number of bytes to read (16 bytes maximum)
Response data byte number	**Data field**
1	Completion code. See **DCMI specific commands**.
2	Group extension identification = DCh
3	Total length. **NOTE:** The maximum length of the identifier string must not exceed 64 bytes.
4–N	Data

Set controller ID string command

Table 135: Set controller ID string command request and response data

Request data byte number	Data field
1	Group extension identification = DCh
2	Offset to write
3	Number of bytes to write (16 bytes maximum)
4–N	Data
Response data byte number	**Data field**
1	Completion code. See **DCMI specific commands**.

Table Continued

2	Group extension identification = DCh
3	Total length written. **NOTE:** The maximum length of the identifier string must not exceed 64 bytes.

Get Temperature Readings command

This DCMI command provides a way to get the DCMI temperature sensor readings in degrees Celsius. With this command, the sensor reading conversion factors from the IPMI sensor data records are not required.

In some cases, a temperature reading source might return a value that is relative to another temperature threshold point. For example, this situation might occur with processor temperature readings. Since most users prefer readings that appear as absolute temperature values, this implementation synthesizes what appears to be an absolute temperature when an absolute temperature value is unavailable.

Table 136: Get Temperature Readings command request and response data

Request data byte number	Data field
1	Group extension identification = DCh
2	Sensor type—For more information, see **Table 131: DCMI Entity ID extension**.
3	Entity ID—For more information, see **Table 131: DCMI Entity ID extension**. The command automatically maps the DCMI 1.1 entity IDs to the corresponding IPMI entity IDs. However, for compatibility with future versions of the DCMI specification, Hewlett Packard Enterprise recommends using the IPMI entity ID values.
4	Entity instance • 00h = Retrieve information about all instances associated with the entity ID. • 01h-FFh = Retrieve information about a specific instance. **NOTE:** Hewlett Packard Enterprise recommends using DCMI sensor entity instance numbers that are sequential and contiguous starting from 1.
5	Entity Instance Start—Used with entity instance 00h. Use 01h to return temperature data starting with the first set of temperature data.

Response data byte number	Data field
1	Completion code—For more information, see **DCMI specific commands**.
2	Group extension identification = DCh
3	Total number of available instances for the entity ID.

Table Continued

Response data byte number	Data field
4	Number of sets of temperature data in this response (maximum 8 per response). 01h for an entity instance not equal to 00h.
5:6 + N	Temperature Data Byte1: Bit 7—Sign bit (1 is negative and 0 is positive). Bit 6:0—Temperature data in degrees Celsius (the range is -128°C to 128°C) Byte 2— Entity instance number for the sensor.

OEM commands

Get IPMI Configuration Parameters command

Table 137: Get IPMI Configuration Parameters command request and response data

netfn = 0x30, cmd = 0xbc, user privilege

Request data byte number	Data field
1	• [7] ◦ 0b = get parameter ◦ 1b = get parameter revision only • [6:0]—reserved
2	Parameter selector
3	Set Selector. Selects a particular set or block data under the given parameter selector. 00h if parameter does not use a set selector.
4	Block Selector (00h if parameter does not require a block number)

Response data byte number	Data field
1	Completion Code. Generic plus the following command-specific completion codes: 80h = parameter not supported.
2	[7:0] - Parameter revision. Format: MSN = present revision. LSN = oldest revision parameter is backward compatible with. 11h for parameters in this specification.
The following data bytes are not returned when the 'get parameter revision only' bit is 1b.	

Table Continued

Response data byte number	Data field
3:N	Information parameter data If the rollback feature is implemented, the BMC makes a copy of the existing parameters when the 'set in progress' state becomes asserted. For more information, see the Set In Progress parameter #0. When the 'set in progress' state is active, the BMC returns data from this copy of the parameters, plus any uncommitted changes that were made to the data. Otherwise, the BMC returns parameter data from nonvolatile storage.
0	data 1—This parameter is used to indicate when any of the following parameters are being updated, and when the updates are completed. The bit is primarily provided to alert software that some other software or utility is in the process of changing the data. An implementation can also elect to provide a rollback feature that uses this information to decide whether to roll back to the previous configuration information, or to accept the configuration change. If used, the rollback will restore all parameters to their previous state. Otherwise, the change will take effect when the write occurs. [7:2]—reserved [1:0]—00b = set complete. If a system reset or transition to powered down state occurs while 'set in progress' is active, the BMC will go to the 'set complete' state. If rollback is implemented, going directly to 'set complete' without first doing a 'commit write' will cause any pending write data to be discarded. 01b = set in progress. This flag indicates that some utility or other software is writing to parameter data. It is a notification flag only; it is not a resource lock. The BMC does not provide any interlock mechanism that would prevent other software from writing parameter data. 10b = commit write (optional). This value is used only if a rollback is implemented. The BMC will save the data that was written since the last time the 'set in progress' state was active, and then go to the 'set in progress' state. An error completion code is returned if this option is not supported. 11b = reserved

Table 138: IPMI Configuration Parameters

Parameter	#	Parameter Data
Test Mode	1	This field is used to configure the BMC IPMI stack for running a particular test suite. This setting takes effect immediately. [7:1]—reserved [0]—mode 0h = None (default) 1h = DCTS (responders address not scrutinized)
SEL Rollover Mode	2	This field is used to configure the BMC IPMI stack SEL behavior. This setting takes effect immediately. [7:1]—reserved [0]—mode 0h = No Rollover 1h = Rollover, Record level SEL flush (default)
Device ID Configuration	3	This field is used to configure the BMC IPMI stack Device ID for Get Device ID response. [7:1]—reserved [0]—Device ID Mode 0h = Standard (HPE IANA) (default) 1h = Legacy (HP IANA) (C-class blades will have this hard set.)

Table Continued

Parameter	#	Parameter Data
Sensor Configuration	4	This field is used to configure the BMC IPMI stack Sensor generation. A BMC reset is required for changes to take effect. [7:3]—reserved [2]—Compact Discrete Sensor Record Mode 0h = Use Compact Sensor Records for Discrete Sensors (default) 1h = Legacy—Use Full Records for Discrete Sensors (C-class blades will have this hard set) [1]—Analog/Discrete Combined Sensor Mode 0h = Standard (default) 1h = Legacy [0]—Sensor Naming Convention 0h = Legacy (default) 1h = Programmatic
Fan Sensor Configuration	5	This field is used to configure the BMC IPMI stack for Fan Sensor generation. A BMC reset is required for changes to take effect. [7:1]—reserved [0]—Fan Sensor Type 0h = Severity (07h) (default) 1h = Availability (0Ah)
System Health Sensor Configuration	6	This field is used to configure the BMC IPMI stack for System Health Sensor generation. A BMC reset is required for changes to take effect. [7:1]—reserved [0]—Generate Legacy OEM type 0h = Disabled (default) 1h = Enabled (C-class blades will have this hard set)

Table Continued

Parameter	#	Parameter Data
DIMM FRU Configuration	7	This field is used to configure the BMC IPMI stack FRU generation. A BMC reset is required for changes to take effect. [7:1]—reserved [0]—DIMM FRU generation on LUN 1 0h = Disabled (default) (C-class blades will have this hard set) 1h = Enabled
DIMM Sensor Configuration	8	This field is used to configure the BMC IPMI stack DIMM Sensor generation. A BMC reset is required for changes to take effect. [7:1]—reserved [0]—Individual DIMM sensor generation on LUN 1 0h = Disabled (default) (C-class blades will have this hard set) 1h = Enabled

Set IPMI Configuration Parameters command

Table 139: Set IPMI Configuration Parameters command request and response data

netfn = 0x30, cmd = 0xbb, admin privilege

Request data byte number	Data field
1	Parameter selector
2:n	Configuration Parameter Data

Response data byte number	Data field
1	Completion Code. Generic plus the following command-specific completion codes: 80h = parameter not supported. 81h = attempt to set the 'set in progress' value (in parameter #0) when not in the 'set complete' state. (This completion code provides a way to recognize that another party has already 'claimed' the parameters) 82h = attempt to write read-only parameter.

Table 140: IPMI Configuration Parameters

Parameter	#	Parameter data
Test Mode	1	This field is used to configure the BMC IPMI stack for running a particular test suite. This setting takes effect immediately. [7:1]—reserved [0]—mode 0h = None (default) 1h = DCTS (responders address not scrutinized)
SEL Rollover Mode	2	This field is used to configure the BMC IPMI stack SEL behavior. This setting takes effect immediately. [7:1]—reserved [0]—mode 0h = No Rollover 1h = Rollover, Record level SEL flush (default)
Device ID Configuration	3	This field is used to configure the BMC IPMI stack Device ID for Get Device ID response. [7:1]—reserved [0]—Device ID Mode 0h = Standard (HPE IANA) (default) 1h = Legacy (HP IANA) (C-class blades will have this hard set)

Table Continued

Parameter	#	Parameter data
Sensor Configuration	4	This field is used to configure the BMC IPMI stack Sensor generation. A BMC reset is required for changes to take effect. [7:3]—reserved [2]—Compact Discrete Sensor Record Mode 0h = Use Compact Sensor Records for Discrete Sensors (default) 1h = Legacy—Use Full Records for Discrete Sensors (C-class blades will have this hard set) [1]—Analog/Discrete Combined Sensor Mode 0h = Standard (default) 1h = Legacy [0]—Sensor Naming Convention 0h = Legacy (default) 1h = Programmatic
Fan Sensor Configuration	5	This field is used to configure the BMC IPMI stack for Fan Sensor generation. A BMC reset is required for changes to take effect. [7:1]—reserved [0]—Fan Sensor Type 0h = Severity (07h) (default) 1h = Availability (0Ah)
System Health Sensor Configuration	6	This field is used to configure the BMC IPMI stack for System Health Sensor generation. A BMC reset is required for changes to take effect. [7:1]—reserved [0]—Generate Legacy OEM type 0h = Disabled (default) 1h = Enabled (C-class blades will have this hard set)

Table Continued

Parameter	#	Parameter data
DIMM FRU Configuration	7	This field is used to configure the BMC IPMI stack FRU generation. A BMC reset is required for changes to take effect. [7:1]—reserved [0]—DIMM FRU generation on LUN 1 0h = Disabled (default) (C-class blades will have this hard set) 1h = Enabled
DIMM Sensor Configuration	8	This field is used to configure the BMC IPMI stack DIMM Sensor generation. A BMC reset is required for changes to take effect. [7:1]—reserved [0]—Individual DIMM sensor generation on LUN 1 0h = Disabled (default) (C-class blades will have this hard set.) 1h = Enabled

Restore Defaults command

This command takes either a reset to factory defaults or a bit map of targeted areas to reset. Once defaults have been set, a warm reset is done by iLO. If there is an error condition specific to setting a default, the response byte 2 bitmap indicates where the error occurred.

Table 141: Restore Defaults command request and response data

netfn = 0x30, cmd = 0xbf, admin privilege

Request data byte number	Data field
1	Default Target [7]—1b = Factory Default, [6:0] - - reserved 0b = Targeted Default, [6:0] – bitmap of targets [6:1] – reserved [0] – User Database

Response data byte number	Data field
1	Completion Code: 0x00 = success 0xC7 = request data length invalid 0xCC = Invalid data field 0xCF = cannot execute duplicated request. 0xD4 = insufficient privilege level Command-specific codes: 0x81—Failed to set defaults in specified areas - see byte 2.
2	Target Error: [7]—1b = Factory Default, [6:0] - - reserved 0b = Targeted Default, [6:0] -Default Target [6:1]—reserved [4]—Security State [3]—IEL [2]—IML [1]—Network Configuration [0]—User Database

IPMI Messaging and Interfaces

IPMI uses message based interfaces for the different interfaces to the platform management subsystem such as IPMB, LAN, and the system interface to the MC.

All IPMI messages share the same fields in the message payload regardless of the interface (transport) that they are transferred over. The same core of IPMI messages is available over every IPMI specified interface, just wrapped differently according to the needs of the particular transport. This enables management software that works on one interface to be converted to use a different interface by changing the underlying driver for that particular transport. This also enables knowledge reuse, that is a developer who understands the operation of IPMI commands over one interface can readily apply that knowledge to a different IPMI interface.

IPMI messaging uses a request/response protocol. IPMI request messages are commonly referred to as commands. The use of a request/response protocol facilitates the transfer of IPMI messages over different transports. It also facilitates multi-master operations on busses like the IPMB, allowing messages to be interleaved and multiple management controllers to directly intercommunicate on the bus.

IPMI commands are grouped into functional command sets, using a field called the network function code (NetFn). There are command sets for sensor and event related commands, chassis commands, and others. This functional grouping makes it easier to organize and manage the assignment and allocation of command values.

All IPMI request messages have a network function command and optional data fields. All IPMI response messages carry network function command optional data and a completion code field. As inferred earlier, the differences between the different interfaces has to do with the framing and protocols used to transfer this payload. For example, the IPMB protocol adds fields for I^2C and controller addressing and data integrity checking and handling whereas the LAN interface adds formatting for sending IPMI messages as LAN packets.

System Interfaces

IPMI defines three standard system interfaces that system software use for transferring IPMI messages to the MC. In order to support a variety of microcontrollers, IPMI offers a choice of system interface, this is also key to enabling cross-platform software. The system interfaces are similar enough so that a single driver can be created that supports all IPMI system interfaces.

The system interface connects to a system bus that can be driven by the main processor(s). The present IPMI system interfaces can be I/O or memory mapped. Any system bus that allows the main processor(s) to access the specified I/O or memory locations, and meet the timing specifications, can be used. Thus, an IPMI system interface could be hooked to the X-bus, PCI, LPC or a proprietary bus off the baseboard chipset.

IPMI system interfaces:

Keyboard controller style (KCS) — the bit definitions and operation of the registers follow that used in the Intel 8742 Universal Peripheral Interface microcontroller. KCS reflects the fact that the 8742 interface was used as the legacy keyboard controller interface in PC architecture computer systems. This interface is available built-in to several commercially available microcontrollers. Data is transferred across the KCS interface using a per byte handshake.

Message interface description

The heart of this specification is the definition of the messages and data formats used for implementing sensors, event messages, event generators and event receivers, the SDR Repository, and the SEL in the platform management architecture. These messages are designed for delivery using a messaging interface with a particular set of characteristics. This section presents the general specification of that interface, and the messages.

The message interface is defined as a request/response interface. That is, a request message is used to initiate an action or set data, and a response message is returned to the requester. In this document, request messages are often referred to as commands, and response messages as responses.

All messages in this specification share the same common elements as the payload to the command interpreter in the logical device that receives the message. The messaging interfaces differ in the framing, physical addressing, and data integrity mechanisms that are used to deliver the payload.

Table 142: Common message components

Component	Description
Network Function (NetFn)	A field that identifies the functional class of the message. The network function clusters IPMI commands into different sets.
Request/Response identifier	A field that unambiguously differentiates request messages from response messages. In the IPMB protocol, this identifier is merged with the NetFn code where even numbered network function codes identify request messages and odd numbered network function codes identify response messages.
Requester's ID	Information that identifies the source of the request. This information must be sufficient to allow the response to be returned to the correct requestor. For example, the IPMB requesters ID consists of the slave address and LUN of the requester device. For a multiple stream system interface the requesters ID is the stream ID for the stream through which the request was issued.
Responder's ID	A field that identifies the Responder to the request. In request messages this field is used to address the request to the desired responder, in response messages this field is used to assist the requester in matching up a response with a given request.
Command	The messages specified in this document contain a one-bye command field. Commands are unique within a given network function. Command values can range from 00h through FDh. Code FEh is reserved for future extension of the specification and code FFh is reserved for message interface level error reporting on potential future interfaces.
Data	The data field carries the additional parameters for a request or a response, if any.

IPMI Messaging Interfaces

In iLO, there are two system interface implementations specified for the MC: KCS and a proprietary high speed interface, CHIF. The MC can also be reached through additional interfaces such as the IPMB and LAN interfaces.

Network function codes

The network layer in the connection header includes a six-bit field identifying the function accessed. The remaining two bits are the LUN field. The LUN field provides further sub-addressing within the node.

The network function clusters commands into functional command sets. In a parsing hierarchy, the LUN field may be thought of as the selector for a particular network function handler in the node, and the network function may be considered the selector for a particular command set handler within the node.

The *iLO network function codes* table defines the supported network functions. With the exception of the application and firmware transfer network functions, the commands and responses for a given network function are not node specific. The format and function for standard command sets is specified later.

Table 143: iLO network function codes

Value(s)	Name	Meaning	Description
00, 01	Chassis	Chassis device requests and responses	00h identifies the message as a command/request and 01h as a response, relating to the common chassis control and status functions.
04, 05	Sensor/Event	Sensor and event requests and responses	This functionality can be present on any node. 04h identifies the message as a command/request and 05h as a response, relating to the configuration and transmission of event messages and system sensors.
06, 07	Application	Application requests and responses	06h identifies the message as an application command/ request and 07h a response. The exact format of application messages are implementation-specific for a particular device, with the exception of App messages that are defined by the IPMI specifications.
0A, 0B	Storage	Non-volatile storage requests and responses	This functionality can be present on any node that provides non-volatile storage and retrieval services.
0C, 0D	Transport	Media-specific configuration & control	Requests (0Ch) and responses (0Dh) for IPMI-specified messages that are media-specific configuration and operation, such as configuration of serial and LAN interfaces.
2Ch–2Dh	Group extension	Non-IPMI group requests and responses	The first data byte position in requests and responses under this network function identifies the defining body that specifies command functionality. Software assumes that the command and completion code field positions will hold command and completion code values. The following values are used to identify the defining body. • DCh DCMI specifications at **http://www.intel.com/go/dcmi** • All other reserved When this network function is used, the ID for the defining body occupies the first data byte in a request, and the second data byte (following the completion code) in a response.

Completion codes

All response messages specified in this document include a completion code as the first byte in the data field of the response. A management controller that gets a request to an invalid (unimplemented) LUN must return an error completion code using that LUN as the responder's LUN (RsLUN) in the response. The completion code indicates whether the associate request messages completed successfully and normally, and if not, provides a value indicating the completion condition.

Completion codes work at the command level. They are responses to the interpretation of the command after it has been received and validated through the messaging interface. Errors at the network (messaging interface) level are handled with a different error reporting mechanism.

Completion code values are split into generic, device-specific (which covers OEM) and command-specific ranges. All commands can return generic completion codes. Commands that complete successfully return the 00h command completed normally, completion code. Commands that produce error conditions or return a response that varied from what was specified by the request parameters for the command, return a non-zero completion code as specified in the following *Completion Codes* table.

Table 144: Completion Codes

Code	Definition
GENERIC COMPLETION CODES 00h, C0h-FFh	
00h	Command completed normally
C0h	Node Busy. Command could not be processed— command processing resources temporarily unavailable.
C1h	Invalid Command. Indicates an unrecognized or unsupported command.
C2h	Command invalid for given LUN.
C3h	Timeout while processing command. Response unavailable.
C4h	Out of Space. Command could not be completed because of a lack of storage space.
C5h	Reservation canceled or invalid reservation id.
C6h	Request data truncated.
C7h	Invalid request data length.
C8h	Request data field length limit exceeded.
C9h	Parameter out of range. One or more parameters in the data field of the Request are out of range. This is different from Invalid data field (CCh) code in that it indicates that the erroneous field(s) has a contiguous range of possible values.
CAh	Cannot return number of requested data bytes.
CBh	Requested sensor, data, or record not present.
CCh	Invalid data field in request.
CDh	Command illegal for specified sensor or record type.
CEh	Command response could not be provided.

Table Continued

Code	Definition
CFh	Cannot execute duplicated request. This completion code is for devices which cannot return the response that was returned for the original instance of the request. Such devices should provide separate commands that allow the completion status of the original request to be determined.
D0h	Command response could not be provided. SDR repository in update mode.
D1h	Command response could not be provided. Device in firmware update mode.
D2h	Command response could not be provided. MC initialization or initialization agent in progress.
D3h	Destination unavailable. Cannot deliver request to selected destination. Example, this code can be returned if a request message is targeted to SMS, but receive message queue reception is disabled for the particular channel.
D4h	Cannot execute command due to insufficient privilege level or other security based restriction (example, disabled for firmware firewall).
D5h	Cannot execute command. Command or request parameter(s) not supported in present state.
D6h	Cannot execute command. Parameter is illegal because command sub-function has been disabled or is unavailable (example, disabled for firmware firewall).
FFh	Unspecified error.
DEVICE SPECIFIC (OEM) CODESs 01h — 7Eh	
01h — 7Eh	Device specific (OEM) completion codes. This range is used for command specific codes that are also specific for a particular device and version. A priori knowledge of the device command set is required for interpretation of these codes.
COMMAND SPECIFIC CODES 80h — BEh	
80h — BEh	Standard command specific codes. This range is reserved for command specific completion codes for commands specified in this document.
all other	Reserved.

Additional command specific completion codes, if any, are listed in the completion code field description for the command. In some cases, use of certain command specific completion codes is mandatory. This will be listed alongside the description of the completion code in the command table. If no command specific completion codes are listed, the description will solely indicate that the field is the completion code field.

NOTE: The generic completion code values can be used with any command, regardless of whether additional command specific completion codes are defined.

Channel Model, Authentication, Sessions, and Users

IPMI v2.0 incorporates a common communication infrastructure referred to as the Channel Model.

Channels provide the mechanism for directing the routing of IPMI messages between different media connections to the MC. A channel number identifies a particular connection. For example, 0 is the channel number for the primary IPMB. Up to nine total channels can be supported (the System Interface and primary IPMB, plus seven additional channels with a media type assigned by the implementer.) Channels can thus be used to support multiple IPMB, LAN, Serial, etc., connections to the MC.

Channels can be session-based or session-less. A session is used for two purposes:

1. As a framework for user authentication.
2. To support multiple IPMI messaging streams on a single channel.

Session-based channels thus have at least one user login and support user and message authentication. Session-less channels do not have users or authentication. A LAN channel is session-based, while the System Interface and IPMB are examples of session-less channels.

In order to do IPMI messaging using a session, a session must first be activated. Activating a session authenticates a particular user.

A session has a *Session ID* that is used for tracking the state of a session. The *Session ID* mechanism allows multiple sessions to be simultaneously supported on a channel.

The concept of user is essentially a way to identify a collection of privilege and authentication information. User configuration is done on a per channel basis. This means that a given user could have a different password and set of privileges for accessing the MC via a LAN channel than via a serial channel.

Privilege Levels determine which IPMI commands a given user can execute over a given channel.

Privilege Limits set the maximum privilege level at which a user can operate. A user is configured with a given maximum privilege limit for each channel. In addition, there is a *Channel Privilege Limit* that sets the maximum limit for all users on a given channel. The *Channel Privilege Limit* takes precedence over the privilege configured for the user. Thus, a user can operate at a privilege level that is no higher than the lower of the *User Privilege Limit* and the *Channel Privilege* Limit.

Channel numbers

Each interface has a channel number that is used when configuring the channel and for routing messages between channels. Only the channel number assignments for the primary IPMB and the System Interface are fixed, the assignment of other channel numbers can vary on a per-platform basis. Software uses a `Get Channel Info` command to determine what types of channels are available and what channel number assignments are used on a given platform. The following table describes the assignment and use of the channel numbers:

Table 145: iLO channel number assignments

Channel Number	Type/Protocol	Description
0h	Primary IPMB	Assigned for communication with the primary IPMB. IPMB protocols are used for IPMI messages.
2h	LAN	Assigned for communication between the zone MC and the LAN. RMCP+ is used as the protocol.

Table Continued

Channel Number	Type/Protocol	Description
Eh	Present channel	The value Eh is used as a way to identify the current channel that the command is being received from, for example, if software wants to know what channel number it's presently communicating over, it can find out by issuing a `Get Channel Info` command for channel E.
Fh	System interface	Assigned for routing messages to the system interface.

Logical channels

From the IPMI Messaging point-of-view, a party that bridges a message from one channel to another only is mainly concerned that it gets the correct response from the MC. Often, it does not matter to remote console or system software whether the target channel and devices are physically implemented or not. For example, in iLO the IPMB is a logical channel.

Channel Privilege Levels

Channel privilege limits determine the maximum privilege that a user can have on a given channel. One channel can be configured to allow users to have up to Administrator level privilege, while another channel may be restricted to allow no higher than User level. The privilege level limits take precedence over the privilege level capabilities assigned per user.

Channels can be configured to operate with a particular maximum Privilege Level. Privilege levels tell the MC which commands are allowed to be executed via the channel. The `Set Channel Access` command sets the maximum privilege level limit for a channel. The `Set Session Privilege Level` command requests the ability to perform operations at a particular privilege level. The `Set Session Privilege Level` command can only be used to set privilege levels that are less than or equal to the privilege level limit for the entire channel, regardless of the privilege level of the user.

Table 146: Channel privilege levels

Channel privilege level	Description
Callback	Lowest privilege level. Only commands necessary to support initiating a `Callback` are allowed.
User	Primarily commands that read data structures and retrieve status. Commands that can be used to alter MC configuration, write data to the management controllers, or perform system actions such as resets, power on/off, and watchdog activation are disallowed.
Operator	All MC commands are allowed, except for configuration commands that can change the behavior of the out-of-band interfaces. For example, operator privilege does not allow the capability to disable individual channels or higher.
Administrator	All MC commands are allowed, including configuration commands. An administrator can even execute configuration commands that would disable the channel that the administrator is communicating over.

Users & Password support

User in this specification refers to a collection of data that identifies a password for establishing an authenticated session and the privileges associated with that password. For configuration purposes, sets of user information are organized and accessed according to a numeric *User ID*. When activating a session, user information is looked up with a text *username*.

NOTE: In iLO, anonymous users are not supported due to security concerns.

User access can be enabled on a per channel basis. Thus, different channels can have different sets of users enabled.

If desired, a username on one channel can be associated with a different password than the same username on a different channel. When a session is activated the MC scans usernames sequentially starting with User ID 1 and looks for the first user with matching username and access granted for the given channel. Thus, having different passwords for a given username requires configuring multiple user entities — one for each different password being used for a particular set of channels.

The specification allows a number of different implementations for supporting users on a channel. Minimum requirements include:

- No anonymous user access.
- User names may be fixed or configured, or a combination of both, at the choice of the implementation.
- Support for configuring user passwords for all User IDs is required.
- Support for setting per user privilege limits is optional. If the `Set User Access` command is not supported, the privilege limits for the channel are used for all users.

IPMI sessions

Authenticated IPMI communication to the MC is accomplished by establishing a session. Once established, a session is identified by a Session ID. The Session ID may be thought of as a handle that identifies a connection between a given remote user and the MC, using either the LAN or Serial/Modem connection.

The specification supports having multiple active sessions established with the MC. iLO supports up to 32 simultaneous sessions.

The specification also allows a given endpoint (identified by an IP address) on the LAN to open more than one session to a MC. This capability allows a single system to serve as a proxy providing MC LAN sessions for other systems. It is not intended for one system to use this provision to open multiple session to the MC for that systems sole use.

An IPMI messaging connection to the MC fits one of three classifications, session-less, single-session, or multi-session.

Session-less connections

A session-less connection is unauthenticated. There is no user login required for performing IPMI messaging. The System Interface and IPMB are examples of session-less connections.

A special case of a session-less connection can occur over an interface that supports session-based messaging. Session-based connections have certain commands that are accepted and responded to outside of a session. When that occurs, the channel is effectively operating in a session-less manner for those commands. Commands that are handled outside of a session have fixed values for session-specific fields in the message. For example, when the `Get Channel Authentication Capabilities` is sent over a LAN channel outside of a session, the Session ID is set to NULL and authentication type set to NONE in the IPMI session header. Note that commands accepted outside of a session can also be accepted within the context of a session, in which case they must have valid Session IDs, etc. in the session header to be accepted.

Session inactivity timeouts

A session is automatically closed if no new, valid message has been received for the session within the specified interval since the last message. The session must be re-authenticated to be restored. A remote console can optionally use the `Activate Session` command to keep a session open during periods of inactivity.

Note that only an active session will keep the `Session Inactivity Timeout` from expiring. IPMI message activity that occurs outside of an active session has no effect. This is to prevent someone from keeping a phone connection indefinitely while trying to guess different passwords to activate a session.

The MC only monitors for inactivity while the connection is switched over to the MC. Note that closing a session is not always the same as hanging up a modem connection. Serial/modem sessions are also automatically closed when the connection is switched over to the system, but the phone connection remains active. The MC only terminates the phone connection if a session is closed due to an inactivity timeout while the serial connection is routed to the MC.

The timeout and tolerance values are specified for the MC that will timeout and close the session. System software should take this tolerance into account, plus any additional delays due to media transmission times, etc.

An implementation can provide an option to allow timeout configuration via a parameter in the configuration parameters for the given channel type.

System interface messaging

Messaging between system software and the other management BUSes such as the IPMB, is accomplished using channels and a `Receive Message Queue`. A channel is a path through the MC that allows messages to be sent between the system interface and a given bus or message transport. The `Receive Message Queue` is used to hold message data for system software until system software can collect it. All channels share the `Receive Message Queue` for transferring messages to system management software. The `Receive Message Queue` data contains channel, session, and IPMI addressing information that allows system software to identify the source of the message, and to format a message back to the source if necessary.

System management software is responsible for emptying the `Receive Message Queue` whenever it has data in it. Messages are rejected if the Receive Message Queue gets full. It is recommended that the `Receive Message Queue` have at least two slots for each channel. The `Receive Message Queue` is a logical concept. An implementation may choose to implement it as an actual queue, or could implement separate internal buffers for each channel. It is recommended that the implementation attempt to leave a slot open for each channel that does not presently have a message in the queue. This helps prevent lockout by having the queue fill with just messages from one interface.

The MC itself can, if necessary, use the `Receive Message Queue` and `Messaging Channels` to send asynchronous messages to system management software. The recommended mechanism for accomplishing this is to define a unique channel with a protocol type of system. To send an asynchronous message to system software the MC would place a message from that channel directly into the `Receive Message Queue` in System format. System software would be able to respond back to the MC using a `Send Message` command for that channel.

Bridging

MC Messaging Bridging provides a mechanism for routing IPMI Messages between different media. Bridging is only specified for delivering messages between channels; it is not specified for delivering messages between two sessions on the same channel.

With IPMI 2.0, bridging is extended to support delivering IPMI messages between active connections/sessions on the same channel.

There are three mechanisms for bridging messages between different media connected to the MC, depending on the message target:

- MC LUN 10b — used for delivering messages to the System Interface. The MC automatically routes any messages it receives via LUN 10b to the Receive Message Queue.
- `Send Message`

  command from System Interface — used for delivering messages to other channels, such as the IPMB. The messages appear on the channel as if they've come from MC LUN 10b. Thus, if the message is a request message, the response goes to MC LUN 10b and the MC automatically places the response into the Receive Message Queue for retrieval. System software is responsible for matching the response up with the original request, thus the No Tracking setting in the

  `Send Message`

  command.
- `Send Message`

  command with response tracking — used with response tracking for bridging request messages to all other channels except when the System Interface is the source or destination of the message.

MC LUN 10b

Messages to SMS are always routed to the Receive Message Queue and the `Send Message` command is not typically used. Messages to SMS are delivered via the MC SMS LUN 10b. The MC automatically reformats and places any messages that are addressed to LUN 10b into the Receive Message Queue for SMS to retrieve using the `Get Message` command.

Sending a request to SMS requires formatting the command so that it is address to MC LUN 10b. SMS can retrieve the request from the Receive Message Queue, extract the originator's address and channel info, and then use the `Send Message` command to deliver a response.

The MC does not track requests and responses for messages to system software because the Receive Message Queue provides the channel and session information necessary to format the `Send Message` command to deliver the response. System software is capable of tracking the channel and session information it used when generating a request. The No Tracking option is used for `Send Message` commands from system software.

The responder then delivers its response message to MC LUN 10b and the response gets routed to the Receive Message Queue. Conversely, if a channel wants to deliver a message to SMS, it sends the request message to MC LUN 10b, and later SMS uses a `Send Message` command to return the response from MC LUN 10b.

Send Message command with response tracking

The `Send Message` command is used primarily to direct the MC to act as a proxy that translates a message from one IPMI messaging protocol to another. The MC formats the data for the target channel type and protocol and delivers it to the selected medium.

Media such as the IPMB do not include channel number and session information as part of their addressing information. As a result, request messages from another channel must be delivered as if they originated from the MC itself.

If the bridged message is a request, it is necessary for the MC to hold onto certain data, such as originating channel and session information, so that when the response message comes back it can reformat the response and forward it back to the originator of the request. The primary way the MC accomplishes this is by assigning a unique sequence number to each request that it genearates, and saving a set of information in a Pending Bridged Response table that is later used to reformat and route a response back to the originator of the request.

The sequence number returned in the response is used to look up who generated the original response, the saved formatting and address information. The MC then reformats and delivers the response to the original requester and deletes the request from its list of pending responses. The `Send Message` command includes a parameter that directs the MC to save translation information for and track outstanding request messages for the purpose of routing the response back to the originator of the `Send Message` command.

NOTE:

With the exception of messages to SMS, when the `Send Message` command is used to deliver a message to a given medium the message appears to have been originated by the MC. This means that a controller on the IPMB can't generically distinguish a bridged request from SMS from a bridged request from LAN.

Table 147: Message bridging mechanism by source and destination

Message Type and direction	Delivery Mechanism	MC tracks pending responses
Request or Response from system interface to any other channel	Send Message	no
Request or Response to system interface from any other channel	MC LUN 10b	no
Request from any channel except system interface to IPMB	Send Message	yes
Response from IPMB to any channel except system interface	MC LUN 00b	yes

Bridged Request Example

This example illustrates a `Send Message` command from the LAN being used to deliver a request to IPMB.

The MC uses the sequence number that it places on the bridged request to identify the channel where the request came from and where to send the response. It is important for the MC to ensure that unique sequence numbers are used for pending requests from each channel. It is also important that sequence numbers are unique for successive requests to a given responder. One way to manage sequence numbers to the IPMB is to track them on a per responder basis. This can be kept in a table of Pending Bridged Response information.

In order to get the response back to the LAN, the IPMB response must return the same sequence number that was passed in the request. The management controller uses the sequence number to look up the channel type specific addressing, sequence number, and security information that it stored when the request was forwarded. For example, if the channel type is LAN then the response message must be formatted up in an RMCP/UDP packet with the IP address of the requester, the sequence number passed in the original request, the appropriate security key information, and so on.

When a request message is bridged to another channel by encapsulating it in a `Send Message` command (from a source channel other than the system interface), the MC immediately returns a response to the `Send Message` command itself. Meanwhile, the request is extracted from the `Send Message` command and forwarded to the specified target channel.

The `Send Message` command must be configured to direct the MC to keep track of data in the request so when the response comes back from the target device it can be forwarded by the MC back to the channel that delivered the original `Send Message` command to the MC. When the response comes back from the target, the MC uses the tracking information to form at the response for the given channel. To the party that initiated the `Send Message` command, the response will appear as if the encapsulated request was directly executed by the MC.

For example, suppose a `Get Device ID` command has been encapsulated in a `Send Message` command directed to the IPMB from a LAN channel. The MC will immediately send a response to the `Send Message` command back on the LAN. The MC will extract the encapsulated `Get Device ID` message content and format it as a `Get Device ID` request for IPMB. The target device on IPMB responds with a Get Device ID response message in IPMB format. The MC takes the tracking information that was stored when the `Send Message` command was issued, and uses it to create a Get Device ID response in LAN format. The Responder's address information in that response can either be that of the MC, of the device on IPMB that the request was targeted at the choice of the MC implementation. Parties that

initiate this type of bridged request using the `Send Message` command should accept responses from the MC that use either address.

The following figure and steps present an example high-level design for handling a bridged request. Note that the example shows information that is generated and stored, but it does not show any particular code module that would perform that operation. That is, the choice of which functions are centralized, which are in a LAN module, and which are in an IPMB module is left to the implementer.

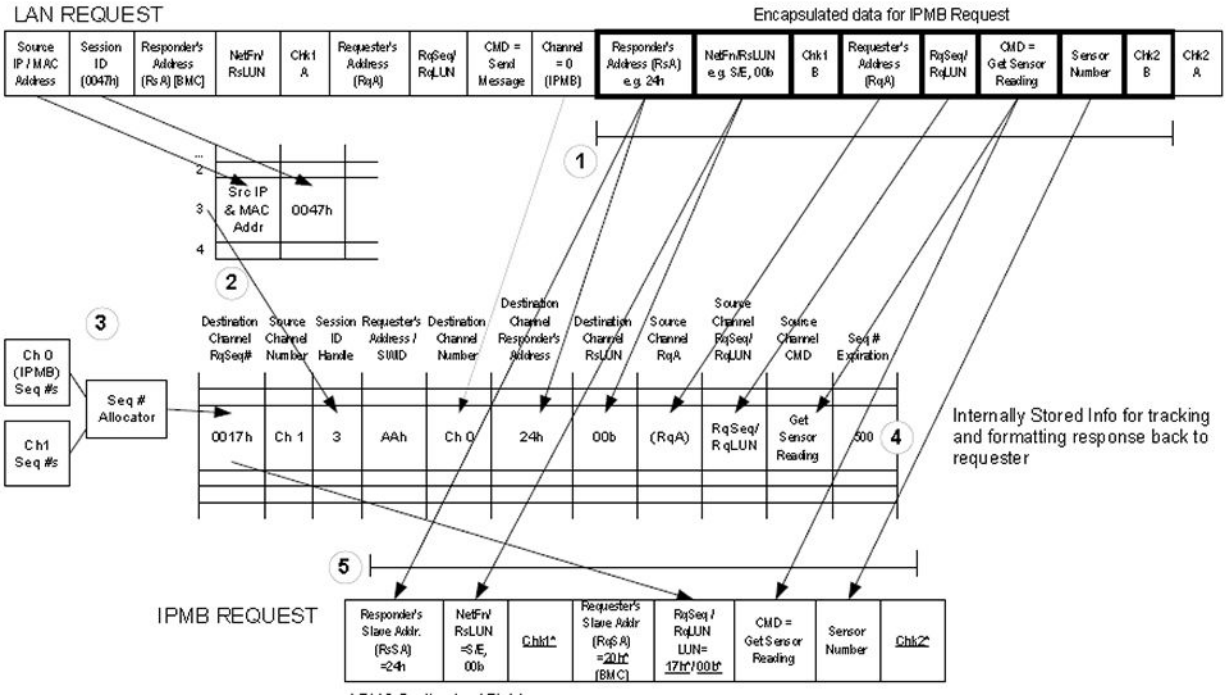


Figure 3: LAN to IPMB bridged request example

Procedure

1. When the MC receives the `Send Message` command with the Bridged Request parameter bit set, it checks for an available entry in a Pending Bridged Response table and copies parameters from the request to be bridged. When the response comes back, these parameters will be used to validate that the response matches the earlier request and to reformat the response for the originating channel. The bold outlined boxes represent parameters and data in the `Send Message` command that will ultimately be copied to the resulting request on the target channel.

2. Any channel session information necessary to get the response back to the original requester will also need to be recorded. In this example, the MC maintains a separate table of session information for the LAN channel. An offset into that table is used as a handle for identifying the session information associated with the request. This handle is used in the Pending Bridged Response table in place of copying all the session information. Note that with such an implementation, it is important to remember details such as invalidating and freeing any bridge table entry associated with that session if the session should get closed while responses are pending.

3. In this example, the MC has a separate Sequence Number Allocator routine that ensures that sequence numbers used in bridged requests are kept unique for a given channel. This is done so that the response comes back, the sequence number can be used to look up corresponding request info entries from the Pending Bridged Response table.

4. Responses have a five second Sequence Number Expiration interval. If a response is not received by the expiration interval, the corresponding entry in the Pending Bridged Response entry is deleted and the sequence number associated with the request can be reused. The Sequence Number Expiration column in this example represents a possible implementation where the value is decremented nominally once every10 ms. The entry is considered to be

free when the number reaches 0. In this example the Sequence Number Expiration field could be used both for tracking sequence number expiration as well as a mechanism for marking availability of the table entry.

5. The MC takes the indicated values and uses them to construct the bridged request. The request is a combination of field values copied from the original Send Message command and values generated by the MC. The MC generated values are shown with a bold, underlined typeface with an asterisk.

IPMB access via master write-read command

When an IPMB is implemented in the system, the MC serves as a controller to give system software access to the IPMB. The IPMB allows non-intelligent devices as well as management controllers on the bus. To support this operation, the MC provides the Master Write-Read command via its interface with system software. The Master Write-Read command provides low-level access to non-intelligent devices on the IPMB, such as FRU SEEPROMs.

The Master Write-Read command provides a subset of the possible I^2C and SM BUS operations that covers most I^2C/SM BUS-compatible devices.

In addition to supporting non-intelligent devices on the IPMB, the Master Write-Read command also provides access to non-intelligent devices on Private Busses behind management controllers. The main purpose of this is to support FRU SEEPROMs on Private Busses.

MC IPMB LUNs

A MC supports several LUNs to which messages are sent via the IPMB interface. These LUNs are used to identify different sub-addresses within the MC.

Table 148: MC IPMB LUNs

LUN	Short Description	Long Description
00b	MC commands and Event Request Messages	Event Request Messages received on this LUN are routed to the Event Receiver Function in the MC and automatically logged if SEL logging is enabled.
01b	OEM LUN 1	OEM — reserved for MC implementer/system integrator definition.
10b	SMS Message LUN (intended for messages to System Management Software)	Messages received on this LUN are routed to the Receiver Message Queue and retrieved using a Read Message command. The SMS_Avail flag is set whenever the Receive Message Queue has valid contents.
11b	OEM LUN 2	OEM — reserved for MC implementer/system integrator definition

Sending Messages to IPMB from system software

SMS can use the MC for sending and receiving IPMB messages. Both IPMB request and response messages can be sent and received using this mechanism. Therefore, not only can system software send requests to the IPMB and receive responses from the IPMB, it is also possible for system software to receive requests from the IPMB to send back IPMB responses.

System software sends messages to the IPMB through the system interface using the MC as an IPMB controller. This is accomplished by using the `Send Message` command to write the message to the IPMB (channel 0). The MC does not place any restrictions on the type or content of the IPMB message being sent. System management software can send any IPMB request or response message it desires provided that the message meets the maximum length requirements of the `Send Message` command.

System Management Software is responsible for providing all fields for the IPMB message, including Requester and Responder Slave addresses and checksums. The following figures show an example using the Send Message command to send a `Set Event Receiver` command to an IPMB device at slave address 52h, LUN 00b, via the system interface. The example command sets the Event Receiver address to 20h — MC.

The heavy bordered fields show the bytes for the IPMB message carried in the `Send Message` command. The requesters LUN field (rqLUN) is set to 10b (MC SMS LUN). This directs the responder to send the response to the `Set Event Receiver` command to the system node's Receive Message Queue.

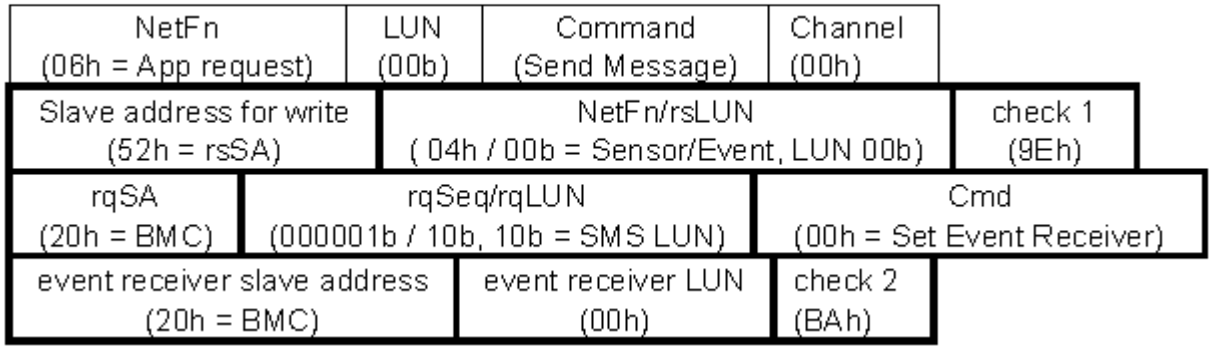


Figure 4: IPMB request sent using `Send Message` command

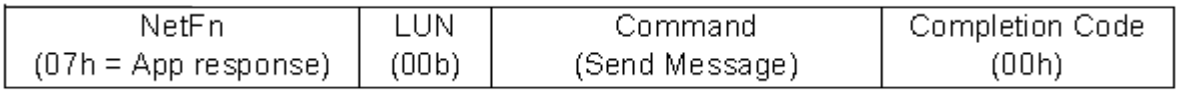


Figure 5: `Send Message` command response

The response is for the `Send Message` command and not for the `Set Event Receiver` command. The response to the `Set Event Receiver` command is returned later in the Receive Message Queue. System software uses the `Get Message` command to read messages from the Receive Message Queue. System software keeps track of any outstanding responses and matches responses up with corresponding requests as they are returned. System software must also keep track of the protocol assigned to the particular channel in order to interpret the response to the `Get Message` command.

Keyboard Controller Style Interface

The Keyboard Controller Style (KCS) is one of the supported MC to SMS interfaces. The KCS interface is specified solely for SMS messages.

The KCS Interface supports polled operations. Implementations optionally provide an interrupt driven by the OBF flag, this must not prevent driver software from using the interface in a polled manner. This allows software to default to polled operation. It also allows software to use the KCS interface in a polled mode until it determines the type of interrupt support. Methods for assigning and enabling such an interrupt are outside the scope of this specification.

KCS Interface/MC LUNs

LUN 00b is typically used for all messages to the MC through the KCS Interface. LUN 10b is reserved for Receive Message Queue use and should not be used for sending commands to the MC. Note that messages encapsulated in a `Send Message` command can use any LUN in the encapsulated portion.

KCS Interface-MC Request message format

Request Messages are sent to the MC from system software using a write transfer through the KCS interface. The message bytes are organized according to the following format specification:

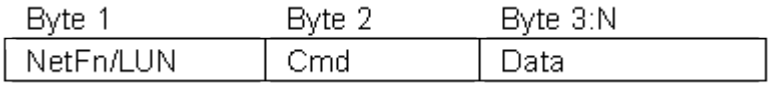


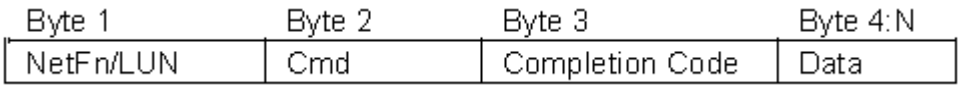
Figure 6: KCS Interface/MC Request Message format

Where:

LUN	This is a sub-address that allows messages to be routed to different LUNS that reside behind the same physical interface. The LUN field occupies the least significant two bits of the first message byte.
NetFN	Provides the first level of functional routing for messages received by the MC via the KCS Interface. The NetFn field occupies the most significant six bits of the first message byte. Even NetFn values are used for requests to the MC, and odd NetFn values are returned in responses from the MC.
Cmd	This message byte specifies the operation that is to be executed under the specified Network Function.
Data	Zero or more bytes of data, as required by the given command. The general convention is to pass data LS-byte first, but check the individual command specifications to be sure.

MC-KCS Interface Response Message format

Response Messages are Read Transfers from the MC to system software via the KCS interface. The MC only returns responses via the KCS Interface when data needs to be returned. The message bytes are organized according to the following format specification:



Where:

LUN	Returns the LUN that was passed in the Request Message.
NetFn	A return of the NetFn code that was passed in the Request Message. Except that an odd NetFn value is returned.
Cmd	Return of the Cmd code that was passed in the Request Message.
Completion Code	Indicates whether the request completed successfully.
Data	Zero or more bytes of data. The MC always returns a response to acknowledge the request, regardless of whether data is returned.

LAN Interface

The LAN interface specifications define how IPMI messages can be sent to and from the MC encapsulated in Remote Management Control Protocol (RMCP) packet datagrams. This capability is also referred to as IPMI over LAN. IPMI also defines the associated LAN specific configuration interfaces for settings things such as IP address other options, as well as commands for discovering IPMI based systems.

The Distributed Management Task Force (DMTF) specifies the RMCP format. This same packet format is used for non-IPMI messaging via the DMTF's Alert Standard Forum (ASF) specification. Using the RMCP packet format enables more commonality between management applications that operate in an environment that includes both IPMI based and ASF based systems.

IPMI v2.0 defines and extended packet format and capabilities that are collectively referred to as RMCP+ and defined under the IPMI specific portion of an RMCP packet. RMCP+ utilizes authentication algorithms that are more closely aligned with the mechanisms used for the ASF 2.0 specification. In addition, RMCP+ adds data confidentiality (encryption) and payload capability. In iLO, IPMI v2.0/RMCP+ is implemented for the LAN interface.

LAN alerting

IPMI supports LAN alerting in the form of SNMP traps that follow the Platform Even Trap (PET) format. SNMP traps are typically sent as unreliable datagrams. However, IPMI includes a PET Acknowledge and Retry option that allows an IPMI aware remote application to provide a positive acknowledge that the trap was received.

IPMI LAN interface

This section describes the mechanisms specific to transferring IPMI messages between the MC and a remote management system (remote console) over an Ethernet LAN connection using UDP under IPv4. The UPD datagrams are formatted to contain IPMI request and response messages, plus additional messages for discovery and authentication.

While an IPMI LAN interface can be accomplished using a LAN Controller that is dedicated to the MC, it will usually be accomplished using LAN Controller that can be shared for both MC and system use.

There are two implementation that are likely to be used to deploy an IPMI LAN interface using a shared LAN controller. The first implementation is using an embedded LAN controller, as shown in **Figure 7: Embedded LAN Controller Implementation**, and the second is using a LAN controller on an add-in card, as shown in **Figure 8: PCI Management BUS Implementation**

Both examples show a LAN Controller with the capability to detect UDP datagrams sent to a management port. Any datagrams received on that port are forwarded to a side-band interface that allows them to be delivered to or retrieved by the MC. These incoming platform management datagrams may also be delivered to system software in parallel with being delivered to the MC.

The MC can use this same interface to inject datagrams onto the LAN. These datagrams are interleaved with the network packets generated by the system software.

The LAN controller can be designed so that the interface for the management port is powered by standby power and remains operative even when the system is powered down. This provides a mechanism that allows IPMI LAN messaging to occur independent from system software and the systems power state. A LAN controller dedicated to the MC can also be used.

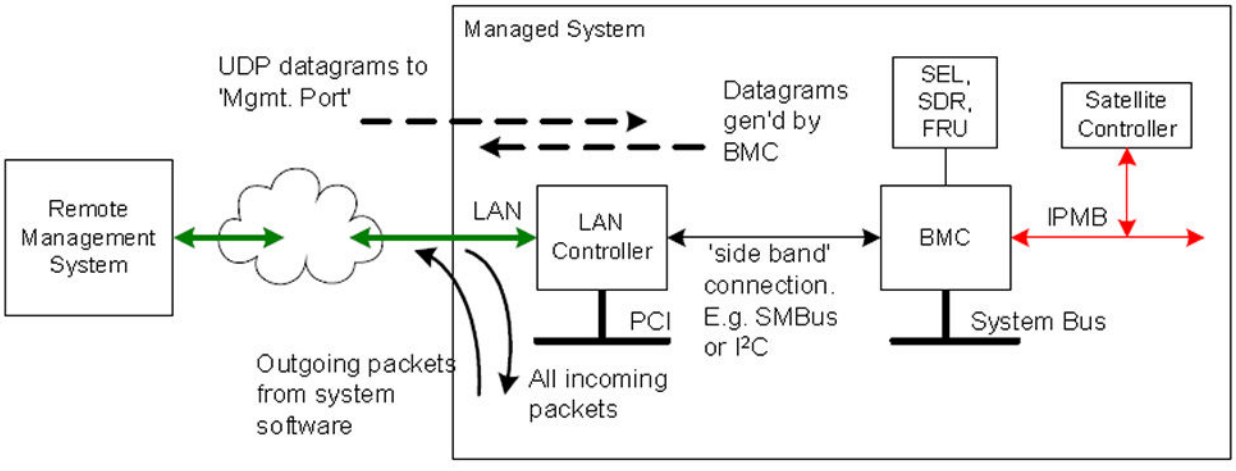


Figure 7: Embedded LAN Controller Implementation

Figure 8: PCI Management BUS Implementation shows an implementation where the LAN controller is implemented as a PCI add-in card connected to the MC via a PCI Management BUS connection. This approach avoids the need to have the LAN Controller built into the system, allowing the LAN controller portion of the IPMI LAN Interface to be added or updated at a later time.

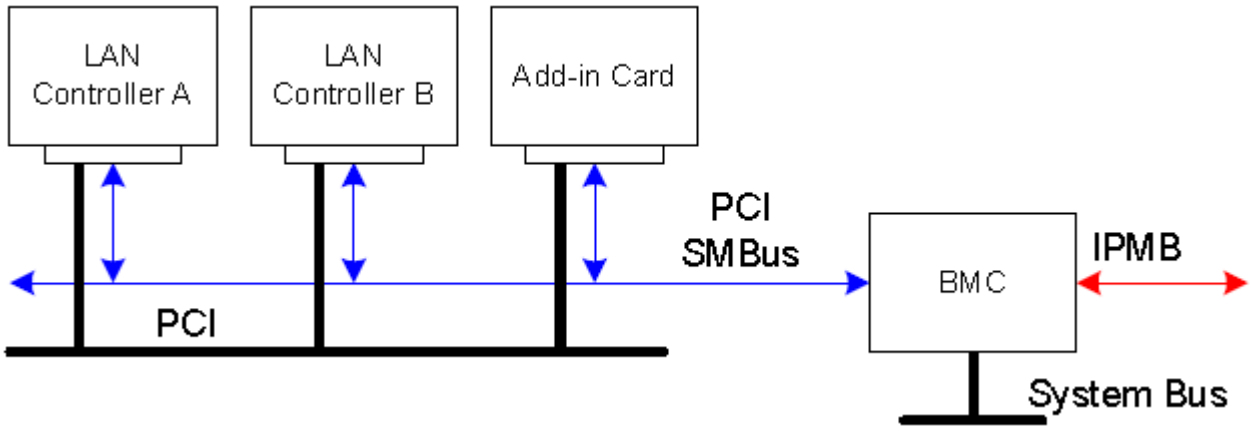


Figure 8: PCI Management BUS Implementation

Remote Management Control Protocol (RMCP)

The Distributed Management Task Force (DMTF) has defined a RMCP for supporting pre-OS and OS absent management. RCMP is a simple request-response protocol that can be delivered using UDP datagrams. IPMI-over-LAN uses version 1 of the RMCP protocol and packet format.

RMCP includes a field that indicates the class of messages that can be embedded in an RMCP message packet, including a class for IPMI messages. Other message classes are ASF and OEM.

An IPMI LAN implementation can also use ASF-class Ping and Pong messages to support the discovery of IPMI managed systems on the network.

RMCP port numbers

RMCP uses two well-known ports under UDP. The following *RMCP Port Numbers* table describes these ports and summarizes their use.

Table 149: RMCP Port Numbers

Port#	Name	Description
623 (26Fh)	Aux bus Shunt (Primary RMCP Port)	The Primary RMCP Port — This port and the required RMCP messages must be provided to conform with RMCP specifications. There is a mandatory set of messages required to be supported on this port. These messages are always sent in the clear so that system software can discover systems that have RMCP support.
664 (298h)	Secure Aux BUS (Secondary RMCP Port)	Referred to as the secondary RMCP port or secure port, it is only used when it is necessary to encrypt packets using an algorithm or specification that prevents also sending unencrypted packets from being transferred via the same port. Since discovery requires sending in the clear RMCP ping/pong packets, the secondary port is used to transfer encrypted transfers while the primary port continues to support unencrypted packets. An implementation that utilizes this port must still support the Primary RMCP port and the required messages on that port in order to be conformant with the RMCP specifications. Note that the common IPMI messaging protocols and authentication mechanisms in this pecification do not use encrypted packets at the RMCP level (encrypted packets in IPMI are defined unter the IPMI message class), therefore IPMI messaging does not need to use the secondary port.

RMCP Message Format

There are two types of RMCP messages: Data or Normal RMCP messages and RMCP acknowledge messages. Data messages and ACK messages are differentiated by the ACK/normal bit of the class of message field.

Table 150: RMCP Message Format

Field	Size in bytes	Description
RMCP Header		
Version	1	06h = RMCP Version 1.0
Reserved	1	00h
Sequence Number	1	Varies, see text

Table Continued

Field	Size in bytes	Description
Class of Message	1	This field identifies the form of the messages that follow this header. All messages of class ASF (6) conform to the formats defined in this specification and can be extended via an OEM IANA. Bit 7 RMCP ACK 0 — Normal RMCP message 1 — RMCP ACK message Bit 6:5 Reserved Bit 4:0 Message Class 0–5 = Reserved 6 = ASF 7 = IPMI 8 = OEM defined all other = Reserved
RMCP Data		
Data	Variable	data based class of message

The following table presents how the ACK/Normal Bit and the message class combine to identify the type of message under RMCP and which specification defines the format of the associated message data.

Table 151: Message Type Determination Under RMCP

ACK/Normal bit	Message Class	Message Type	Message Data
ACK	ASF	RMCP ACK	No Data. Message contains only RMCP heading, and sequence number from the last message received.
ACK	all other	undefined	not allowed
normal	ASF	ASF Messages	Per ASF specification
normal	OEM	OEM message under RMCP	bytes:3 = OEM IANA
			bytes 4:N = OEM message data (defined by manufacturer or organization identified by the OEM IANA field value)
normal	IPMI	IPMI messages	Per this specification

Serial Over LAN (SOL)

SOL is the name for the redirection of baseboard serial controller traffic over an IPMI session. This enables asynchronous serial-based OS and pre-OS communication over a connection to the MC. SOL can be used to provide a user at a remote console a means of interacting with serial text-based applications over the IPMI LAN session. A single remote console application can use SOL to simultaneously provide LAN access to IPMI platform management and serial text redirection under a unified user interface. SOL is implemented as a payload type under the IPMI v2.0 RMCP+ protocol. Access privileges for SOL are managed under the same user configuration interfaces that are used for IPMI management. This simplifies the creation of configuration software, remote management applications, and cross-platform configuration utilities.

iLO Command number assignments and privilege levels

The following section lists the commands defined in this specification and the minimum privilege level required to execute a given command.

Consider the following:

- Unless otherwise specified, unauthenticated, sessionless interfaces, such as the System Interface and IPMB, can support any IPMI command.
- The privilege level requirements for OEM commands (NetFn=OEM, OEM/Group) are specified by the OEM identified by the corresponding manufacturer ID.
- The `Send Message` and `Master Write-Read` commands are not available at the User privilege level, with the exception of using a `Send Message` command to deliver a message to the System Interface. This limit exists because these commands enable unfiltered access to the IPMB, ICMB, private management buses, and PCI Management Bus. Unfiltered access could allow someone to use these commands to send commands to other controllers, or to write to nonintelligent devices on those buses. As a consequence, a User is only able to read FRU and sensors directly managed by the management controller. In addition, FRU must be accessed through the `Read FRU` command and not `Master Write-Read`.
- The `Send Message` command can be used to deliver a message to the System Interface at User privilege level. It is up to the system software to determine the privilege level and place any additional restrictions on messages received through the Receive Message Queue. This can be accomplished by using the session handle associated with the message and the `Get Session Info` command to look up the privilege level that the user is operating at. Software can also check the limits for the channel and the user by using information from the `Get Channel Access` and `Get User Access` commands to determine whether a user has sufficient privileges to deliver a command to system software.

Unless otherwise specified, the listed IPMI commands, if supported, must be accessible through LUN 00b.

Key for the iLO Command number assignments and privilege level tables

p

The command works at any privilege level, and can be sent prior to a session being established.

s

The command is executable via system interface only.

X

The command is supported at the given privilege level or higher.

I

The command is executable from local interfaces only. For example, IPMB, SMBus, PCI Mgmt. bus, or System Interface)

C

Callback privilege

U

User Privilege level

O

Operator Privilege level

A

Administrator Privilege level

App

Application Network Function Code

S/E

Sensor/Event Network Function Code

- =

Reserved, unassigned, or OEM-specified

Table 152: IPM device “global” commands

Command	NetFn	CMD	C	U	O	A
Get Device ID	App	01h		X		
Broadcast ‘Get Device ID’[1]	App	01h	I	I	I	I
Cold Reset	App	02h				X
Warm Reset	App	03h				X
Get Self Test Results	App	04h		X		
Get ACPI Power State	App	07h		X		

[1] This command is sent using the Broadcast format on IPMB. See the command description for details.

Table 153: Management controller watchdog timer commands

	NetFn	CMD	C	U	O	A
Reset Watchdog Timer	App	22h			X	
Set Watchdog Timer	App	24h			X	
Get Watchdog Timer	App	25h		X		

Table 154: Management controller device and messaging commands

	NetFn	CMD	C	U	O	A
Set BMC Global Enables	App	2Eh	s	s	s	s
Get BMC Global Enables	App	2Fh		X		
Clear Message Flags	App	30h	s	s	s	s
Get Message Flags	App	31h	s	s	s	s
Enable Message Channel Receive	App	32h	s	s	s	s
Get Message	App	33h	s	s	s	s
Send Message	App	34h		X[1]	X	
Get System GUID	App	37h	p[2]	p[2]	p[2]	p[2]
Set System Info Parameters	App	58h				X
Get System Info Parameters	App	59h		X		
Set Session Privilege Level	App	3Bh		X[3]		
Close Session	App	3Ch	X[4]			
Get Session Info	App	3Dh		X		
Get AuthCode	App	3Fh			X	
Set Channel Access	App	40h				X
Get Channel Access	App	41h		X		
Get Channel Info	App	42h		X		
Set User Access	App	43h				X
Get User Access	App	44h			X	
Set User Name	App	45h				X
Get User Name	App	46h			X	
Set User Password	App	47h				X

Table Continued

	NetFn	CMD	C	U	O	A
Activate Payload	App	48h	X[4]	X[5]	X[5]	X[5]
Deactivate Payload	App	49h	X[4]	X[5]	X[5]	X[5]
Get Payload Activation Status	App	4Ah		X		
Get Payload Instance Info	App	4Bh		X		
Set User Payload Access	App	4Ch				X
Get User Payload Access	App	4Dh			X	
Get Channel Payload Support	App	4Eh		X		
Get Channel Payload Version	App	4Fh		X		
Master Write-Read	App	52h			X	
Get Channel Cipher Suites	App	54h	p	p	p	p
Suspend/Resume Payload Encryption	App	55h		X[6]		
Set Channel Security Keys	App	56h				X
Get System Interface Capabilities	App	57h		X		

[1] A User can use a Send Message command to deliver a message to system software. The Operator privilege is required to use the command to access other channels.

[2] Command applies only to authenticated channels.

[3] This is effectively a no-op if the user has a maximum privilege limit of User since the command could not be used to change the operating privilege level to a higher value.

[4] A session operating at Callback, User, or Operator can only use this command to terminate their own session. An Administrator or system software can use the command to terminate any session.

[5] The configuration parameters for a given payload type determine the privilege level required to activate or deactivate the payload.

[6] The Suspend/Resume Payload Encryption command may be overridden by a configuration option for the particular payload type that forces encryption to be used. In this case, an Admin level command would typically be required to change the configuration.

Table 155: Chassis device commands

	NetFn	CMD	C	U	O	A
Get Chassis Capabilities	Chassis	00h		X		
Get Chassis Status	Chassis	01h		X		

Table Continued

	NetFn	CMD	C	U	O	A
Chassis Control	Chassis	02h			X	
Chassis Identify	Chassis	04h			X	
Set Power Restore Policy	Chassis	06h			X	
Set System Boot Options	Chassis	08h			X[1]	
Get System Boot Options	Chassis	09h			X	
unassigned	Chassis	0Ch-0Eh	-	-	-	-
Get POH Counter	Chassis	0Fh		X		

[1] There is a bit in this command that can only be set at Administrator privilege level.

Table 156: Event commands

Command	NetFn	CMD	C	U	O	A
Set Event Receiver	S/E	00h				X
Get Event Receiver	S/E	01h		X		
Platform Event (also called Event Message)	S/E	02h			X	
unassigned	S/E	03h-0Fh	-	-	-	-

Table 157: Sensor device commands

Command	NetFn	CMD	C	U	O	A
Get Device SDR Info	S/E	20h	I	I	I	I
Get Device SDR	S/E	21h	I	I	I	I
Reserve Device SDR Repository						
Get Sensor Reading	S/E	2Dh		X		

Table 158: FRU device commands

	NetFn	CMD	C	U	O	A
Get FRU Inventory Area Info	Storage	10h		X		
Read FRU Data	Storage	11h		X		
Write FRU Data	Storage	12h			X	

Table 159: SDR device commands

Command	NetFn	CMD	C	U	O	A
Get SDR Repository Info	Storage	20h		X		
Get SDR Repository Allocation Info	Storage	21h		X		
Reserve SDR Repository	Storage	22h		X		
Get SDR	Storage	23h		X		
Add SDR	Storage	24h			X	
Delete SDR	Storage	26h			X	
Clear SDR Repository	Storage	27h			X	
Run Initialization Agent	Storage	2Ch			X	

Table 160: LAN device commands

Command	NetFn	CMD	C	U	O	A
Set LAN Configuration Parameters	Transport	01h				X
Get LAN Configuration Parameters	Transport	02h			X	

Table 161: Serial/modem device commands

Command	NetFn	CMD	C	U	O	A
Set SOL Configuration Parameters	Transport	21h				X
Get SOL Configuration Parameters	Transport	22h		X		

Table 162: DCMI-specific commands

Command	NetFn	CMD	C	U	O	A
Get DCMI Capability Info	DCGRP (2ch)	01h		X		
Get Asset Tag	DCGRP (2ch)	06h		X		
Get DCMI Sensor Info	DCGRP (2ch)	07h			X	
Set Asset Tag	DCGRP (2ch)	08h			X	
Get Controller ID String	DCGRP (2ch)	09h		X		
Set Controller ID String	DCGRP (2ch)	0Ah			X	

Table 163: PICMG-specific commands

Command	NetFn	CMD	C	U	O	A
Get PICMG Properties	PICMG (00h)	00h		X		
Get Address Info	PICMG (00h)	01h		X		
FRU Inventory Device Lock Control	PICMG (00h)	1Fh			X	

Verbose output examples

Example using the `fru print -v` command

```
root@USER:~# ipmitool -I lanplus -H <iLO IP address> -u <username> -p <password> fru
print -v

FRU Device Description : Builtin FRU Device (ID 0)
 Chassis Type          : Rack Mount Chassis
 Chassis Serial        : XXXXXXXXXX
 Board Mfg Date        : Tue Apr 25 05:30:00 2017
 Board Mfg             : HPE
 Board Product         : ProLiant DL580 Gen10
 Board Serial          : XXXXXXXXXX
 Board Part Number     : WC1949777007
 Product Manufacturer  : HPE
 Product Name          : ProLiant DL580 Gen10
 Product Part Number   : WC1949777007
 Product Serial        : XXXXXXXXXX

FRU Device Description : BMC CONTROLLER (ID 238)
 Product Manufacturer  : HPE
 Product Name          : BMC CONTROLLER
 Product Part Number   : iLO 5

FRU Device Description : MB BIOS (ID 239)
 Product Manufacturer  : HPE
 Product Name          : SYSTEM BIOS
 Product Part Number   : U34
 Product Version       : 02/02/2019

FRU Device Description : CPU 1 (ID 16)
 Product Manufacturer  : Intel(R) Corporation
 Product Name          : Intel(R) Xeon(R) Gold 6134 CPU @ 3.20GHz

FRU Device Description : CPU 2 (ID 17)
 Product Manufacturer  : Intel(R) Corporation
 Product Name          : Intel(R) Xeon(R) Gold 6134 CPU @ 3.20GHz

FRU Device Description : PCI-E Slot 7 (ID 2)
 Device not present (Requested sensor, data, or record not found)

FRU Device Description : Ethernet Adptr (ID 3)
 Board Mfg Date        : Sat Feb 11 03:20:00 2017
 Board Mfg             : HPE
 Board Product         : HPE Ethernet 1Gb 4-port 331FLR Adapter
 Board Serial          : XXXXXXXXXX
 Board Part Number     : 789897-001
 Board FRU ID          : 07/08/14
 Board Extra           : d23041
 Board Extra           : d5629133b002
 Product Manufacturer  : HPE
 Product Name          : HPE Ethernet 1Gb 4-port 331FLR Adapter
 Product Part Number   : 629135-B22
 Product Version       : 00
 Product Serial        : XXXXXXXXXX

FRU Device Description : Embedded 0 (ID 4)
 Board Mfg Date        : Fri Feb 17 05:30:00 2017
 Board Part Number     : 871264-001
```

```
 Board FRU ID          : 05/04/16
 Product Manufacturer  : STL
 Product Name          : HPE Smart Storage Bat(ery
 Product Part Number   : 727258-B21
 Product Version       : 01
 Product Serial        : XXXXXXXXXXXXXX
 Power Supply Record
  Capacity                   : 96 W
  Peak VA                    : 96 VA
  Inrush Current             : 30 A
  Inrush Interval            : 5 ms
  Input Voltage Range 1      : 90-132 V
  Input Voltage Range 2      : 180-264 V
  Input Frequency Range      : 47-63 Hz
  A/C Dropout Tolerance      : 10 ms
  Flags                      : 'Power factor correction' 'Hot swap'
  Peak capacity              : 96 W
  Peak capacity holdup       : 2 s
  Combined capacity          : not specified
 DC Output Record
  Output Number              : 1
  Standby power              : No
  Nominal voltage            : 7.20 V
  Max negative deviation     : 6.00 V
  Max positive deviation     : 8.00 V
  Ripple and noise pk-pk     : 120 mV
  Minimum current draw       : 0.000 A
  Maximum current draw       : 1.600 A
 DC Output Record
  Output Number              : 2
  Standby power              : Yes
  Nominal voltage            : 7.20 V
  Max negative deviation     : 6.00 V
  Max positive deviation     : 8.00 V
  Ripple and noise pk-pk     : 120 mV
  Minimum current draw       : 0.000 A
  Maximum current draw       : 1.600 A

FRU Device Description : Embedded 0 (ID 5)
 Board Mfg Date        : Thu Nov 15 05:30:00 2018
 Board Part Number     : 878643-001
 Board FRU ID          : 03/10/17
 Product Manufacturer  : STL
 Product Name          : HPE Smart Storage Battery
 Product Part Number   : 875241-B21
 Product Version       : 01
 Product Serial        : XXXXXXXXXXXXXX
 Power Supply Record
  Capacity                   : 96 W
  Peak VA                    : 96 VA
  Inrush Current             : 30 A
  Inrush Interval            : 5 ms
  Input Voltage Range 1      : 90-132 V
  Input Voltage Range 2      : 180-264 V
  Input Frequency Range      : 47-63 Hz
  A/C Dropout Tolerance      : 10 ms
  Flags                      : 'Power factor correction' 'Hot swap'
  Peak capacity              : 96 W
  Peak capacity holdup       : 2 s
  Combined capacity          : not specified
 DC Output Record
  Output Number              : 1
```

```
  Standby power              : No
  Nominal voltage            : 7.20 V
  Max negative deviation     : 6.00 V
  Max positive deviation     : 8.00 V
  Ripple and noise pk-pk     : 120 mV
  Minimum current draw       : 0.000 A
  Maximum current draw       : 1.600 A
 DC Output Record
  Output Number              : 2
  Standby power              : Yes
  Nominal voltage            : 7.20 V
  Max negative deviation     : 6.00 V
  Max positive deviation     : 8.00 V
  Ripple and noise pk-pk     : 120 mV
  Minimum current draw       : 0.000 A
  Maximum current draw       : 1.600 A

FRU Device Description : PCI-E Slot 1 (ID 6)
 Device not present (Requested sensor, data, or record not found)

FRU Device Description : PCI-E Slot 2 (ID 7)
 Device not present (Requested sensor, data, or record not found)

FRU Device Description : PCI-E Slot 3 (ID 8)
 Device not present (Requested sensor, data, or record not found)

FRU Device Description : PCI-E Slot 4 (ID 9)
 Device not present (Requested sensor, data, or record not found)

FRU Device Description : PCI-E Slot 5 (ID 10)
 Device not present (Requested sensor, data, or record not found)

FRU Device Description : PCI-E Slot 6 (ID 11)
 Device not present (Requested sensor, data, or record not found)

FRU Device Description : SAS Ctlr (ID 12)
 Product Name          : Empty

FRU Device Description : Video (ID 13)
 Product Name          : Embedded Video Controller
```

Example using the `sdr list all -v` command

```
root@USER:~#ipmitool -I lanplus -H <iLO IP Address> -u <username> -p <password> sdr
list all -v

Sensor ID              : UID (0xae)
 Entity ID             : 23.1 (System Chassis)
 Sensor Type (Discrete): Unknown (0xc0)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 OEM                   : 0

Sensor ID              : SysHealth_Stat (0xac)
 Entity ID             : 23.1 (System Chassis)
 Sensor Type (Discrete): Chassis (0x18)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Chassis
                         [Transition to Non-critical from OK]
 Assertions Enabled    : Chassis
                         [Transition to Non-critical from OK]
```

```
                        [Transition to Non-recoverable from less severe]
                        [Transition to Non-critical from more severe]
 OEM                  : 0

Sensor ID             : 01-Inlet Ambient (0xa2)
 Entity ID            : 55.1 (Air Inlet)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading       : 17 (+/- 0) degrees C
 Status               : ok
 Positive Hysteresis  : Unspecified
 Negative Hysteresis  : Unspecified
 Minimum sensor range : Unspecified
 Maximum sensor range : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds  : ucr unr
 Settable Thresholds  :
 Threshold Read Mask  : ucr unr
 Assertions Enabled   : ucr+ unr+

Sensor ID             : 02-CPU 1 (0x2)
 Entity ID            : 3.1 (Processor)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading       : 40 (+/- 0) degrees C
 Status               : ok
 Positive Hysteresis  : Unspecified
 Negative Hysteresis  : Unspecified
 Minimum sensor range : 110.000
 Maximum sensor range : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds  : ucr
 Settable Thresholds  :
 Threshold Read Mask  : ucr
 Assertions Enabled   : ucr+

Sensor ID             : 03-CPU 2 (0x3)
 Entity ID            : 3.2 (Processor)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading       : 40 (+/- 0) degrees C
 Status               : ok
 Positive Hysteresis  : Unspecified
 Negative Hysteresis  : Unspecified
 Minimum sensor range : 110.000
 Maximum sensor range : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds  : ucr
 Settable Thresholds  :
 Threshold Read Mask  : ucr
 Assertions Enabled   : ucr+

Sensor ID             : 04-CPU 3 (0x4)
 Entity ID            : 3.3 (Processor)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading       : Disabled
 Status               : Not Available
 Positive Hysteresis  : Unspecified
 Negative Hysteresis  : Unspecified
 Minimum sensor range : 110.000
 Maximum sensor range : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds  : ucr
 Settable Thresholds  :
```

```
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 05-CPU 4 (0x5)
 Entity ID             : 3.4 (Processor)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 110.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 06-P1 DIMM 1-6 (0x24)
 Entity ID             : 32.11 (Memory Device)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 21 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 07-NV DIMM 1-6 (0xfe)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 08-P1 DIMM 7-12 (0x25)
 Entity ID             : 32.12 (Memory Device)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
```

```
 Assertions Enabled    : ucr+

Sensor ID              : 09-NV DIMM 7-12 (0xfd)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 10-P2 DIMM 1-6 (0x26)
 Entity ID             : 32.13 (Memory Device)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 18 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 11-NV DIMM 1-6 (0xfc)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 12-P2 DIMM 7-12 (0x27)
 Entity ID             : 32.14 (Memory Device)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+
```

```
Sensor ID              : 13-NV DIMM 7-12 (0xfb)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 14-P3 DIMM 1-6 (0x28)
 Entity ID             : 32.15 (Memory Device)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 15-NV DIMM 1-6 (0xfa)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 16-P3 DIMM 7-12 (0x29)
 Entity ID             : 32.16 (Memory Device)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+
```

```
Sensor ID              : 17-NV DIMM 7-12 (0xf9)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 18-P4 DIMM 1-6 (0x2a)
 Entity ID             : 32.17 (Memory Device)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 19-NV DIMM 1-6 (0xf8)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 20-P4 DIMM 7-12 (0x2b)
 Entity ID             : 32.18 (Memory Device)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 21-NV DIMM 7-12 (0xf7)
```

```
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 22-HD Max (0x97)
 Entity ID             : 4.2 (Disk or Disk Bay)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 35.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 23-Exp Bay Drive (0x96)
 Entity ID             : 4.1 (Disk or Disk Bay)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 35.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 24-Stor Batt 1 (0xbf)
 Entity ID             : 40.1 (Battery)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 17 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 25-Stor Batt 2 (0xc0)
 Entity ID             : 40.2 (Battery)
```

```
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 17 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 26-VR P1 (0xa)
 Entity ID             : 20.1 (Power Module)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 23 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 27-VR P2 (0xb)
 Entity ID             : 20.2 (Power Module)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 23 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 28-VR P3 (0xc)
 Entity ID             : 20.3 (Power Module)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 29-VR P4 (0xd)
 Entity ID             : 20.4 (Power Module)
 Sensor Type (Threshold)  : Temperature (0x01)
```

```
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 30-VR P1 Mem 1 (0x61)
 Entity ID             : 20.21 (Power Module)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 21 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 31-VR P1 Mem 2 (0x62)
 Entity ID             : 20.22 (Power Module)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 20 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 32-VR P2 Mem 1 (0x63)
 Entity ID             : 20.23 (Power Module)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 18 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 33-VR P2 Mem 2 (0x64)
 Entity ID             : 20.24 (Power Module)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 20 (+/- 0) degrees C
```

```
 Status                 : ok
 Positive Hysteresis    : Unspecified
 Negative Hysteresis    : Unspecified
 Minimum sensor range   : Unspecified
 Maximum sensor range   : Unspecified
 Event Message Control  : Global Disable Only
 Readable Thresholds    : ucr unr
 Settable Thresholds    :
 Threshold Read Mask    : ucr unr
 Assertions Enabled     : ucr+ unr+

Sensor ID               : 34-VR P3 Mem 1 (0x65)
 Entity ID              : 20.25 (Power Module)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading         : Disabled
 Status                 : Not Available
 Positive Hysteresis    : Unspecified
 Negative Hysteresis    : Unspecified
 Minimum sensor range   : Unspecified
 Maximum sensor range   : Unspecified
 Event Message Control  : Global Disable Only
 Readable Thresholds    : ucr unr
 Settable Thresholds    :
 Threshold Read Mask    : ucr unr
 Assertions Enabled     : ucr+ unr+

Sensor ID               : 35-VR P3 Mem 2 (0x66)
 Entity ID              : 20.26 (Power Module)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading         : Disabled
 Status                 : Not Available
 Positive Hysteresis    : Unspecified
 Negative Hysteresis    : Unspecified
 Minimum sensor range   : Unspecified
 Maximum sensor range   : Unspecified
 Event Message Control  : Global Disable Only
 Readable Thresholds    : ucr unr
 Settable Thresholds    :
 Threshold Read Mask    : ucr unr
 Assertions Enabled     : ucr+ unr+

Sensor ID               : 36-VR P4 Mem 1 (0x67)
 Entity ID              : 20.27 (Power Module)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading         : Disabled
 Status                 : Not Available
 Positive Hysteresis    : Unspecified
 Negative Hysteresis    : Unspecified
 Minimum sensor range   : Unspecified
 Maximum sensor range   : Unspecified
 Event Message Control  : Global Disable Only
 Readable Thresholds    : ucr unr
 Settable Thresholds    :
 Threshold Read Mask    : ucr unr
 Assertions Enabled     : ucr+ unr+

Sensor ID               : 37-VR P4 Mem 2 (0x68)
 Entity ID              : 20.28 (Power Module)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading         : Disabled
 Status                 : Not Available
```

```
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 38-Front Ambient (0xa3)
 Entity ID             : 55.2 (Air Inlet)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 17 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 39-VR P2Mem Zone (0xf6)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 17 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 40-Chipset (0xa4)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 28 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 41-BMC (0xf5)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 53 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
```

```
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 42-BMC Zone (0xf4)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 24 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 43-LOM Card (0xb5)
 Entity ID             : 11.4 (Add-in Card)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 42 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 40.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 44-I/O Zone (0xf3)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 23 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 45-PCI 1 (0x84)
 Entity ID             : 49.1 (PCI Express Bus)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
```

```
 Minimum sensor range  : 40.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 46-PCI 1 Zone (0xf2)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 18 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 47-PCI 2 (0x85)
 Entity ID             : 49.2 (PCI Express Bus)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 40.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 48-PCI 2 Zone (0xf1)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 18 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 49-PCI 3 (0x86)
 Entity ID             : 49.3 (PCI Express Bus)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 40.000
```

```
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 50-PCI 3 Zone (0xf0)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 18 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 51-PCI 4 (0x87)
 Entity ID             : 49.4 (PCI Express Bus)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 40.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 52-PCI 4 Zone (0xef)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 20 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 53-PCI 5 (0x88)
 Entity ID             : 49.5 (PCI Express Bus)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 40.000
 Maximum sensor range  : Unspecified
```

```
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 54-PCI 5 Zone (0xee)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 21 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 55-PCI 6 (0x89)
 Entity ID             : 49.6 (PCI Express Bus)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 40.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 56-PCI 6 Zone (0xed)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 22 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 57-PCI 7 (0x8a)
 Entity ID             : 49.7 (PCI Express Bus)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 40.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
```

```
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 58-PCI 7 Zone (0xec)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 22 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 59-PCI 8 (0x8b)
 Entity ID             : 49.8 (PCI Express Bus)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 40.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 60-PCI 8 Zone (0xeb)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 61-PCI 9 (0x8c)
 Entity ID             : 49.9 (PCI Express Bus)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 40.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
```

```
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 62-PCI 9 Zone (0xea)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 63-PCI 10 (0x8d)
 Entity ID             : 49.10 (PCI Express Bus)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 40.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 64-PCI 10 Zone (0xe9)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 65-PCI 11 (0x8e)
 Entity ID             : 49.11 (PCI Express Bus)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 40.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
```

```
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 66-PCI 11 Zone (0xe8)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 67-PCI 12 (0x8f)
 Entity ID             : 49.12 (PCI Express Bus)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 40.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 68-PCI 12 Zone (0xe7)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 69-PCI 13 (0x90)
 Entity ID             : 49.13 (PCI Express Bus)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 40.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
```

```
 Assertions Enabled    : ucr+

Sensor ID              : 70-PCI 13 Zone (0xe6)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 71-PCI 14 (0x91)
 Entity ID             : 49.14 (PCI Express Bus)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 40.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 72-PCI 14 Zone (0xe5)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 73-PCI 15 (0x92)
 Entity ID             : 49.15 (PCI Express Bus)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 40.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+
```

```
Sensor ID              : 74-PCI 15 Zone (0xe4)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 75-PCI 16 (0x93)
 Entity ID             : 49.16 (PCI Express Bus)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 40.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 76-PCI 16 Zone (0xe3)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 77-I/O Board 1 (0xe2)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 40 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 40.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+
```

```
Sensor ID              : 78-I/O Board 2 (0xe1)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 40.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 79-Battery Zone (0xe0)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 22 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 80-P/S 1 Inlet (0x4b)
 Entity ID             : 10.1 (Power Supply)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 22 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   :
 Settable Thresholds   :

Sensor ID              : 81-P/S 2 Inlet (0x4c)
 Entity ID             : 10.2 (Power Supply)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   :
 Settable Thresholds   :

Sensor ID              : 82-P/S 3 Inlet (0x4d)
 Entity ID             : 10.3 (Power Supply)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
```

```
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   :
 Settable Thresholds   :

Sensor ID              : 83-P/S 4 Inlet (0x4e)
 Entity ID             : 10.4 (Power Supply)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   :
 Settable Thresholds   :

Sensor ID              : 84-P/S 1 (0x43)
 Entity ID             : 10.1 (Power Supply)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 40 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 40.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   :
 Settable Thresholds   :

Sensor ID              : 85-P/S 2 (0x44)
 Entity ID             : 10.2 (Power Supply)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 40.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   :
 Settable Thresholds   :

Sensor ID              : 86-P/S 3 (0x45)
 Entity ID             : 10.3 (Power Supply)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 40.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   :
 Settable Thresholds   :

Sensor ID              : 87-P/S 4 (0x46)
```

```
 Entity ID             : 10.4 (Power Supply)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 40.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   :
 Settable Thresholds   :

Sensor ID              : 88-P/S 2 Zone (0xdf)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 20 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr unr
 Settable Thresholds   :
 Threshold Read Mask   : ucr unr
 Assertions Enabled    : ucr+ unr+

Sensor ID              : 89-E-Fuse (0xa5)
 Entity ID             : 20.0 (Power Module)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : 17 (+/- 0) degrees C
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 97-AHCI HD Max (0xde)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : 35.000
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 120-PCI 1 M2 (0xdd)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
```

```
 Status                 : Not Available
 Positive Hysteresis    : Unspecified
 Negative Hysteresis    : Unspecified
 Minimum sensor range   : Unspecified
 Maximum sensor range   : Unspecified
 Event Message Control  : Global Disable Only
 Readable Thresholds    : ucr
 Settable Thresholds    :
 Threshold Read Mask    : ucr
 Assertions Enabled     : ucr+

Sensor ID               : 121-PCI 1 M2 Zn (0xdc)
 Entity ID              : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading         : Disabled
 Status                 : Not Available
 Positive Hysteresis    : Unspecified
 Negative Hysteresis    : Unspecified
 Minimum sensor range   : Unspecified
 Maximum sensor range   : Unspecified
 Event Message Control  : Global Disable Only
 Readable Thresholds    :
 Settable Thresholds    :
 Assertions Enabled     :

Sensor ID               : 126-PCI 2 M2 (0xdb)
 Entity ID              : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading         : Disabled
 Status                 : Not Available
 Positive Hysteresis    : Unspecified
 Negative Hysteresis    : Unspecified
 Minimum sensor range   : Unspecified
 Maximum sensor range   : Unspecified
 Event Message Control  : Global Disable Only
 Readable Thresholds    : ucr
 Settable Thresholds    :
 Threshold Read Mask    : ucr
 Assertions Enabled     : ucr+

Sensor ID               : 127-PCI 2 M2 Zn (0xda)
 Entity ID              : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading         : Disabled
 Status                 : Not Available
 Positive Hysteresis    : Unspecified
 Negative Hysteresis    : Unspecified
 Minimum sensor range   : Unspecified
 Maximum sensor range   : Unspecified
 Event Message Control  : Global Disable Only
 Readable Thresholds    :
 Settable Thresholds    :
 Assertions Enabled     :

Sensor ID               : 138-PCI 3 M2 (0xd9)
 Entity ID              : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading         : Disabled
 Status                 : Not Available
 Positive Hysteresis    : Unspecified
 Negative Hysteresis    : Unspecified
```

```
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 139-PCI 3 M2 Zn (0xd8)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   :
 Settable Thresholds   :
 Assertions Enabled    :

Sensor ID              : 144-PCI 4 M2 (0xd7)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 145-PCI 4 M2 Zn (0xd6)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   :
 Settable Thresholds   :
 Assertions Enabled    :

Sensor ID              : 150-PCI 5 M2 (0xd5)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
```

```
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 151-PCI 5 M2 Zn (0xd4)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   :
 Settable Thresholds   :
 Assertions Enabled    :

Sensor ID              : 156-PCI 6 M2 (0xd3)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 157-PCI 6 M2 Zn (0xd2)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   :
 Settable Thresholds   :
 Assertions Enabled    :

Sensor ID              : 162-PCI 7 M2 (0xd1)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
```

```
 Assertions Enabled    : ucr+

Sensor ID              : 163-PCI 7 M2 Zn (0xd0)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   :
 Settable Thresholds   :
 Assertions Enabled    :

Sensor ID              : 168-PCI 8 M2 (0xcf)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   : ucr
 Settable Thresholds   :
 Threshold Read Mask   : ucr
 Assertions Enabled    : ucr+

Sensor ID              : 169-PCI 8 M2 Zn (0xce)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Temperature (0x01)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Global Disable Only
 Readable Thresholds   :
 Settable Thresholds   :
 Assertions Enabled    :

Sensor ID              : Fan 1 (0x3)
 Entity ID             : 29.1 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Fan
                         [Transition to OK]
 OEM                   : 1

Sensor ID              : Fan 1 DutyCycle (0x1)
 Entity ID             : 29.1 (Fan Device)
 Sensor Type (Threshold)  : Fan (0x04)
 Sensor Reading        : 37.632 (+/- 0) percent
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
```

```
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : No Events From Sensor
 Readable Thresholds   : No Thresholds
 Settable Thresholds   : No Thresholds

Sensor ID              : Fan 1 Presence (0x2)
 Entity ID             : 29.1 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Fan
                         [Device Present]
 OEM                   : 1

Sensor ID              : Fan 2 (0x6)
 Entity ID             : 29.2 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Fan
                         [Transition to OK]
 OEM                   : 2

Sensor ID              : Fan 2 DutyCycle (0x4)
 Entity ID             : 29.2 (Fan Device)
 Sensor Type (Threshold)  : Fan (0x04)
 Sensor Reading        : 37.632 (+/- 0) percent
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : No Events From Sensor
 Readable Thresholds   : No Thresholds
 Settable Thresholds   : No Thresholds

Sensor ID              : Fan 2 Presence (0x5)
 Entity ID             : 29.2 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Fan
                         [Device Present]
 OEM                   : 2

Sensor ID              : Fan 3 (0x9)
 Entity ID             : 29.3 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Fan
                         [Transition to OK]
 OEM                   : 3

Sensor ID              : Fan 3 DutyCycle (0x7)
 Entity ID             : 29.3 (Fan Device)
 Sensor Type (Threshold)  : Fan (0x04)
 Sensor Reading        : 37.632 (+/- 0) percent
 Status                : ok
 Positive Hysteresis   : Unspecified
```

```
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : No Events From Sensor
 Readable Thresholds   : No Thresholds
 Settable Thresholds   : No Thresholds

Sensor ID              : Fan 3 Presence (0x8)
 Entity ID             : 29.3 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Fan
                         [Device Present]
 OEM                   : 3

Sensor ID              : Fan 4 (0xc)
 Entity ID             : 29.4 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Fan
                         [Transition to OK]
 OEM                   : 4

Sensor ID              : Fan 4 DutyCycle (0xa)
 Entity ID             : 29.4 (Fan Device)
 Sensor Type (Threshold)  : Fan (0x04)
 Sensor Reading        : 37.632 (+/- 0) percent
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : No Events From Sensor
 Readable Thresholds   : No Thresholds
 Settable Thresholds   : No Thresholds

Sensor ID              : Fan 4 Presence (0xb)
 Entity ID             : 29.4 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Fan
                         [Device Present]
 OEM                   : 4

Sensor ID              : Fan 5 (0xf)
 Entity ID             : 29.5 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Fan
                         [Transition to OK]
 OEM                   : 5

Sensor ID              : Fan 5 DutyCycle (0xd)
 Entity ID             : 29.5 (Fan Device)
 Sensor Type (Threshold)  : Fan (0x04)
 Sensor Reading        : 37.632 (+/- 0) percent
 Status                : ok
```

```
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : No Events From Sensor
 Readable Thresholds   : No Thresholds
 Settable Thresholds   : No Thresholds

Sensor ID              : Fan 5 Presence (0xe)
 Entity ID             : 29.5 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Fan
                         [Device Present]
 OEM                   : 5

Sensor ID              : Fan 6 (0x12)
 Entity ID             : 29.6 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Fan
                         [Transition to OK]
 OEM                   : 6

Sensor ID              : Fan 6 DutyCycle (0x10)
 Entity ID             : 29.6 (Fan Device)
 Sensor Type (Threshold)  : Fan (0x04)
 Sensor Reading        : 37.632 (+/- 0) percent
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : No Events From Sensor
 Readable Thresholds   : No Thresholds
 Settable Thresholds   : No Thresholds

Sensor ID              : Fan 6 Presence (0x11)
 Entity ID             : 29.6 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Fan
                         [Device Present]
 OEM                   : 6

Sensor ID              : Fan 7 (0x15)
 Entity ID             : 29.7 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Fan
                         [Transition to OK]
 OEM                   : 7

Sensor ID              : Fan 7 DutyCycle (0x13)
 Entity ID             : 29.7 (Fan Device)
 Sensor Type (Threshold)  : Fan (0x04)
 Sensor Reading        : 37.632 (+/- 0) percent
```

```
 Status                 : ok
 Positive Hysteresis    : Unspecified
 Negative Hysteresis    : Unspecified
 Minimum sensor range   : Unspecified
 Maximum sensor range   : Unspecified
 Event Message Control : No Events From Sensor
 Readable Thresholds    : No Thresholds
 Settable Thresholds    : No Thresholds

Sensor ID               : Fan 7 Presence (0x14)
 Entity ID              : 29.7 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading         : 0h
 Event Message Control : Global Disable Only
 States Asserted        : Fan
                          [Device Present]
 OEM                    : 7

Sensor ID               : Fan 8 (0x18)
 Entity ID              : 29.8 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading         : 0h
 Event Message Control : Global Disable Only
 States Asserted        : Fan
                          [Transition to OK]
 OEM                    : 8

Sensor ID               : Fan 8 DutyCycle (0x16)
 Entity ID              : 29.8 (Fan Device)
 Sensor Type (Threshold)  : Fan (0x04)
 Sensor Reading         : 37.632 (+/- 0) percent
 Status                 : ok
 Positive Hysteresis    : Unspecified
 Negative Hysteresis    : Unspecified
 Minimum sensor range   : Unspecified
 Maximum sensor range   : Unspecified
 Event Message Control : No Events From Sensor
 Readable Thresholds    : No Thresholds
 Settable Thresholds    : No Thresholds

Sensor ID               : Fan 8 Presence (0x17)
 Entity ID              : 29.8 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading         : 0h
 Event Message Control : Global Disable Only
 States Asserted        : Fan
                          [Device Present]
 OEM                    : 8

Sensor ID               : Fan 9 (0x1b)
 Entity ID              : 29.9 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading         : 0h
 Event Message Control : Global Disable Only
 States Asserted        : Fan
                          [Transition to OK]
 OEM                    : 9

Sensor ID               : Fan 9 DutyCycle (0x19)
 Entity ID              : 29.9 (Fan Device)
 Sensor Type (Threshold)  : Fan (0x04)
```

```
 Sensor Reading        : 37.632 (+/- 0) percent
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : No Events From Sensor
 Readable Thresholds   : No Thresholds
 Settable Thresholds   : No Thresholds

Sensor ID              : Fan 9 Presence (0x1a)
 Entity ID             : 29.9 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Fan
                         [Device Present]
 OEM                   : 9

Sensor ID              : Fan 10 (0x1e)
 Entity ID             : 29.10 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Fan
                         [Transition to OK]
 OEM                   : A

Sensor ID              : Fan 10 DutyCycle (0x1c)
 Entity ID             : 29.10 (Fan Device)
 Sensor Type (Threshold)  : Fan (0x04)
 Sensor Reading        : 37.632 (+/- 0) percent
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : No Events From Sensor
 Readable Thresholds   : No Thresholds
 Settable Thresholds   : No Thresholds

Sensor ID              : Fan 10 Presence (0x1d)
 Entity ID             : 29.10 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Fan
                         [Device Present]
 OEM                   : A

Sensor ID              : Fan 11 (0x21)
 Entity ID             : 29.11 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Fan
                         [Transition to OK]
 OEM                   : B

Sensor ID              : Fan 11 DutyCycle (0x1f)
 Entity ID             : 29.11 (Fan Device)
```

```
 Sensor Type (Threshold)  : Fan (0x04)
 Sensor Reading        : 37.632 (+/- 0) percent
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : No Events From Sensor
 Readable Thresholds   : No Thresholds
 Settable Thresholds   : No Thresholds

Sensor ID              : Fan 11 Presence (0x20)
 Entity ID             : 29.11 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Fan
                         [Device Present]
 OEM                   : B

Sensor ID              : Fan 12 (0x24)
 Entity ID             : 29.12 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Fan
                         [Transition to OK]
 OEM                   : C

Sensor ID              : Fan 12 DutyCycle (0x22)
 Entity ID             : 29.12 (Fan Device)
 Sensor Type (Threshold)  : Fan (0x04)
 Sensor Reading        : 37.632 (+/- 0) percent
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : No Events From Sensor
 Readable Thresholds   : No Thresholds
 Settable Thresholds   : No Thresholds

Sensor ID              : Fan 12 Presence (0x23)
 Entity ID             : 29.12 (Fan Device)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Fan
                         [Device Present]
 OEM                   : C

Sensor ID              : Power Supply 1 (0x32)
 Entity ID             : 10.1 (Power Supply)
 Sensor Type (Discrete): Power Supply (0x08)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Power Supply
                         [Presence detected]
 OEM                   : 1

Sensor ID              : PS 1 Output (0x3a)
```

```
 Entity ID             : 10.1 (Power Supply)
 Sensor Type (Threshold)  : Power Supply (0x08)
 Sensor Reading        : 230 (+/- 0) Watts
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : No Events From Sensor
 Readable Thresholds   : No Thresholds
 Settable Thresholds   : No Thresholds

Sensor ID              : Power Supply 2 (0x33)
 Entity ID             : 10.2 (Power Supply)
 Sensor Type (Discrete): Power Supply (0x08)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 OEM                   : 2

Sensor ID              : PS 2 Output (0x3b)
 Entity ID             : 10.2 (Power Supply)
 Sensor Type (Threshold)  : Power Supply (0x08)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : No Events From Sensor
 Readable Thresholds   : No Thresholds
 Settable Thresholds   : No Thresholds

Sensor ID              : Power Supply 3 (0x34)
 Entity ID             : 10.3 (Power Supply)
 Sensor Type (Discrete): Power Supply (0x08)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 OEM                   : 3

Sensor ID              : PS 3 Output (0x3c)
 Entity ID             : 10.3 (Power Supply)
 Sensor Type (Threshold)  : Power Supply (0x08)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : No Events From Sensor
 Readable Thresholds   : No Thresholds
 Settable Thresholds   : No Thresholds

Sensor ID              : Power Supply 4 (0x35)
 Entity ID             : 10.4 (Power Supply)
 Sensor Type (Discrete): Power Supply (0x08)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 OEM                   : 4

Sensor ID              : PS 4 Output (0x3d)
 Entity ID             : 10.4 (Power Supply)
```

```
 Sensor Type (Threshold)  : Power Supply (0x08)
 Sensor Reading        : Disabled
 Status                : Not Available
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : No Events From Sensor
 Readable Thresholds   : No Thresholds
 Settable Thresholds   : No Thresholds

Sensor ID              : Power Meter (0xb6)
 Entity ID             : 7.1 (System Board)
 Sensor Type (Threshold)  : Other (0x0b)
 Sensor Reading        : 140 (+/- 0) Watts
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : No Events From Sensor
 Readable Thresholds   : No Thresholds
 Settable Thresholds   : No Thresholds
 Assertions Enabled    :

Sensor ID              : Power Supplies (0x42)
 Entity ID             : 19.1 (Power Unit)
 Sensor Type (Discrete): Power Supply (0x08)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Power Supply
                         [Redundancy Lost]
 OEM                   : 0

Sensor ID              : Fans (0x31)
 Entity ID             : 30.1 (Cooling Unit)
 Sensor Type (Discrete): Fan (0x04)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Fan
                         [Fully Redundant]
 OEM                   : 0

Sensor ID              : Memory Status (0x48)
 Entity ID             : 32.0 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Per-threshold
 States Asserted       : Memory
                         [Uncorrectable ECC]
                         [Presence Detected]
 Assertions Enabled    : Memory
                         [Uncorrectable ECC]
                         [Correctable ECC logging limit reached]
 OEM                   : 0

Sensor ID              : Megacell Status1 (0xbb)
 Entity ID             : 40.1 (Battery)
 Sensor Type (Discrete): Battery (0x29)
 Sensor Reading        : 0h
 Event Message Control : Per-threshold
```

```
 States Asserted       : Battery
                         [Presence Detected]
 Assertions Enabled    : Battery
                         [Failed]
 OEM                   : 0

Sensor ID              : Megacell Status2 (0xbc)
 Entity ID             : 40.2 (Battery)
 Sensor Type (Discrete): Battery (0x29)
 Sensor Reading        : 0h
 Event Message Control : Per-threshold
 States Asserted       : Battery
                         [Presence Detected]
 Assertions Enabled    : Battery
                         [Failed]
 OEM                   : 0

Sensor ID              : Intrusion (0xab)
 Entity ID             : 23.1 (System Chassis)
 Sensor Type (Discrete): Physical Security (0x05)
 Sensor Reading        : Disabled
 Event Message Control : Per-threshold
 Assertions Enabled    : Physical Security
                         [General Chassis intrusion]
 OEM                   : 0

Sensor ID              : CPU Utilization (0x23)
 Entity ID             : 3.1 (Processor)
 Sensor Type (Threshold)  : Processor (0x07)
 Sensor Reading        : 12 (+/- 0) unspecified
 Status                : ok
 Positive Hysteresis   : Unspecified
 Negative Hysteresis   : Unspecified
 Minimum sensor range  : Unspecified
 Maximum sensor range  : Unspecified
 Event Message Control : Per-threshold
 Readable Thresholds   : No Thresholds
 Settable Thresholds   : No Thresholds
 Assertions Enabled    :

Sensor ID              : CPU_Stat_C1 (0x12)
 Entity ID             : 3.1 (Processor)
 Sensor Type (Discrete): Processor (0x07)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Processor
                         [Presence detected]
 Assertions Enabled    : Processor
                         [IERR]
                         [Configuration Error]
                         [Uncorrectable machine check exception]
 OEM                   : 0

Sensor ID              : CPU_Stat_C2 (0x13)
 Entity ID             : 3.2 (Processor)
 Sensor Type (Discrete): Processor (0x07)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Processor
                         [Presence detected]
 Assertions Enabled    : Processor
```

```
                          [IERR]
                          [Configuration Error]
                          [Uncorrectable machine check exception]
 OEM                    : 0

Sensor ID               : CPU_Stat_C3 (0x14)
 Entity ID              : 3.3 (Processor)
 Sensor Type (Discrete): Processor (0x07)
 Sensor Reading         : 0h
 Event Message Control  : Global Disable Only
 Assertions Enabled     : Processor
                          [IERR]
                          [Configuration Error]
                          [Uncorrectable machine check exception]
 OEM                    : 0

Sensor ID               : CPU_Stat_C4 (0x15)
 Entity ID              : 3.4 (Processor)
 Sensor Type (Discrete): Processor (0x07)
 Sensor Reading         : 0h
 Event Message Control  : Global Disable Only
 Assertions Enabled     : Processor
                          [IERR]
                          [Configuration Error]
                          [Uncorrectable machine check exception]
 OEM                    : 0

Sensor ID               : Mem_Stat_C01S01 (0x0)
 Entity ID              : 32.1 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading         : 0h
 Event Message Control  : Global Disable Only
 States Asserted        : Memory
                          [Presence Detected]
 Assertions Enabled     : Memory
                          [Correctable ECC]
                          [Uncorrectable ECC]
                          [Memory Device Disabled]
                          [Configuration Error]
 OEM                    : 0

Sensor ID               : Mem_Stat_C01S02 (0x1)
 Entity ID              : 32.2 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading         : 0h
 Event Message Control  : Global Disable Only
 Assertions Enabled     : Memory
                          [Correctable ECC]
                          [Uncorrectable ECC]
                          [Memory Device Disabled]
                          [Configuration Error]
 OEM                    : 0

Sensor ID               : Mem_Stat_C01S03 (0x2)
 Entity ID              : 32.3 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading         : 0h
 Event Message Control  : Global Disable Only
 Assertions Enabled     : Memory
                          [Correctable ECC]
                          [Uncorrectable ECC]
```

```
                          [Memory Device Disabled]
                          [Configuration Error]
 OEM                    : 0

Sensor ID               : Mem_Stat_C01S04 (0x3)
 Entity ID              : 32.4 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading         : 0h
 Event Message Control  : Global Disable Only
 Assertions Enabled     : Memory
                          [Correctable ECC]
                          [Uncorrectable ECC]
                          [Memory Device Disabled]
                          [Configuration Error]
 OEM                    : 0

Sensor ID               : Mem_Stat_C01S05 (0x4)
 Entity ID              : 32.5 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading         : 0h
 Event Message Control  : Global Disable Only
 Assertions Enabled     : Memory
                          [Correctable ECC]
                          [Uncorrectable ECC]
                          [Memory Device Disabled]
                          [Configuration Error]
 OEM                    : 0

Sensor ID               : Mem_Stat_C01S06 (0x5)
 Entity ID              : 32.6 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading         : 0h
 Event Message Control  : Global Disable Only
 Assertions Enabled     : Memory
                          [Correctable ECC]
                          [Uncorrectable ECC]
                          [Memory Device Disabled]
                          [Configuration Error]
 OEM                    : 0

Sensor ID               : Mem_Stat_C01S07 (0x6)
 Entity ID              : 32.7 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading         : 0h
 Event Message Control  : Global Disable Only
 Assertions Enabled     : Memory
                          [Correctable ECC]
                          [Uncorrectable ECC]
                          [Memory Device Disabled]
                          [Configuration Error]
 OEM                    : 0

Sensor ID               : Mem_Stat_C01S08 (0x7)
 Entity ID              : 32.8 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading         : 0h
 Event Message Control  : Global Disable Only
 Assertions Enabled     : Memory
                          [Correctable ECC]
                          [Uncorrectable ECC]
                          [Memory Device Disabled]
```

```
                          [Configuration Error]
 OEM                    : 0

Sensor ID               : Mem_Stat_C01S09 (0x8)
 Entity ID              : 32.9 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading         : 0h
 Event Message Control  : Global Disable Only
 Assertions Enabled     : Memory
                          [Correctable ECC]
                          [Uncorrectable ECC]
                          [Memory Device Disabled]
                          [Configuration Error]
 OEM                    : 0

Sensor ID               : Mem_Stat_C01S10 (0x9)
 Entity ID              : 32.10 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading         : 0h
 Event Message Control  : Global Disable Only
 Assertions Enabled     : Memory
                          [Correctable ECC]
                          [Uncorrectable ECC]
                          [Memory Device Disabled]
                          [Configuration Error]
 OEM                    : 0

Sensor ID               : Mem_Stat_C01S11 (0xa)
 Entity ID              : 32.11 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading         : 0h
 Event Message Control  : Global Disable Only
 Assertions Enabled     : Memory
                          [Correctable ECC]
                          [Uncorrectable ECC]
                          [Memory Device Disabled]
                          [Configuration Error]
 OEM                    : 0

Sensor ID               : Mem_Stat_C01S12 (0xb)
 Entity ID              : 32.12 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading         : 0h
 Event Message Control  : Global Disable Only
 Assertions Enabled     : Memory
                          [Correctable ECC]
                          [Uncorrectable ECC]
                          [Memory Device Disabled]
                          [Configuration Error]
 OEM                    : 0

Sensor ID               : Mem_Stat_C02S01 (0x10)
 Entity ID              : 32.17 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading         : 0h
 Event Message Control  : Global Disable Only
 Assertions Enabled     : Memory
                          [Correctable ECC]
                          [Uncorrectable ECC]
                          [Memory Device Disabled]
                          [Configuration Error]
```

```
 OEM                   : 0

Sensor ID              : Mem_Stat_C02S02 (0x11)
 Entity ID             : 32.18 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 States Asserted       : Memory
                         [Presence Detected]
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C02S03 (0x12)
 Entity ID             : 32.19 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C02S04 (0x13)
 Entity ID             : 32.20 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C02S05 (0x14)
 Entity ID             : 32.21 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C02S06 (0x15)
 Entity ID             : 32.22 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
```

```
                          [Configuration Error]
 OEM                    : 0

Sensor ID               : Mem_Stat_C02S07 (0x16)
 Entity ID              : 32.23 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading         : 0h
 Event Message Control  : Global Disable Only
 Assertions Enabled     : Memory
                          [Correctable ECC]
                          [Uncorrectable ECC]
                          [Memory Device Disabled]
                          [Configuration Error]
 OEM                    : 0

Sensor ID               : Mem_Stat_C02S08 (0x17)
 Entity ID              : 32.24 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading         : 0h
 Event Message Control  : Global Disable Only
 Assertions Enabled     : Memory
                          [Correctable ECC]
                          [Uncorrectable ECC]
                          [Memory Device Disabled]
                          [Configuration Error]
 OEM                    : 0

Sensor ID               : Mem_Stat_C02S09 (0x18)
 Entity ID              : 32.25 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading         : 0h
 Event Message Control  : Global Disable Only
 Assertions Enabled     : Memory
                          [Correctable ECC]
                          [Uncorrectable ECC]
                          [Memory Device Disabled]
                          [Configuration Error]
 OEM                    : 0

Sensor ID               : Mem_Stat_C02S10 (0x19)
 Entity ID              : 32.26 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading         : 0h
 Event Message Control  : Global Disable Only
 Assertions Enabled     : Memory
                          [Correctable ECC]
                          [Uncorrectable ECC]
                          [Memory Device Disabled]
                          [Configuration Error]
 OEM                    : 0

Sensor ID               : Mem_Stat_C02S11 (0x1a)
 Entity ID              : 32.27 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading         : 0h
 Event Message Control  : Global Disable Only
 Assertions Enabled     : Memory
                          [Correctable ECC]
                          [Uncorrectable ECC]
                          [Memory Device Disabled]
                          [Configuration Error]
```

```
 OEM                   : 0

Sensor ID              : Mem_Stat_C02S12 (0x1b)
 Entity ID             : 32.28 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C03S01 (0x20)
 Entity ID             : 32.33 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C03S02 (0x21)
 Entity ID             : 32.34 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C03S03 (0x22)
 Entity ID             : 32.35 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C03S04 (0x23)
 Entity ID             : 32.36 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0
```

```
Sensor ID              : Mem_Stat_C03S05 (0x24)
 Entity ID             : 32.37 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C03S06 (0x25)
 Entity ID             : 32.38 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C03S07 (0x26)
 Entity ID             : 32.39 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C03S08 (0x27)
 Entity ID             : 32.40 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C03S09 (0x28)
 Entity ID             : 32.41 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0
```

```
Sensor ID              : Mem_Stat_C03S10 (0x29)
 Entity ID             : 32.42 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C03S11 (0x2a)
 Entity ID             : 32.43 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C03S12 (0x2b)
 Entity ID             : 32.44 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C04S01 (0x30)
 Entity ID             : 32.49 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C04S02 (0x31)
 Entity ID             : 32.50 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C04S03 (0x32)
```

```
 Entity ID             : 32.51 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C04S04 (0x33)
 Entity ID             : 32.52 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C04S05 (0x34)
 Entity ID             : 32.53 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C04S06 (0x35)
 Entity ID             : 32.54 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C04S07 (0x36)
 Entity ID             : 32.55 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C04S08 (0x37)
 Entity ID             : 32.56 (Memory Device)
```

```
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C04S09 (0x38)
 Entity ID             : 32.57 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C04S10 (0x39)
 Entity ID             : 32.58 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C04S11 (0x3a)
 Entity ID             : 32.59 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : Mem_Stat_C04S12 (0x3b)
 Entity ID             : 32.60 (Memory Device)
 Sensor Type (Discrete): Memory (0x0c)
 Sensor Reading        : 0h
 Event Message Control : Global Disable Only
 Assertions Enabled    : Memory
                         [Correctable ECC]
                         [Uncorrectable ECC]
                         [Memory Device Disabled]
                         [Configuration Error]
 OEM                   : 0

Sensor ID              : PCI-E Bus Error (0xc4)
Entity ID              : 49.0 (PCI Express Bus)
Sensor Type            : Critical Interrupt (0x13)
```

```
Sensor ID              : OS_NMI (0xc5)
Entity ID              : 35.0 (Operating System)
Sensor Type            : Critical Interrupt (0x13)

Sensor ID              : CPU_ThermTrip (0xc3)
Entity ID              : 3.0 (Processor)
Sensor Type            : Processor (0x07)

Sensor ID              : SYS_ThermTrip (0xb7)
Entity ID              : 7.1 (System Board)
Sensor Type            : Module / Board (0x15)

Sensor ID              : ACPI_Stat (0xaa)
Entity ID              : 7.1 (System Board)
Sensor Type            : System ACPI Power State (0x22)

Device ID              : BMC CONTROLLER
Entity ID              : 23.1 (System Chassis)
Device Access Address  : 20h
Logical FRU Device     : EEh
Channel Number         : 0h
LUN.Bus                : 0h.0h
Device Type.Modifier   : 10h.0h (IPMI FRU Inventory)
OEM                    : 00h

Device ID              : MB BIOS
Entity ID              : 34.1 (BIOS)
Device Access Address  : 20h
Logical FRU Device     : EFh
Channel Number         : 0h
LUN.Bus                : 0h.0h
Device Type.Modifier   : 10h.0h (IPMI FRU Inventory)
OEM                    : 00h

Device ID              : CPU 1
Entity ID              : 3.1 (Processor)
Device Access Address  : 20h
Logical FRU Device     : 10h
Channel Number         : 0h
LUN.Bus                : 0h.0h
Device Type.Modifier   : 10h.0h (IPMI FRU Inventory)
OEM                    : 00h

Device ID              : CPU 2
Entity ID              : 3.2 (Processor)
Device Access Address  : 20h
Logical FRU Device     : 11h
Channel Number         : 0h
LUN.Bus                : 0h.0h
Device Type.Modifier   : 10h.0h (IPMI FRU Inventory)
OEM                    : 00h

Device ID              : PCI-E Slot 7
Entity ID              : 49.7 (PCI Express Bus)
Device Access Address  : 20h
Logical FRU Device     : 02h
Channel Number         : 0h
LUN.Bus                : 0h.0h
Device Type.Modifier   : 10h.0h (IPMI FRU Inventory)
OEM                    : 00h
```

```
Device ID              : Ethernet Adptr
Entity ID              : 44.1 (I/O Module)
Device Access Address  : 20h
Logical FRU Device     : 03h
Channel Number         : 0h
LUN.Bus                : 0h.0h
Device Type.Modifier   : 10h.0h (IPMI FRU Inventory)
OEM                    : 00h

Device ID              : Embedded 0
Entity ID              : 0.0 (Unspecified)
Device Access Address  : 20h
Logical FRU Device     : 04h
Channel Number         : 0h
LUN.Bus                : 0h.0h
Device Type.Modifier   : 10h.0h (IPMI FRU Inventory)
OEM                    : 00h

Device ID              : Embedded 0
Entity ID              : 0.0 (Unspecified)
Device Access Address  : 20h
Logical FRU Device     : 05h
Channel Number         : 0h
LUN.Bus                : 0h.0h
Device Type.Modifier   : 10h.0h (IPMI FRU Inventory)
OEM                    : 00h

Device ID              : PCI-E Slot 1
Entity ID              : 49.1 (PCI Express Bus)
Device Access Address  : 20h
Logical FRU Device     : 06h
Channel Number         : 0h
LUN.Bus                : 0h.0h
Device Type.Modifier   : 10h.0h (IPMI FRU Inventory)
OEM                    : 00h

Device ID              : PCI-E Slot 2
Entity ID              : 49.2 (PCI Express Bus)
Device Access Address  : 20h
Logical FRU Device     : 07h
Channel Number         : 0h
LUN.Bus                : 0h.0h
Device Type.Modifier   : 10h.0h (IPMI FRU Inventory)
OEM                    : 00h

Device ID              : PCI-E Slot 3
Entity ID              : 49.3 (PCI Express Bus)
Device Access Address  : 20h
Logical FRU Device     : 08h
Channel Number         : 0h
LUN.Bus                : 0h.0h
Device Type.Modifier   : 10h.0h (IPMI FRU Inventory)
OEM                    : 00h

Device ID              : PCI-E Slot 4
Entity ID              : 49.4 (PCI Express Bus)
Device Access Address  : 20h
Logical FRU Device     : 09h
Channel Number         : 0h
LUN.Bus                : 0h.0h
```

```
Device Type.Modifier   : 10h.0h (IPMI FRU Inventory)
OEM                    : 00h

Device ID              : PCI-E Slot 5
Entity ID              : 49.5 (PCI Express Bus)
Device Access Address  : 20h
Logical FRU Device     : 0Ah
Channel Number         : 0h
LUN.Bus                : 0h.0h
Device Type.Modifier   : 10h.0h (IPMI FRU Inventory)
OEM                    : 00h

Device ID              : PCI-E Slot 6
Entity ID              : 49.6 (PCI Express Bus)
Device Access Address  : 20h
Logical FRU Device     : 0Bh
Channel Number         : 0h
LUN.Bus                : 0h.0h
Device Type.Modifier   : 10h.0h (IPMI FRU Inventory)
OEM                    : 00h

Device ID              : SAS Ctlr
Entity ID              : 51.1 (SATA/SAS Bus)
Device Access Address  : 20h
Logical FRU Device     : 0Ch
Channel Number         : 0h
LUN.Bus                : 0h.0h
Device Type.Modifier   : 10h.0h (IPMI FRU Inventory)
OEM                    : 00h

Device ID              : Video
Entity ID              : 5.1 (Peripheral Bay)
Device Access Address  : 20h
Logical FRU Device     : 0Dh
Channel Number         : 0h
LUN.Bus                : 0h.0h
Device Type.Modifier   : 10h.0h (IPMI FRU Inventory)
OEM                    : 00h

Device ID              : ChasMC
Entity ID              : 23.1 (System Chassis)
Device Slave Address   : 44h
Channel Number         : 0h
ACPI System P/S Notif  : Not Required
ACPI Device P/S Notif  : Not Required
Controller Presence    : Static
Logs Init Agent Errors : No
Event Message Gen      : Enable
Device Capabilities
 Chassis Device        : No
 Bridge                : No
 IPMB Event Generator  : Yes
 IPMB Event Receiver   : No
 FRU Inventory Device  : No
 SEL Device            : No
 SDR Repository        : No
 Sensor Device         : Yes

Device ID              : BMC
Entity ID              : 6.1 (System Management Module)
Device Slave Address   : 20h
```

```
Channel Number          : 0h
ACPI System P/S Notif   : Not Required
ACPI Device P/S Notif   : Not Required
Controller Presence     : Dynamic
Logs Init Agent Errors  : No
Event Message Gen       : Enable
Device Capabilities
 Chassis Device         : Yes
 Bridge                 : No
 IPMB Event Generator   : Yes
 IPMB Event Receiver    : Yes
 FRU Inventory Device   : Yes
 SEL Device             : Yes
 SDR Repository         : Yes
 Sensor Device          : Yes
```

DCTS (DCMI Conformance Test Suite)

The DCMI Conformance Test Suite (DCTS) provides a baseline set of test for verifying compliance with the Data Center Management Interface (DCMI) specification (in both version 1.1 and 1.5)

DCTS presents a simple menu driven user interface. Each test scenario verifies a logical unit of functionality and reports a pass, a fail or a skipped.

Based on the method chosen for communication with the target system, the following two modes of testing are supported.

- In-Band Testing
  - Using KCS Interface
  - The test tool resides on the Server Platform (UUT)
- Out-of-Band Testing
  - Using Ethernet LAN-based connectivity through IPMI/RMCP+ protocol
  - The test tool can reside on any remote Windows-based PC

Steps to run the DCTS over LAN Interface

Download the corresponding DCTS executable from Intel website. Go to **http://www.intel.com/content/www/us/en/data-center/dcmi/binary-download.html** to download.

Userconf.cfg

The basic network and session configuration for the test environment is extracted from the file `UserConf.cfg` file available in the same folder as the executable file.

- Enter the correct Cipher Suite number
- Enter the file name for

  `log-file-name`
- Enter the correct target IP (IP address of iLO)
- Enter the Username (iLO User name to authenticate)

DCMIConformance.exe

WIN32 application. It serves as the:

- Presentation layer for user Input/output
- Send, receive, and process the commands
- Report results

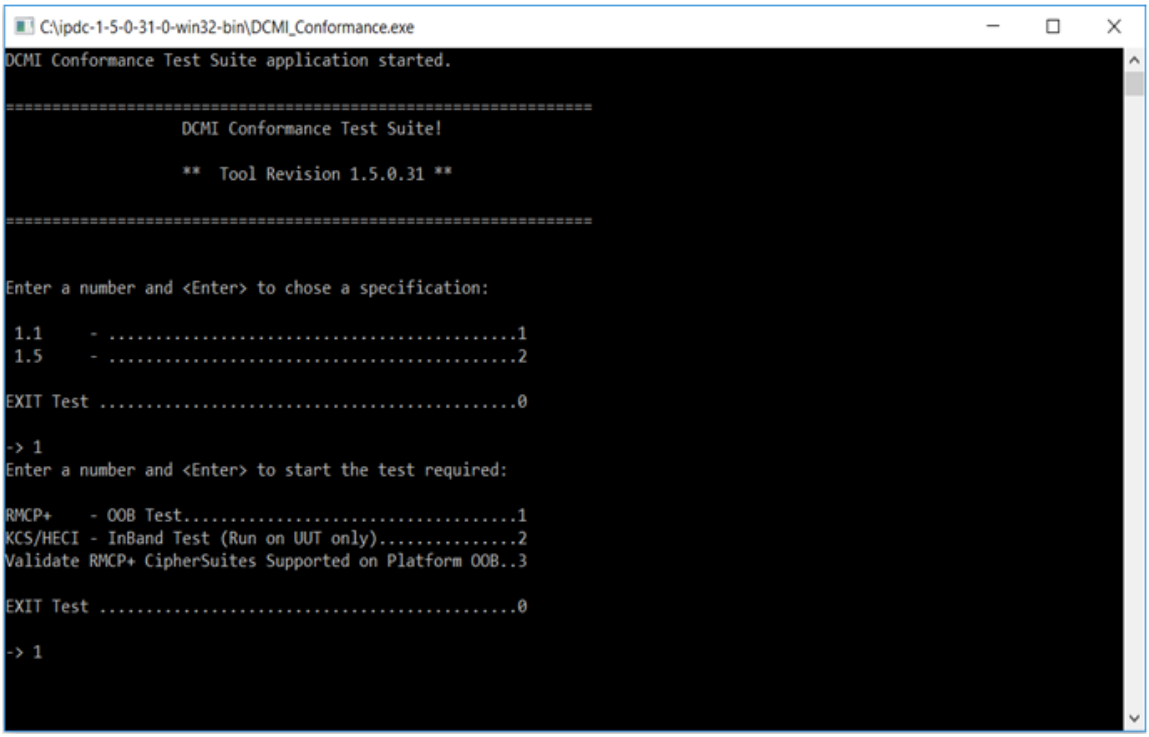


Figure 9: DCMIConformance

Known issues or limitations

Request sent with responder address set to 0

Symptom

Packets with an incorrect responder address are dropped.

Cause

DCMI Conformance Test Suite (DCTS) tool revision v1.5 requests the DCMI Get Capabilities with the responder address set to `0x00`. The BMC expects the responder address to be set to `0x20`.

Action

When you run the DCTS tool, set parameter 1 to DCTS mode by using the OEM `Set IPMI configuration parameter` command.

For example:

`ipmitool -H <iLO IP address> -u <username> -p <password> -I lanplus raw 0x30 0xbb 1 1`

More information

DCTS (DCMI Conformance Test Suite)

DCMI Conformance Test Summary (DCMI v1.1 rev 2)

Table 164: DCMI Conformance Test Summary (DCMI v1.1 rev 2)

Tests 1, Basic Discovery		
Test Case 1.1	Supported DCMI Platform Capabilities	PASS
Test Case 1.2	Manageability Access Attributes	PASS
Test Case 1.3	Session Less Capabilities	PASS
Test Case 1.4	Minimum Platform Attributes	PASS
Test Case 1.5	Optional Platform Attributes	PASS
Test Case 1.6	Enhanced System Power Statistics Attributes	PASS
Tests 2, Basics Test		
Test Case 2.1	Device IP from Management Controller	PASS
Test Case 2.2	System GUID from Management Controller	PASS
Test Case 2.3	Asset Tag from UUT	PASS
Tests 3, Cipher Tests		
Test Case 3.1	Checking Supported CipherSuites	PASS
Tests 4, SEL Tests		
Test Case 4.1	Get SEL Info	PASS
Test Case 4.2	Reserve SEL	PASS
Test Case 4.3	Get SEL Entry, with Reservation ID	PASS
Test Case 4.4	Get First SEL Entry after Reservation	PASS
Test Case 4.5	Get Last SEL ENtry and Verify	PASS
Test Case 4.6	Clear SEL	PASS
Test Case 4.7	Verify SEL Clear Action	PASS
Tests 5, DCMI Sensor Tests		

Table Continued

Test Case 5.1	Sensors Entity: Inlet (0x40)	PASS
Test Case 5.2	Sensors Entity: CPU (0x41)	PASS
Test Case 5.3	Sensors Entity: Baseboard (0x42)	PASS
Test Case 5.4	Sensor Entity ID: 0x40, Type: 0x1, Instance: 1	PASS
Test Case 5.5	Sensor Entity ID: 0x41, Type: 0x1, Instance: 1	PASS
Test Case 5.6	Sensor Entity ID: 0x41, Type: 0x1, Instance: 2	PASS
Test Case 5.7	Sensor Entity ID: 0x42, Type: 0x1, Instance: 1	PASS
Test Case 5.8	Sensor Entity ID: 0x42, Type: 0x1, Instance: 2	PASS
Test Case 5.9	Sensor Entity ID: 0x42, Type: 0x1, Instance: 3	PASS
Test Case 5.10	Sensor Entity ID: 0x42, Type: 0x1, Instance: 4	PASS
Test Case 5.11	Sensor Entity ID: 0x42, Type: 0x1, Instance: 5	PASS
Test Case 5.12	Sensor Entity ID: 0x42, Type: 0x1, Instance: 6	PASS
Tests 6, DCMI SDR Tests		
Test Case 6.1	Checking SDR Repository Info	PASS
Tests 7, Chassis Commands		
Test Case 7.1	Issue Get Chassis Capabilities Command	PASS
Test Case 7.2	Checking Chassis Status for Initial Power State	PASS
Test Case 7.3	Checking Chassis Identify Command supported	PASS

Table Continued

Test Case 7.4	Check ACPI Power State	PASS
Test Case 7.5	Check Turn System Off	PASS
Test Case 7.6	Check Turn System On	PASS
Test Case 7.7	Check Reboot System	PASS
Tests 8, Verify support for the LAN Configuration Commands		
Test Case 8.1	VLAN Support Test	PASS
Test Case 8.2	VLAN Priority Test	PASS
Test Case 8.3	VLAN RMCPP+ Entry Support	PASS
Test Case 8.4	VLAN RMCPP+ Entries	PASS
Test Case 8.5	VLAN RMCPP+ Privilege level	PASS
Tests 9, DCMI SOL Tests		
Test Case 9.1	Checking Serial Over Lan Configuration	PASS
Test Case 9.2	Checking SOL Channel Auth. Capabilities	PASS
Test Case 9.3	Checking SOL Payload Activation - Type SOL	PASS
Test Case 9.4	Checking SOL Payload Instance Info - Type SOL	PASS
Tests 10, DCMI TMode Tests		
Test Case 10.1	TMODE Support Test	SKIPPED
Tests 11, DCMI Discovery for Power Management Controller Info		
Test Case 11.1	DCMI Get Power Reading	PASS
Test Case 11.2	DCMI Get Power Limit	PASS

Table Continued

Tests 12, LAN Configuration Check Tests		
Test Case 12.1	Lan Configuration Gratutious Arp Check	PASS
Test Case 12.2	Lan Configuration Arp Control Check	PASS
Test Case 12.3	Lan Configuration IP Source Check	PASS
Test Case 12.4	Lan Configuration Access Mode Check	PASS
Test Case 12.5	User Access Check	PASS
Test Case 12.6	User Payload Access Check	PASS
Test Case 12.7	Multi Session Test	PASS
Test Case 12.8	Get MC ID String Test	PASS

NOTE: DCMI v1.5 compliance is currently NOT supported. It expects the following additional test cases to be passed.

Tests 13, Configuration Parameters Test		
Test Case 13.1	Get Active DHCP	FAIL
Test Case 13.2	Get Discovery Configuration	FAIL
Test Case 13.3	Get DHCP Timing 1	FAIL
Test Case 13.4	Get DHCP Timing 2	FAIL
Test Case 13.5	Get DHCP Timing 3	FAIL
Tests 14, Thermal Management Tests		
Test Case 14.1	Temperature reading for sensor: Inlet (0x40)	PASS
Test Case 14.2	Temperature reading for sensor: CPU (0x41)	PASS
Test Case 14.3	Temperature reading for sensor: Baseboard (0x42)	PASS

Standard sensor list

Sensor LUN 0

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Alt. Old Name	Old Name	New Name
1	0x01 Threshold	0x04 Fan	0x1D Cooling	1		FAN 1	Fan_Speed_01
2	0x01 Threshold	0x01 Temperature	0x03 Processor	1		XX-CPU 1	CPU_Temp_C1
3	0x01 Threshold	0x01 Temperature	0x03 Processor	2		XX-CPU 2	CPU_Temp_C2
4	0x01 Threshold	0x01 Temperature	0x03 Processor	3		XX-CPU 3	CPU_Temp_C3
5	0x01 Threshold	0x01 Temperature	0x03 Processor	4		XX-CPU 4	CPU_Temp_C4
6	0x01 Threshold	0x01 Temperature	0x03 Processor	5		XX-CPU 5	CPU_Temp_C5
7	0x01 Threshold	0x01 Temperature	0x03 Processor	6		XX-CPU 6	CPU_Temp_C6
8	0x01 Threshold	0x01 Temperature	0x03 Processor	7		XX-CPU 7	CPU_Temp_C7
9	0x01 Threshold	0x01 Temperature	0x03 Processor	8		XX-CPU 8	CPU_Temp_C8
10	0x01 Threshold	0x01 Temperature	0x14 Power Module	1		XX-VR P1	CPUVR_Temp_C1
11	0x01 Threshold	0x01 Temperature	0x14 Power Module	2		XX-VR P2	CPUVR_Temp_C2
12	0x01 Threshold	0x01 Temperature	0x14 Power Module	3		XX-VR P3	CPUVR_Temp_C3
13	0x01 Threshold	0x01 Temperature	0x14 Power Module	4		XX-VR P4	CPUVR_Temp_C4
14	0x01 Threshold	0x01 Temperature	0x14 Power Module	5		XX-VR P5	CPUVR_Temp_C5
15	0x01 Threshold	0x01 Temperature	0x14 Power Module	6		XX-VR P6	CPUVR_Temp_C6
16	0x01 Threshold	0x01 Temperature	0x14 Power Module	7		XX-VR P7	CPUVR_Temp_C7
17	0x01 Threshold	0x01 Temperature	0x14 Power Module	8		XX-VR P8	CPUVR_Temp_C8

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Alt. Old Name	Old Name	New Name
18	0x6F Sensor-specific	0x07 Processor	0x03 Processor	1		CPU-1	CPU_Stat_C1
19	0x6F Sensor-specific	0x07 Processor	0x03 Processor	2		CPU-2	CPU_Stat_C2
20	0x6F Sensor-specific	0x07 Processor	0x03 Processor	3		CPU-3	CPU_Stat_C3
21	0x6F Sensor-specific	0x07 Processor	0x03 Processor	4		CPU-4	CPU_Stat_C4
22	0x6F Sensor-specific	0x07 Processor	0x03 Processor	5		CPU-5	CPU_Stat_C5
23	0x6F Sensor-specific	0x07 Processor	0x03 Processor	6		CPU-6	CPU_Stat_C6
24	0x6F Sensor-specific	0x07 Processor	0x03 Processor	7		CPU-7	CPU_Stat_C7
25	0x6F Sensor-specific	0x07 Processor	0x03 Processor	8		CPU-8	CPU_Stat_C8
26	0x01 Threshold	0x02 Voltage	0x03 Processor	1		n/a	CPU_Volt_C1
27	0x01 Threshold	0x02 Voltage	0x03 Processor	2		n/a	CPU_Volt_C2
28	0x01 Threshold	0x02 Voltage	0x03 Processor	3		n/a	CPU_Volt_C3
29	0x01 Threshold	0x02 Voltage	0x03 Processor	4		n/a	CPU_Volt_C4
30	0x01 Threshold	0x02 Voltage	0x03 Processor	5		n/a	CPU_Volt_C5
31	0x01 Threshold	0x02 Voltage	0x03 Processor	6		n/a	CPU_Volt_C6
32	0x01 Threshold	0x02 Voltage	0x03 Processor	7		n/a	CPU_Volt_C7
33	0x01 Threshold	0x02 Voltage	0x03 Processor	8		n/a	CPU_Volt_C8
34[1]	0x01 Threshold	0x0B Other	0x03 Processor	1		n/a	CPU_Watt

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Alt. Old Name	Old Name	New Name
35[1]	0x01 Threshold	0x0B Other	0x03 Processor	1		CPU Utilization	CPU_Util
36[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	11		XX-P1 DIMM1-6	Mem_Temp_C1G1
37[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	12		XX-P1 DIMM7-12	Mem_Temp_C1G2
38[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	13		XX-P2 DIMM1-6	Mem_Temp_C2G1
39[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	14		XX-P2 DIMM7-12	Mem_Temp_C2G2
40[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	15		XX-P3 DIMM1-6	Mem_Temp_C2G1
41[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	16		XX-P3 DIMM7-12	Mem_Temp_C2G2
42[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	17		XX-P4 DIMM1-6	Mem_Temp_C3G1
43[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	18		XX-P4 DIMM7-12	Mem_Temp_C3G2
44[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	19		XX-P5 DIMM1-6	Mem_Temp_C4G1
45[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	20		XX-P5 DIMM7-12	Mem_Temp_C4G2
46[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	21		XX-P6 DIMM1-6	Mem_Temp_C5G1
47[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	22		XX-P6 DIMM7-12	Mem_Temp_C5G2
48[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	23		XX-P4 DIMM1-6	Mem_Temp_C6G1
49[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	24		XX-P4 DIMM7-12	Mem_Temp_C6G2
50[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	25		XX-P5 DIMM1-6	Mem_Temp_C7G1
51[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	26		XX-P5 DIMM7-12	Mem_Temp_C7G2
52[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	27		XX-P6 DIMM1-6	Mem_Temp_C8G1
53[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	28		XX-P6 DIMM7-12	Mem_Temp_C8G2
54[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	11		XX-P1 NVDIMM1-6	NVMem_Temp_C1G1

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Alt. Old Name	Old Name	New Name
55[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	12		XX-P1 NVDIMM7-12	NVMem_Temp_C1 G2
56[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	13		XX-P2 NVDIMM1-6	NVMem_Temp_C2 G1
57[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	14		XX-P2 NVDIMM7-12	NVMem_Temp_C2 G2
58[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	15		XX-P2 NVDIMM1-6	NVMem_Temp_C2 G1
59[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	16		XX-P2 NVDIMM7-12	NVMem_Temp_C2 G2
60[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	17		XX-P3 NVDIMM1-6	NVMem_Temp_C3 G1
61[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	18		XX-P3 NVDIMM7-12	NVMem_Temp_C3 G2
62[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	19		XX-P4 NVDIMM1-6	NVMem_Temp_C4 G1
63[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	20		XX-P4 NVDIMM7-12	NVMem_Temp_C4 G2
64[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	21		XX-P5 NVDIMM1-6	NVMem_Temp_C5 G1
65[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	22		XX-P5 NVDIMM7-12	NVMem_Temp_C5 G2
66[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	23		XX-P6 NVDIMM1-6	NVMem_Temp_C6 G1
67[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	24		XX-P6 NVDIMM7-12	NVMem_Temp_C6 G2
68[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	25		XX-P7 NVDIMM1-6	NVMem_Temp_C7 G1
69[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	26		XX-P7 NVDIMM7-12	NVMem_Temp_C7 G2
70[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	27		XX-P8 NVDIMM1-6	NVMem_Temp_C8 G1
71[1]	0x01 Threshold	0x01 Temperature	0x20 Memory	28		XX-P8 NVDIMM7-12	NVMem_Temp_C8 G2
72[1]	0x6F Sensor-specific	0x0C Memory	0x20 Memory	0		Memory Status	Mem_Stat
73[1]	0x6F Sensor-specific	0x0C Memory	0x20 Memory	1		n/a	Mem_Stat_C1

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Alt. Old Name	Old Name	New Name
74[1]	0x6F Sensor-specific	0x0C Memory	0x20 Memory	2		n/a	Mem_Stat_C2
75[1]	0x6F Sensor-specific	0x0C Memory	0x20 Memory	3		n/a	Mem_Stat_C3
76[1]	0x6F Sensor-specific	0x0C Memory	0x20 Memory	4		n/a	Mem_Stat_C4
77[1]	0x6F Sensor-specific	0x0C Memory	0x20 Memory	5		n/a	Mem_Stat_C5
78[1]	0x6F Sensor-specific	0x0C Memory	0x20 Memory	6		n/a	Mem_Stat_C6
79[1]	0x6F Sensor-specific	0x0C Memory	0x20 Memory	7		n/a	Mem_Stat_C7
80[1]	0x6F Sensor-specific	0x0C Memory	0x20 Memory	8		n/a	Mem_Stat_C8
81[1]	0x01 Threshold	0x02 Voltage	0x14 Power Module	21		n/a	MemVR_Volt_C1G1
82[1]	0x01 Threshold	0x02 Voltage	0x14 Power Module	22		n/a	MemVR_Volt_C1G2
83[1]	0x01 Threshold	0x02 Voltage	0x14 Power Module	23		n/a	MemVR_Volt_C2G1
84[1]	0x01 Threshold	0x02 Voltage	0x14 Power Module	24		n/a	MemVR_Volt_C2G2
85[1]	0x01 Threshold	0x02 Voltage	0x14 Power Module	25		n/a	MemVR_Volt_C3G1
86[1]	0x01 Threshold	0x02 Voltage	0x14 Power Module	26		n/a	MemVR_Volt_C3G2
87[1]	0x01 Threshold	0x02 Voltage	0x14 Power Module	27		n/a	MemVR_Volt_C4G1
88[1]	0x01 Threshold	0x02 Voltage	0x14 Power Module	28		n/a	MemVR_Volt_C4G2
89[1]	0x01 Threshold	0x02 Voltage	0x14 Power Module	28		n/a	MemVR_Volt_C5G1
90[1]	0x01 Threshold	0x02 Voltage	0x14 Power Module	30		n/a	MemVR_Volt_C5G2

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Alt. Old Name	Old Name	New Name
91[1]	0x01 Threshold	0x02 Voltage	0x14 Power Module	31		n/a	MemVR_Volt_C6G1
92[1]	0x01 Threshold	0x02 Voltage	0x14 Power Module	32		n/a	MemVR_Volt_C6G2
93[1]	0x01 Threshold	0x02 Voltage	0x14 Power Module	33		n/a	MemVR_Volt_C7G1
94[1]	0x01 Threshold	0x02 Voltage	0x14 Power Module	34		n/a	MemVR_Volt_C7G2
95[1]	0x01 Threshold	0x02 Voltage	0x14 Power Module	35		n/a	MemVR_Volt_C8G1
96[1]	0x01 Threshold	0x02 Voltage	0x14 Power Module	36		n/a	MemVR_Volt_C8G2
97[1]	0x01 Threshold	0x01 Temperature	0x14 Power Module	21		XX-VR P1 Mem 1	MemVR_Temp_C01G1
98[1]	0x01 Threshold	0x01 Temperature	0x14 Power Module	22		XX-VR P1 Mem 2	MemVR_Temp_C01G2
99[1]	0x01 Threshold	0x01 Temperature	0x14 Power Module	23		XX-VR P2 Mem 1	MemVR_Temp_C02G1
100[1]	0x01 Threshold	0x01 Temperature	0x14 Power Module	24		XX-VR P2 Mem 2	MemVR_Temp_C02G2
101[1]	0x01 Threshold	0x01 Temperature	0x14 Power Module	25		XX-VR P3 Mem 1	MemVR_Temp_C03G1
102[1]	0x01 Threshold	0x01 Temperature	0x14 Power Module	26		XX-VR P3 Mem 2	MemVR_Temp_C03G2
103[1]	0x01 Threshold	0x01 Temperature	0x14 Power Module	27		XX-VR P4 Mem 1	MemVR_Temp_C04G1
104[1]	0x01 Threshold	0x01 Temperature	0x14 Power Module	28		XX-VR P4 Mem 2	MemVR_Temp_C04G2
105[1]	0x01 Threshold	0x01 Temperature	0x14 Power Module	29		XX-VR P5 Mem 1	MemVR_Temp_C05G1
106[1]	0x01 Threshold	0x01 Temperature	0x14 Power Module	30		XX-VR P5 Mem 2	MemVR_Temp_C05G2
107[1]	0x01 Threshold	0x01 Temperature	0x14 Power Module	31		XX-VR P6 Mem 1	MemVR_Temp_C06G1
108[1]	0x01 Threshold	0x01 Temperature	0x14 Power Module	32		XX-VR P6 Mem 2	MemVR_Temp_C06G2
109[1]	0x01 Threshold	0x01 Temperature	0x14 Power Module	33		XX-VR P7 Mem 1	MemVR_Temp_C07G1
110[1]	0x01 Threshold	0x01 Temperature	0x14 Power Module	34		XX-VR P7 Mem 2	MemVR_Temp_C07G2

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Alt. Old Name	Old Name	New Name
111[1]	0x01 Threshold	0x01 Temperature	0x14 Power Module	35		XX-VR P8 Mem 1	MemVR_Temp_C08 G1
112[1]	0x01 Threshold	0x01 Temperature	0x14 Power Module	36		XX-VR P8 Mem 2	MemVR_Temp_C08 G2
113[1]	0x01 Threshold	0x0B Other	0x20 Memory	0		n/a	Mem_Watt
114	0x6F Sensor-specific	0x21 Slot	0x31 PCIe	1		n/a	Slot_Stat_01
115	0x6F Sensor-specific	0x21 Slot	0x31 PCIe	2		n/a	Slot_Stat_02
116	0x6F Sensor-specific	0x21 Slot	0x31 PCIe	3		n/a	Slot_Stat_03
117	0x6F Sensor-specific	0x21 Slot	0x31 PCIe	4		n/a	Slot_Stat_04
118	0x6F Sensor-specific	0x21 Slot	0x31 PCIe	5		n/a	Slot_Stat_05
119	0x6F Sensor-specific	0x21 Slot	0x31 PCIe	6		n/a	Slot_Stat_06
120	0x6F Sensor-specific	0x21 Slot	0x31 PCIe	7		n/a	Slot_Stat_07
121	0x6F Sensor-specific	0x21 Slot	0x31 PCIe	8		n/a	Slot_Stat_08
122	0x6F Sensor-specific	0x21 Slot	0x31 PCIe	9		n/a	Slot_Stat_09
123	0x6F Sensor-specific	0x21 Slot	0x31 PCIe	10		n/a	Slot_Stat_10
124	0x6F Sensor-specific	0x21 Slot	0x31 PCIe	11		n/a	Slot_Stat_11
125	0x6F Sensor-specific	0x21 Slot	0x31 PCIe	12		n/a	Slot_Stat_12

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Alt. Old Name	Old Name	New Name
126	0x6F Sensor-specific	0x21 Slot	0x31 PCIe	13		n/a	Slot_Stat_13
127	0x6F Sensor-specific	0x21 Slot	0x31 PCIe	14		n/a	Slot_Stat_14
128	0x6F Sensor-specific	0x21 Slot	0x31 PCIe	15		n/a	Slot_Stat_15
129	0x6F Sensor-specific	0x21 Slot	0x31 PCIe	16		n/a	Slot_Stat_16
130	0x6F Sensor-specific	0x21 Slot	0x31 PCIe	17		n/a	Slot_Stat_17
131	0x6F Sensor-specific	0x21 Slot	0x31 PCIe	18		n/a	Slot_Stat_18
132	0x01 Threshold	0x01 Temperature	0x31 PCIe	1	XX-Mezz 1	XX-PCI 1	Slot_Temp_01
133	0x01 Threshold	0x01 Temperature	0x31 PCIe	2	XX-Mezz 2	XX-PCI 2	Slot_Temp_02
134	0x01 Threshold	0x01 Temperature	0x31 PCIe	3	XX-Mezz 3	XX-PCI 3	Slot_Temp_03
135	0x01 Threshold	0x01 Temperature	0x31 PCIe	4	XX-Mezz 4	XX-PCI 4	Slot_Temp_04
136	0x01 Threshold	0x01 Temperature	0x31 PCIe	5	XX-Mezz 5	XX-PCI 5	Slot_Temp_05
137	0x01 Threshold	0x01 Temperature	0x31 PCIe	6	XX-Mezz 6	XX-PCI 6	Slot_Temp_06
138	0x01 Threshold	0x01 Temperature	0x31 PCIe	7	XX-Mezz 7	XX-PCI 7	Slot_Temp_07
139	0x01 Threshold	0x01 Temperature	0x31 PCIe	8	XX-Mezz 8	XX-PCI 8	Slot_Temp_08
140	0x01 Threshold	0x01 Temperature	0x31 PCIe	9	XX-Mezz 9	XX-PCI 9	Slot_Temp_09
141	0x01 Threshold	0x01 Temperature	0x31 PCIe	10	XX-Mezz 10	XX-PCI 10	Slot_Temp_10
142	0x01 Threshold	0x01 Temperature	0x31 PCIe	11	XX-Mezz 11	XX-PCI 11	Slot_Temp_11
143	0x01 Threshold	0x01 Temperature	0x31 PCIe	12	XX-Mezz 12	XX-PCI 12	Slot_Temp_12

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Alt. Old Name	Old Name	New Name
144	0x01 Threshold	0x01 Temperature	0x31 PCIe	13	XX-Mezz 13	XX-PCI 13	Slot_Temp_13
145	0x01 Threshold	0x01 Temperature	0x31 PCIe	14	XX-Mezz 14	XX-PCI 14	Slot_Temp_14
146	0x01 Threshold	0x01 Temperature	0x31 PCIe	15	XX-Mezz 15	XX-PCI 15	Slot_Temp_15
147	0x01 Threshold	0x01 Temperature	0x31 PCIe	16	XX-Mezz 16	XX-PCI 16	Slot_Temp_16
148	0x01 Threshold	0x01 Temperature	0x31 PCIe	17	XX-Mezz 17	XX-PCI 17	Slot_Temp_17
149	0x01 Threshold	0x01 Temperature	0x31 PCIe	18	XX-Mezz 18	XX-PCI 18	Slot_Temp_18
150[1]	0x01 Threshold	0x01 Temperature	0x04 Disk	1		XX-Exp Bay Drive	NVMeDA_Temp
151[1]	0x01 Threshold	0x01 Temperature	0x04 Disk	2		XX-HD Max	Drive_Temp
152[1]	0x01 Threshold	0x01 Temperature	0x04 Disk	3		XX-XXX HD <X> Max	DriveBx_Temp_XX
153[1]	0x01 Threshold	0x01 Temperature	0x04 Disk	4		XX-XXX HD <X> Max	DriveBk_Temp_XX
154[1]	0x01 Threshold	0x01 Temperature	0x04 Disk	5		XX-XXX HD <X> Max	DriveBx_Temp_XX
155[1]	0x01 Threshold	0x01 Temperature	0x04 Disk	6		XX-XXX HD <X> Max	DriveBk_Temp_XX
156[1]	0x01 Threshold	0x01 Temperature	0x04 Disk	7		XX-XXX HD <X> Max	DriveBx_Temp_XX
157[1]	0x01 Threshold	0x01 Temperature	0x04 Disk	8		XX-XXX HD <X> Max	DriveBk_Temp_XX
158[1]	0x01 Threshold	0x01 Temperature	0x04 Disk	9		XX-XXX HD <X> Max	DriveBx_Temp_XX
159[1]	0x01 Threshold	0x01 Temperature	0x04 Disk	10		XX-XXX HD <X> Max	DriveBk_Temp_XX
160[1]	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	0		n/a	Drive_Stat
161[1]	0x07 Severity	0x17 Add-in Card	0x0B Add-in Card	1		n/a	RAID_Stat
162	0x01 Threshold	0x01 Temperature	0x37 Air Inlet	1		01-Inlet Ambient	Inlet_Temp_01
163	0x01 Threshold	0x01 Temperature	0x37 Air Inlet	2		XX-Front Ambient	Inlet_Temp_02

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Alt. Old Name	Old Name	New Name
164	0x01 Threshold	0x01 Temperature	0x07 System Board	1		XX-Chipset	PCH_Temp_01
165	0x01 Threshold	0x01 Temperature	0x14 Power Module	0		XX-E-Fuse	EFuse_Temp_01
166	0x01 Threshold	0x02 Voltage	0x07 System Board	1		n/a	Sys12_Volt
167	0x01 Threshold	0x01 Temperature	0x06 SysMgmtMod	1		XX-iLO	BMC_Temp
168	0x6F Sensor-specific	0x28 MS Health	0x06 SysMgmtMod	1		n/a	BMC_Stat
169	0x09 Enable/ Disable	0x27 LAN	0x06 SysMgmtMod	1		n/a	BMCLANLnk_Stat
170[2]	0x6F Sensor-specific	0x22 ACPI Power	0x07 System Board	1		n/a	ACPI_Stat
171	0x6F Sensor-specific	0x05 Physical Security	0x17 System Chassis	1		Intrusion	ChasIntr_Stat
172	0x07 Severity	0x18 Chassis	0x17 System Chassis	1			SysHealth_Stat
173	0x71 OEM	0xC0 OEM	0x17 System Chassis	1		Sys Health LED	
174	0x70 OEM	0xC0 OEM	0x17 System Chassis	1		UID	UID_Stat
175	0x01 Threshold	0xD0 OEM	0x00 Unspecified	0		Pattern1	Pattern_Cnt_1
176	0x01 Threshold	0xD0 OEM	0x00 Unspecified	0		Pattern2	Pattern_Cnt_2
177	0x01 Threshold	0xD0 OEM	0x00 Unspecified	0		Pattern3	Pattern_Cnt_3
178	0x01 Threshold	0xD0 OEM	0x00 Unspecified	0		Pattern4	Pattern_Cnt_4
179	0x01 Threshold	0x01 Temperature	0x0B Add-in Card	1		XX-HD Controller	EmbRAID_Temp_1

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Alt. Old Name	Old Name	New Name
180	0x01 Threshold	0x01 Temperature	0x0B Add-in Card	3		xx-LOM	EmLOM_Temp_1
181	0x01 Threshold	0x01 Temperature	0x0B Add-in Card	4		xx-LOM Card	EmLOM_Temp_2
182	0x01 Threshold	0x0B Other	0x07 System Board	1		Power Meter	Sys_Watt
183[2]			0x07 System Board	1			Sys_ThermTrip
184	0x08 Availability	0x25 Entity Presence	0x0B Add-in Card	0		n/a	SwRAID_Pres_1
185	0x08 Availability	0x17 Add-in Card	0x0B Add-in Card	1		n/a	EmbRAID_Pres_1
186	0x06 Performanc e Met/Lags	0x03 Current	0x07 System Board	1		PwrAllocOptimize	PwrAllocOpt_Stat
187	0x6F Sensor-specific	0x29 Battery	0x28 Battery	1		Megacell Status1	Batt_Stat_01
188	0x6F Sensor-specific	0x29 Battery	0x28 Battery	2		Megacell Status2	Batt_Stat_02
189	0x6F Sensor-specific	0x29 Battery	0x28 Battery	3		Megacell Status3	Batt_Stat_03
190	0x6F Sensor-specific	0x29 Battery	0x28 Battery	4		Megacell Status4	Batt_Stat_04
191	0x01 Threshold	0x01 Temperature	0x28 Battery	1		XX-Stor Batt 1	Batt_Temp_01
192	0x01 Threshold	0x01 Temperature	0x28 Battery	2		XX-Stor Batt 2	Batt_Temp_02
193	0x01 Threshold	0x01 Temperature	0x28 Battery	3		XX-Stor Batt 3	Batt_Temp_03
194	0x01 Threshold	0x01 Temperature	0x28 Battery	4		XX-Stor Batt 4	Batt_Temp_04
195[1, 2]	0x6F Sensor-specific	0x07 Processor	0x03 Processor	1			CPU-ThermTrip

[1] Logical container bit: 0x80

[2] Event only

Sensor LUN 1

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Name	Alternate Name
0	0x6F Sensor-specific	0x0C Memory	0x20 Memory	0	Mem_Stat_C01S01	Mem_Stat_B01S01
1	0x6F Sensor-specific	0x0C Memory	0x20 Memory	1	Mem_Stat_C01S02	Mem_Stat_B01S02
2	0x6F Sensor-specific	0x0C Memory	0x20 Memory	2	Mem_Stat_C01S03	Mem_Stat_B01S03
3	0x6F Sensor-specific	0x0C Memory	0x20 Memory	3	Mem_Stat_C01S04	Mem_Stat_B01S04
4	0x6F Sensor-specific	0x0C Memory	0x20 Memory	4	Mem_Stat_C01S05	Mem_Stat_B01S05
5	0x6F Sensor-specific	0x0C Memory	0x20 Memory	5	Mem_Stat_C01S06	Mem_Stat_B01S06
6	0x6F Sensor-specific	0x0C Memory	0x20 Memory	6	Mem_Stat_C01S07	Mem_Stat_B01S07
7	0x6F Sensor-specific	0x0C Memory	0x20 Memory	7	Mem_Stat_C01S08	Mem_Stat_B01S08
8	0x6F Sensor-specific	0x0C Memory	0x20 Memory	8	Mem_Stat_C01S09	Mem_Stat_B01S09
9	0x6F Sensor-specific	0x0C Memory	0x20 Memory	9	Mem_Stat_C01S10	Mem_Stat_B01S10
10	0x6F Sensor-specific	0x0C Memory	0x20 Memory	10	Mem_Stat_C01S11	Mem_Stat_B01S11
11	0x6F Sensor-specific	0x0C Memory	0x20 Memory	11	Mem_Stat_C01S12	Mem_Stat_B01S12
12	0x6F Sensor-specific	0x0C Memory	0x20 Memory	12	Mem_Stat_C01S13	Mem_Stat_B01S13
13	0x6F Sensor-specific	0x0C Memory	0x20 Memory	13	Mem_Stat_C01S14	Mem_Stat_B01S14
14	0x6F Sensor-specific	0x0C Memory	0x20 Memory	14	Mem_Stat_C01S15	Mem_Stat_B01S15
15	0x6F Sensor-specific	0x0C Memory	0x20 Memory	15	Mem_Stat_C01S16	Mem_Stat_B01S16
16	0x6F Sensor-specific	0x0C Memory	0x20 Memory	16	Mem_Stat_C02S01	Mem_Stat_B02S01
17	0x6F Sensor-specific	0x0C Memory	0x20 Memory	17	Mem_Stat_C02S02	Mem_Stat_B02S02
18	0x6F Sensor-specific	0x0C Memory	0x20 Memory	18	Mem_Stat_C02S03	Mem_Stat_B02S03

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Name	Alternate Name
19	0x6F Sensor-specific	0x0C Memory	0x20 Memory	19	Mem_Stat_C02S04	Mem_Stat_B02S04
20	0x6F Sensor-specific	0x0C Memory	0x20 Memory	20	Mem_Stat_C02S05	Mem_Stat_B02S05
21	0x6F Sensor-specific	0x0C Memory	0x20 Memory	21	Mem_Stat_C02S06	Mem_Stat_B02S06
22	0x6F Sensor-specific	0x0C Memory	0x20 Memory	22	Mem_Stat_C02S07	Mem_Stat_B02S07
23	0x6F Sensor-specific	0x0C Memory	0x20 Memory	23	Mem_Stat_C02S08	Mem_Stat_B02S08
24	0x6F Sensor-specific	0x0C Memory	0x20 Memory	24	Mem_Stat_C02S09	Mem_Stat_B02S09
25	0x6F Sensor-specific	0x0C Memory	0x20 Memory	25	Mem_Stat_C02S10	Mem_Stat_B02S10
26	0x6F Sensor-specific	0x0C Memory	0x20 Memory	26	Mem_Stat_C02S11	Mem_Stat_B02S11
27	0x6F Sensor-specific	0x0C Memory	0x20 Memory	27	Mem_Stat_C02S12	Mem_Stat_B02S12
28	0x6F Sensor-specific	0x0C Memory	0x20 Memory	28	Mem_Stat_C02S13	Mem_Stat_B02S13
29	0x6F Sensor-specific	0x0C Memory	0x20 Memory	29	Mem_Stat_C02S14	Mem_Stat_B02S14
30	0x6F Sensor-specific	0x0C Memory	0x20 Memory	30	Mem_Stat_C02S15	Mem_Stat_B02S15
31	0x6F Sensor-specific	0x0C Memory	0x20 Memory	31	Mem_Stat_C02S16	Mem_Stat_B02S16
32	0x6F Sensor-specific	0x0C Memory	0x20 Memory	32	Mem_Stat_C03S01	Mem_Stat_B03S01
33	0x6F Sensor-specific	0x0C Memory	0x20 Memory	33	Mem_Stat_C03S02	Mem_Stat_B03S02
34	0x6F Sensor-specific	0x0C Memory	0x20 Memory	34	Mem_Stat_C03S03	Mem_Stat_B03S03
35	0x6F Sensor-specific	0x0C Memory	0x20 Memory	35	Mem_Stat_C03S04	Mem_Stat_B03S04
36	0x6F Sensor-specific	0x0C Memory	0x20 Memory	36	Mem_Stat_C03S05	Mem_Stat_B03S05
37	0x6F Sensor-specific	0x0C Memory	0x20 Memory	37	Mem_Stat_C03S06	Mem_Stat_B03S06
38	0x6F Sensor-specific	0x0C Memory	0x20 Memory	38	Mem_Stat_C03S07	Mem_Stat_B03S07

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Name	Alternate Name
39	0x6F Sensor-specific	0x0C Memory	0x20 Memory	39	Mem_Stat_C03S08	Mem_Stat_B03S08
40	0x6F Sensor-specific	0x0C Memory	0x20 Memory	40	Mem_Stat_C03S09	Mem_Stat_B03S09
41	0x6F Sensor-specific	0x0C Memory	0x20 Memory	41	Mem_Stat_C03S10	Mem_Stat_B03S10
42	0x6F Sensor-specific	0x0C Memory	0x20 Memory	42	Mem_Stat_C03S11	Mem_Stat_B03S11
43	0x6F Sensor-specific	0x0C Memory	0x20 Memory	43	Mem_Stat_C03S12	Mem_Stat_B03S12
44	0x6F Sensor-specific	0x0C Memory	0x20 Memory	44	Mem_Stat_C03S13	Mem_Stat_B03S13
45	0x6F Sensor-specific	0x0C Memory	0x20 Memory	45	Mem_Stat_C03S14	Mem_Stat_B03S14
46	0x6F Sensor-specific	0x0C Memory	0x20 Memory	46	Mem_Stat_C03S15	Mem_Stat_B03S15
47	0x6F Sensor-specific	0x0C Memory	0x20 Memory	47	Mem_Stat_C03S16	Mem_Stat_B03S16
48	0x6F Sensor-specific	0x0C Memory	0x20 Memory	48	Mem_Stat_C04S01	Mem_Stat_B04S01
49	0x6F Sensor-specific	0x0C Memory	0x20 Memory	49	Mem_Stat_C04S02	Mem_Stat_B04S02
50	0x6F Sensor-specific	0x0C Memory	0x20 Memory	50	Mem_Stat_C04S03	Mem_Stat_B04S03
51	0x6F Sensor-specific	0x0C Memory	0x20 Memory	51	Mem_Stat_C04S04	Mem_Stat_B04S04
52	0x6F Sensor-specific	0x0C Memory	0x20 Memory	52	Mem_Stat_C04S05	Mem_Stat_B04S05
53	0x6F Sensor-specific	0x0C Memory	0x20 Memory	53	Mem_Stat_C04S06	Mem_Stat_B04S06
54	0x6F Sensor-specific	0x0C Memory	0x20 Memory	54	Mem_Stat_C04S07	Mem_Stat_B04S07
55	0x6F Sensor-specific	0x0C Memory	0x20 Memory	55	Mem_Stat_C04S08	Mem_Stat_B04S08
56	0x6F Sensor-specific	0x0C Memory	0x20 Memory	56	Mem_Stat_C04S09	Mem_Stat_B04S09
57	0x6F Sensor-specific	0x0C Memory	0x20 Memory	57	Mem_Stat_C04S10	Mem_Stat_B04S10
58	0x6F Sensor-specific	0x0C Memory	0x20 Memory	58	Mem_Stat_C04S11	Mem_Stat_B04S11

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Name	Alternate Name
59	0x6F Sensor-specific	0x0C Memory	0x20 Memory	59	Mem_Stat_C04S12	Mem_Stat_B04S12
60	0x6F Sensor-specific	0x0C Memory	0x20 Memory	60	Mem_Stat_C04S13	Mem_Stat_B04S13
61	0x6F Sensor-specific	0x0C Memory	0x20 Memory	61	Mem_Stat_C04S14	Mem_Stat_B04S14
62	0x6F Sensor-specific	0x0C Memory	0x20 Memory	62	Mem_Stat_C04S15	Mem_Stat_B04S15
63	0x6F Sensor-specific	0x0C Memory	0x20 Memory	63	Mem_Stat_C04S16	Mem_Stat_B04S16
64	0x6F Sensor-specific	0x0C Memory	0x20 Memory	64	Mem_Stat_C05S01	Mem_Stat_B05S01
65	0x6F Sensor-specific	0x0C Memory	0x20 Memory	65	Mem_Stat_C05S02	Mem_Stat_B05S02
66	0x6F Sensor-specific	0x0C Memory	0x20 Memory	66	Mem_Stat_C05S03	Mem_Stat_B05S03
67	0x6F Sensor-specific	0x0C Memory	0x20 Memory	67	Mem_Stat_C05S04	Mem_Stat_B05S04
68	0x6F Sensor-specific	0x0C Memory	0x20 Memory	68	Mem_Stat_C05S05	Mem_Stat_B05S05
69	0x6F Sensor-specific	0x0C Memory	0x20 Memory	69	Mem_Stat_C05S06	Mem_Stat_B05S06
70	0x6F Sensor-specific	0x0C Memory	0x20 Memory	70	Mem_Stat_C05S07	Mem_Stat_B05S07
71	0x6F Sensor-specific	0x0C Memory	0x20 Memory	71	Mem_Stat_C05S08	Mem_Stat_B05S08
72	0x6F Sensor-specific	0x0C Memory	0x20 Memory	72	Mem_Stat_C05S09	Mem_Stat_B05S09
73	0x6F Sensor-specific	0x0C Memory	0x20 Memory	73	Mem_Stat_C05S10	Mem_Stat_B05S10
74	0x6F Sensor-specific	0x0C Memory	0x20 Memory	74	Mem_Stat_C05S11	Mem_Stat_B05S11
75	0x6F Sensor-specific	0x0C Memory	0x20 Memory	75	Mem_Stat_C05S12	Mem_Stat_B05S12
76	0x6F Sensor-specific	0x0C Memory	0x20 Memory	76	Mem_Stat_C05S13	Mem_Stat_B05S13
77	0x6F Sensor-specific	0x0C Memory	0x20 Memory	77	Mem_Stat_C05S14	Mem_Stat_B05S14
78	0x6F Sensor-specific	0x0C Memory	0x20 Memory	78	Mem_Stat_C05S15	Mem_Stat_B05S15

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Name	Alternate Name
79	0x6F Sensor-specific	0x0C Memory	0x20 Memory	79	Mem_Stat_C05S16	Mem_Stat_B05S16
80	0x6F Sensor-specific	0x0C Memory	0x20 Memory	80	Mem_Stat_C06S01	Mem_Stat_B06S01
81	0x6F Sensor-specific	0x0C Memory	0x20 Memory	81	Mem_Stat_C06S02	Mem_Stat_B06S02
82	0x6F Sensor-specific	0x0C Memory	0x20 Memory	82	Mem_Stat_C06S03	Mem_Stat_B06S03
83	0x6F Sensor-specific	0x0C Memory	0x20 Memory	83	Mem_Stat_C06S04	Mem_Stat_B06S04
84	0x6F Sensor-specific	0x0C Memory	0x20 Memory	84	Mem_Stat_C06S05	Mem_Stat_B06S05
85	0x6F Sensor-specific	0x0C Memory	0x20 Memory	85	Mem_Stat_C06S06	Mem_Stat_B06S06
86	0x6F Sensor-specific	0x0C Memory	0x20 Memory	86	Mem_Stat_C06S07	Mem_Stat_B06S07
87	0x6F Sensor-specific	0x0C Memory	0x20 Memory	87	Mem_Stat_C06S08	Mem_Stat_B06S08
88	0x6F Sensor-specific	0x0C Memory	0x20 Memory	88	Mem_Stat_C06S09	Mem_Stat_B06S09
89	0x6F Sensor-specific	0x0C Memory	0x20 Memory	89	Mem_Stat_C06S10	Mem_Stat_B06S10
90	0x6F Sensor-specific	0x0C Memory	0x20 Memory	90	Mem_Stat_C06S11	Mem_Stat_B06S11
91	0x6F Sensor-specific	0x0C Memory	0x20 Memory	91	Mem_Stat_C06S12	Mem_Stat_B06S12
92	0x6F Sensor-specific	0x0C Memory	0x20 Memory	92	Mem_Stat_C06S13	Mem_Stat_B06S13
93	0x6F Sensor-specific	0x0C Memory	0x20 Memory	93	Mem_Stat_C06S14	Mem_Stat_B06S14
94	0x6F Sensor-specific	0x0C Memory	0x20 Memory	94	Mem_Stat_C06S15	Mem_Stat_B06S15
95	0x6F Sensor-specific	0x0C Memory	0x20 Memory	95	Mem_Stat_C06S16	Mem_Stat_B06S16
96	0x6F Sensor-specific	0x0C Memory	0x20 Memory	96	Mem_Stat_C07S01	Mem_Stat_B07S01
97	0x6F Sensor-specific	0x0C Memory	0x20 Memory	97	Mem_Stat_C07S02	Mem_Stat_B07S02
98	0x6F Sensor-specific	0x0C Memory	0x20 Memory	98	Mem_Stat_C07S03	Mem_Stat_B07S03

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Name	Alternate Name
99	0x6F Sensor-specific	0x0C Memory	0x20 Memory	99	Mem_Stat_C07S04	Mem_Stat_B07S04
100	0x6F Sensor-specific	0x0C Memory	0x20 Memory	100	Mem_Stat_C07S05	Mem_Stat_B07S05
101	0x6F Sensor-specific	0x0C Memory	0x20 Memory	101	Mem_Stat_C07S06	Mem_Stat_B07S06
102	0x6F Sensor-specific	0x0C Memory	0x20 Memory	102	Mem_Stat_C07S07	Mem_Stat_B07S07
103	0x6F Sensor-specific	0x0C Memory	0x20 Memory	103	Mem_Stat_C07S08	Mem_Stat_B07S08
104	0x6F Sensor-specific	0x0C Memory	0x20 Memory	104	Mem_Stat_C07S09	Mem_Stat_B07S09
105	0x6F Sensor-specific	0x0C Memory	0x20 Memory	105	Mem_Stat_C07S10	Mem_Stat_B07S10
106	0x6F Sensor-specific	0x0C Memory	0x20 Memory	106	Mem_Stat_C07S11	Mem_Stat_B07S11
107	0x6F Sensor-specific	0x0C Memory	0x20 Memory	107	Mem_Stat_C07S12	Mem_Stat_B07S12
108	0x6F Sensor-specific	0x0C Memory	0x20 Memory	108	Mem_Stat_C07S13	Mem_Stat_B07S13
109	0x6F Sensor-specific	0x0C Memory	0x20 Memory	109	Mem_Stat_C07S14	Mem_Stat_B07S14
110	0x6F Sensor-specific	0x0C Memory	0x20 Memory	110	Mem_Stat_C07S15	Mem_Stat_B07S15
111	0x6F Sensor-specific	0x0C Memory	0x20 Memory	111	Mem_Stat_C07S16	Mem_Stat_B07S16
112	0x6F Sensor-specific	0x0C Memory	0x20 Memory	112	Mem_Stat_C08S01	Mem_Stat_B08S01
113	0x6F Sensor-specific	0x0C Memory	0x20 Memory	113	Mem_Stat_C08S02	Mem_Stat_B08S02
114	0x6F Sensor-specific	0x0C Memory	0x20 Memory	114	Mem_Stat_C08S03	Mem_Stat_B08S03
115	0x6F Sensor-specific	0x0C Memory	0x20 Memory	115	Mem_Stat_C08S04	Mem_Stat_B08S04
116	0x6F Sensor-specific	0x0C Memory	0x20 Memory	116	Mem_Stat_C08S05	Mem_Stat_B08S05
117	0x6F Sensor-specific	0x0C Memory	0x20 Memory	117	Mem_Stat_C08S06	Mem_Stat_B08S06
118	0x6F Sensor-specific	0x0C Memory	0x20 Memory	118	Mem_Stat_C08S07	Mem_Stat_B08S07

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Name	Alternate Name
119	0x6F Sensor-specific	0x0C Memory	0x20 Memory	119	Mem_Stat_C08S08	Mem_Stat_B08S08
120	0x6F Sensor-specific	0x0C Memory	0x20 Memory	120	Mem_Stat_C08S09	Mem_Stat_B08S09
121	0x6F Sensor-specific	0x0C Memory	0x20 Memory	121	Mem_Stat_C08S10	Mem_Stat_B08S10
122	0x6F Sensor-specific	0x0C Memory	0x20 Memory	122	Mem_Stat_C08S11	Mem_Stat_B08S11
123	0x6F Sensor-specific	0x0C Memory	0x20 Memory	123	Mem_Stat_C08S12	Mem_Stat_B08S12
124	0x6F Sensor-specific	0x0C Memory	0x20 Memory	124	Mem_Stat_C08S13	Mem_Stat_B08S13
125	0x6F Sensor-specific	0x0C Memory	0x20 Memory	125	Mem_Stat_C08S14	Mem_Stat_B08S14
126	0x6F Sensor-specific	0x0C Memory	0x20 Memory	126	Mem_Stat_C08S15	Mem_Stat_B08S15
127	0x6F Sensor-specific	0x0C Memory	0x20 Memory	127	Mem_Stat_C08S16	Mem_Stat_B08S16

FRU LUN 0

FRU ID	Device Type	Modifier	Entity ID	Instance	Old Name	Alternate C-class Name
0	0x10 MC FRU Device	0x00 IPMI FRU	0x07 System Board	1	N/A	BladeFRU
1–15	-	-	-	-	-	-
16	0x10 MC FRU Device	0x00 IPMI FRU	0x03 Processor	1	CPU 1	
17	0x10 MC FRU Device	0x00 IPMI FRU	0x03 Processor	2	CPU 2	
18	0x10 MC FRU Device	0x00 IPMI FRU	0x03 Processor	3	CPU 3	
19	0x10 MC FRU Device	0x00 IPMI FRU	0x03 Processor	4	CPU 4	
20	0x10 MC FRU Device	0x00 IPMI FRU	0x03 Processor	5	CPU 5	
21	0x10 MC FRU Device	0x00 IPMI FRU	0x03 Processor	6	CPU 6	
22	0x10 MC FRU Device	0x00 IPMI FRU	0x03 Processor	7	CPU 7	

Table Continued

FRU ID	Device Type	Modifier	Entity ID	Instance	Old Name	Alternate C-class Name
23	0x10 MC FRU Device	0x00 IPMI FRU	0x03 Processor	8	CPU 8	
24-59	-	-	-	-	-	-
60	-	-	-	-	Battery	-
61	-	-	-	-	Battery	-
62	-	-	-	-	Battery	-
63	-	-	-	-	Battery	-
64	0x10 MC FRU Device	0x00 IPMI FRU	0x2C I/O module	1	Ethernet Adptr	BLOM DC01
65	0x10 MC FRU Device	0x00 IPMI FRU	0x2C I/O module	2	Ethernet Adptr	BLOM DC02
66	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	4	Mezz4	Mezz4
67	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	5	Mezz5	Mezz5
68	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	6	Mezz6	Mezz6
69	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	7	Mezz7	Mezz7
70	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	8	Mezz8	Mezz8
71	0x10 MC FRU Device	0x00 IPMI FRU	0x2C I/O module	3	Ethernet Adptr	BLOM DC03
72	0x10 MC FRU Device	0x00 IPMI FRU	0x2C I/O module	4	Ethernet Adptr	BLOM DC04
73	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	9	Mezz9	Mezz9
74	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	3	Mezz3	Mezz3
75	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	10	Mezz10	Mezz10
76	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	2	Mezz2	Mezz2
77	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	11	Mezz11	Mezz11
78	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	1	Mezz1	Mezz1
79	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	12	Mezz12	Mezz12

Table Continued

FRU ID	Device Type	Modifier	Entity ID	Instance	Old Name	Alternate C-class Name
80	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	13	Mezz13	Mezz13
81	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	14	Mezz14	Mezz14
82	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	15	Mezz15	Mezz15
83	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	16	Mezz16	Mezz16
84	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	17	Mezz17	Mezz17
85	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	18	Mezz18	Mezz18
86-94	-	-	-	-	-	-
95	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	1	Mezz1PassThruFRU	
96	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	2	Mezz2PassThruFRU	
97	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	1	PCI-E<X>RiserFRU/ MXM<X>RiserFRU	
98	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	2	PCI-E<X>RiserFRU/ MXM<X>RiserFRU	
99	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	3	PCI-E<X>RiserFRU/ MXM<X>RiserFRU	
100	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	4	PCI-E<X>RiserFRU/ MXM<X>RiserFRU	
101	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	5	PCI-E<X>RiserFRU/ MXM<X>RiserFRU	
102	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	6	PCI-E<X>RiserFRU/ MXM<X>RiserFRU	
103	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	7	PCI-E<X>RiserFRU/ MXM<X>RiserFRU	
104	0x10 MC FRU Device	0x00 IPMI FRU	0x31 PCI X -Bus	8	PCI-E<X>RiserFRU/ MXM<X>RiserFRU	
105-127	-	-	-	-	-	-
128	0x10 MC FRU Device	0x00 IPMI FRU	0x15 Drive Backplane	1	Stor Bkplane 1A	
129	0x10 MC FRU Device	0x00 IPMI FRU	0x15 Drive Backplane	1	Stor Bkplane 1B	

Table Continued

FRU ID	Device Type	Modifier	Entity ID	Instance	Old Name	Alternate C-class Name
130	0x10 MC FRU Device	0x00 IPMI FRU	0x15 Drive Backplane	2	Stor Bkplane 2A	
131	0x10 MC FRU Device	0x00 IPMI FRU	0x15 Drive Backplane	2	Stor Bkplane 2B	
132	0x10 MC FRU Device	0x00 IPMI FRU	0x15 Drive Backplane	3	Stor Bkplane 3A	
133	0x10 MC FRU Device	0x00 IPMI FRU	0x15 Drive Backplane	3	Stor Bkplane 3B	
134	0x10 MC FRU Device	0x00 IPMI FRU	0x15 Drive Backplane	4	Stor Bkplane 4A	
135	0x10 MC FRU Device	0x00 IPMI FRU	0x15 Drive Backplane	4	Stor Bkplane 4B	
136	0x10 MC FRU Device	0x00 IPMI FRU	0x15 Drive Backplane	5	Stor Bkplane 5A	
137	0x10 MC FRU Device	0x00 IPMI FRU	0x15 Drive Backplane	5	Stor Bkplane 5B	
138	0x10 MC FRU Device	0x00 IPMI FRU	0x15 Drive Backplane	6	Stor Bkplane 6A	
139	0x10 MC FRU Device	0x00 IPMI FRU	0x15 Drive Backplane	6	Stor Bkplane 6B	
140	0x10 MC FRU Device	0x00 IPMI FRU	0x15 Drive Backplane	7	Stor Bkplane 7A	
141	0x10 MC FRU Device	0x00 IPMI FRU	0x15 Drive Backplane	7	Stor Bkplane 7B	
142	0x10 MC FRU Device	0x00 IPMI FRU	0x15 Drive Backplane	8	Stor Bkplane 8A	
143	0x10 MC FRU Device	0x00 IPMI FRU	0x15 Drive Backplane	8	Stor Bkplane 8B	
144-245	-	-	-	-	-	-
246	0x10 MC FRU Device	0x00 IPMI FRU	0x06 SysMgmtMod	1	BMC Controller	N/A
247	0x10 MC FRU Device	0x00 IPMI FRU	0x22 BIOS	1	MB BIOS	N/A
248-254	-	-	-	-	-	-
255					Reserved	

FRU LUN 1

FRU ID	Device Type	Modifier	Entity ID	Instance	Name
0	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	1	MEM_FRU_C01S01
1	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	2	MEM_FRU_C01S02
2	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	3	MEM_FRU_C01S03
3	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	4	MEM_FRU_C01S04
4	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	5	MEM_FRU_C01S05
5	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	6	MEM_FRU_C01S06
6	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	7	MEM_FRU_C01S07
7	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	8	MEM_FRU_C01S08
8	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	9	MEM_FRU_C01S09
9	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	10	MEM_FRU_C01S10
10	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	11	MEM_FRU_C01S11
11	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	12	MEM_FRU_C01S12
12	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	13	MEM_FRU_C01S13
13	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	14	MEM_FRU_C01S14
14	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	15	MEM_FRU_C01S15
15	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	16	MEM_FRU_C01S16
16	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	17	MEM_FRU_C02S01
17	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	18	MEM_FRU_C02S02
18	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	19	MEM_FRU_C02S03
19	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	20	MEM_FRU_C02S04
20	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	21	MEM_FRU_C02S05
21	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	22	MEM_FRU_C02S06
22	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	23	MEM_FRU_C02S07
23	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	24	MEM_FRU_C02S08
24	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	25	MEM_FRU_C02S09
25	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	26	MEM_FRU_C02S10
26	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	27	MEM_FRU_C02S11
27	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	28	MEM_FRU_C02S12
28	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	29	MEM_FRU_C02S13
29	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	30	MEM_FRU_C02S14
30	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	31	MEM_FRU_C02S15
31	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	32	MEM_FRU_C02S16

Table Continued

FRU ID	Device Type	Modifier	Entity ID	Instance	Name
32	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	33	MEM_FRU_C03S01
33	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	34	MEM_FRU_C03S02
34	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	35	MEM_FRU_C03S03
35	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	36	MEM_FRU_C03S04
36	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	37	MEM_FRU_C03S05
37	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	38	MEM_FRU_C03S06
38	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	39	MEM_FRU_C03S07
39	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	40	MEM_FRU_C03S08
40	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	41	MEM_FRU_C03S09
41	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	42	MEM_FRU_C03S10
42	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	43	MEM_FRU_C03S11
43	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	44	MEM_FRU_C03S12
44	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	45	MEM_FRU_C03S13
45	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	46	MEM_FRU_C03S14
46	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	47	MEM_FRU_C03S15
47	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	48	MEM_FRU_C03S16
48	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	49	MEM_FRU_C04S01
49	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	50	MEM_FRU_C04S02
50	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	51	MEM_FRU_C04S03
51	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	52	MEM_FRU_C04S04
52	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	53	MEM_FRU_C04S05
53	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	54	MEM_FRU_C04S06
54	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	55	MEM_FRU_C04S07
55	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	56	MEM_FRU_C04S08
56	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	57	MEM_FRU_C04S09
57	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	58	MEM_FRU_C04S10
58	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	59	MEM_FRU_C04S11
59	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	60	MEM_FRU_C04S12
60	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	61	MEM_FRU_C04S13
61	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	62	MEM_FRU_C04S14
62	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	63	MEM_FRU_C04S15
63	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	64	MEM_FRU_C04S16
64	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	65	MEM_FRU_C05S01

Table Continued

FRU ID	Device Type	Modifier	Entity ID	Instance	Name
65	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	66	MEM_FRU_C05S02
66	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	67	MEM_FRU_C05S03
67	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	68	MEM_FRU_C05S04
68	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	69	MEM_FRU_C05S05
69	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	70	MEM_FRU_C05S06
70	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	71	MEM_FRU_C05S07
71	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	72	MEM_FRU_C05S08
72	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	73	MEM_FRU_C05S09
73	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	74	MEM_FRU_C05S10
74	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	75	MEM_FRU_C05S11
75	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	76	MEM_FRU_C05S12
76	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	77	MEM_FRU_C05S13
77	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	78	MEM_FRU_C05S14
78	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	79	MEM_FRU_C05S15
79	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	80	MEM_FRU_C05S16
80	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	81	MEM_FRU_C06S01
81	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	82	MEM_FRU_C06S02
82	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	83	MEM_FRU_C06S03
83	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	84	MEM_FRU_C06S04
84	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	85	MEM_FRU_C06S05
85	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	86	MEM_FRU_C06S06
86	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	87	MEM_FRU_C06S07
87	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	88	MEM_FRU_C06S08
88	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	89	MEM_FRU_C06S09
89	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	90	MEM_FRU_C06S10
90	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	91	MEM_FRU_C06S11
91	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	92	MEM_FRU_C06S12
92	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	93	MEM_FRU_C06S13
93	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	94	MEM_FRU_C06S14
94	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	95	MEM_FRU_C06S15
95	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	96	MEM_FRU_C06S16
96	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	97	MEM_FRU_C07S01
97	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	98	MEM_FRU_C07S02

Table Continued

FRU ID	Device Type	Modifier	Entity ID	Instance	Name
98	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	99	MEM_FRU_C07S03
99	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	100	MEM_FRU_C07S04
100	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	101	MEM_FRU_C07S05
101	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	102	MEM_FRU_C07S06
102	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	103	MEM_FRU_C07S07
103	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	104	MEM_FRU_C07S08
104	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	105	MEM_FRU_C07S09
105	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	106	MEM_FRU_C07S10
106	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	107	MEM_FRU_C07S11
107	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	108	MEM_FRU_C07S12
108	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	109	MEM_FRU_C07S13
109	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	110	MEM_FRU_C07S14
110	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	111	MEM_FRU_C07S15
111	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	112	MEM_FRU_C07S16
112	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	113	MEM_FRU_C08S01
113	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	114	MEM_FRU_C08S02
114	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	115	MEM_FRU_C08S03
115	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	116	MEM_FRU_C08S04
116	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	117	MEM_FRU_C08S05
117	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	118	MEM_FRU_C08S06
118	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	119	MEM_FRU_C08S07
119	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	120	MEM_FRU_C08S08
120	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	121	MEM_FRU_C08S09
121	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	122	MEM_FRU_C08S10
122	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	123	MEM_FRU_C08S11
123	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	124	MEM_FRU_C08S12
124	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	125	MEM_FRU_C08S13
125	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	126	MEM_FRU_C08S14
126	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	127	MEM_FRU_C08S15
127	0x10 MC FRU Device	0x01 DIMM	0x20 Memory	128	MEM_FRU_C08S16
128–254	-	-	-	-	-
255					Reserved

ChassisMC Sensor LUN 0

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Old Name	New Name
0	0x6F Sensor-specific	0x28 Management Subsystem Health	0x17 System Chassis	0		Enclosure Status
1	0x01 Threshold[1]	0x04 Fan	0x1D Cooling	1	FAN 1	Fan_Speed_01
2	0x08 Availability	0x04 Fan	0x1D Cooling	1		Fan_Pres_01
3	0x07 Severity	0x04 Fan	0x1D Cooling	1		Fan_Stat_01
4	0x01 Threshold[1]	0x04 Fan	0x1D Cooling	2	FAN 2	Fan_Speed_02
5	0x08 Availability	0x04 Fan	0x1D Cooling	2		Fan_Pres_02
6	0x07 Severity	0x04 Fan	0x1D Cooling	2		Fan_Stat_02
7	0x01 Threshold[1]	0x04 Fan	0x1D Cooling	3	FAN 3	Fan_Speed_03
8	0x08 Availability	0x04 Fan	0x1D Cooling	3		Fan_Pres_03
9	0x07 Severity	0x04 Fan	0x1D Cooling	3		Fan_Stat_03
10	0x01 Threshold[1]	0x04 Fan	0x1D Cooling	4	FAN 4	Fan_Speed_04
11	0x08 Availability	0x04 Fan	0x1D Cooling	4		Fan_Pres_04
12	0x07 Severity	0x04 Fan	0x1D Cooling	4		Fan_Stat_04
13	0x01 Threshold[1]	0x04 Fan	0x1D Cooling	5	FAN 5	Fan_Speed_05
14	0x08 Availability	0x04 Fan	0x1D Cooling	5		Fan_Pres_05
15	0x07 Severity	0x04 Fan	0x1D Cooling	5		Fan_Stat_05
16	0x01 Threshold[1]	0x04 Fan	0x1D Cooling	6	FAN 6	Fan_Speed_06
17	0x08 Availability	0x04 Fan	0x1D Cooling	6		Fan_Pres_06
18	0x07 Severity	0x04 Fan	0x1D Cooling	6		Fan_Stat_06
19	0x01 Threshold[1]	0x04 Fan	0x1D Cooling	7	FAN 7	Fan_Speed_07
20	0x08 Availability	0x04 Fan	0x1D Cooling	7		Fan_Pres_07
21	0x07 Severity	0x04 Fan	0x1D Cooling	7		Fan_Stat_07
22	0x01 Threshold[1]	0x04 Fan	0x1D Cooling	8	FAN 8	Fan_Speed_08
23	0x08 Availability	0x04 Fan	0x1D Cooling	8		Fan_Pres_08
24	0x07 Severity	0x04 Fan	0x1D Cooling	8		Fan_Stat_08
25	0x01 Threshold[1]	0x04 Fan	0x1D Cooling	9	FAN 9	Fan_Speed_09
26	0x08 Availability	0x04 Fan	0x1D Cooling	9		Fan_Pres_09
27	0x07 Severity	0x04 Fan	0x1D Cooling	9		Fan_Stat_09
28	0x01 Threshold[1]	0x04 Fan	0x1D Cooling	10	FAN 10	Fan_Speed_10
29	0x08 Availability	0x04 Fan	0x1D Cooling	10		Fan_Pres_10
30	0x07 Severity	0x04 Fan	0x1D Cooling	10		Fan_Stat_10

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Old Name	New Name
31	0x01 Threshold[1]	0x04 Fan	0x1D Cooling	11	FAN 11	Fan_Speed_11
32	0x08 Availability	0x04 Fan	0x1D Cooling	11		Fan_Pres_11
33	0x07 Severity	0x04 Fan	0x1D Cooling	11		Fan_Stat_11
34	0x01 Threshold[1]	0x04 Fan	0x1D Cooling	12	FAN 12	Fan_Speed_12
35	0x08 Availability	0x04 Fan	0x1D Cooling	12		Fan_Pres_12
36	0x07 Severity	0x04 Fan	0x1D Cooling	12		Fan_Stat_12
37	0x01 Threshold[1]	0x04 Fan	0x1D Cooling	13	FAN 13	Fan_Speed_13
38	0x08 Availability	0x04 Fan	0x1D Cooling	13		Fan_Pres_13
39	0x07 Severity	0x04 Fan	0x1D Cooling	13		Fan_Stat_13
40	0x01 Threshold[1]	0x04 Fan	0x1D Cooling	14	FAN 14	Fan_Speed_14
41	0x08 Availability	0x04 Fan	0x1D Cooling	14		Fan_Pres_14
42	0x07 Severity	0x04 Fan	0x1D Cooling	14		Fan_Stat_14
43	0x01 Threshold[1]	0x04 Fan	0x1D Cooling	15	FAN 15	Fan_Speed_15
44	0x08 Availability	0x04 Fan	0x1D Cooling	15		Fan_Pres_15
45	0x07 Severity	0x04 Fan	0x1D Cooling	15		Fan_Stat_15
46	0x01 Threshold[1]	0x04 Fan	0x1D Cooling	16	FAN 16	Fan_Speed_16
47	0x08 Availability	0x04 Fan	0x1D Cooling	16		Fan_Pres_16
48	0x07 Severity	0x04 Fan	0x1D Cooling	16		Fan_Stat_16
49	0x0B Redundancy	0x04 Fan	0x1E Cooling Domain	1	Fans	Fan_Stat_G01
50	0x01 Threshold [1]	0x04 Fan	0x1D Cooling	17		Pump_CPUN_Speed [2]
51	0x8 Availability	0x04 Fan	0x1D Cooling	17		Pump_CPUN_Pres [2]
52	0x07 Severity	0x04 Fan	0x1D Cooling	17		Pump_CPUN_Stat [2]
53	0x0B Redundancy	0x04 Fan	0x1D Cooling	21		Pumps [2]
54	0x01 Threshold [1]	0x04 Fan	0x1D Cooling	18		Loop_GN_Speed [2]
55	0x08 Availability	0x04 Fan	0x1D Cooling	18		Loop_GN_Pres [2]
56	0x07 Severity	0x04 Fan	0x1D Cooling	18		Loop_GN_Stat [2]
57	0x0B Redundancy	0x04 Fan	0x1D Cooling	22		Loops [2]
58	0x6F Sensor-specific	0x08 Power Supply	0x0A Power Supply	1	Power Supply 1	PS_Stat_01
59	0x6F Sensor-specific	0x08 Power Supply	0x0A Power Supply	2	Power Supply 2	PS_Stat_02
60	0x6F Sensor-specific	0x08 Power Supply	0x0A Power Supply	3	Power Supply 3	PS_Stat_03

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Old Name	New Name
61	0x6F Sensor-specific	0x08 Power Supply	0x0A Power Supply	4	Power Supply 4	PS_Stat_04
62	0x6F Sensor-specific	0x08 Power Supply	0x0A Power Supply	5	Power Supply 5	PS_Stat_05
63	0x6F Sensor-specific	0x08 Power Supply	0x0A Power Supply	6	Power Supply 6	PS_Stat_06
64	0x6F Sensor-specific	0x08 Power Supply	0x0A Power Supply	7	Power Supply 7	PS_Stat_07
65	0x6F Sensor-specific	0x08 Power Supply	0x0A Power Supply	8	Power Supply 8	PS_Stat_08
66	0x01 Threshold	0x08 Power Supply	0x0A Power Supply	1	PS 1 Output	PS_Watt_Out_01[3]
67	0x01 Threshold	0x08 Power Supply	0x0A Power Supply	2	PS 2 Output	PS_Watt_Out_02[3]
68	0x01 Threshold	0x08 Power Supply	0x0A Power Supply	3	PS 3 Output	PS_Watt_Out_03[3]
69	0x01 Threshold	0x08 Power Supply	0x0A Power Supply	4	PS 4 Output	PS_Watt_Out_04[3]
70	0x01 Threshold	0x08 Power Supply	0x0A Power Supply	5	PS 5 Output	PS_Watt_Out_05[3]
71	0x01 Threshold	0x08 Power Supply	0x0A Power Supply	6	PS 6 Output	PS_Watt_Out_06[3]
72	0x01 Threshold	0x08 Power Supply	0x0A Power Supply	7	PS 7 Output	PS_Watt_Out_07[3]
73	0x01 Threshold	0x08 Power Supply	0x0A Power Supply	8	PS 8 Output	PS_Watt_Out_08[3]
74	0x0B Redundancy	0x08 Power Supply	0x13 Power Domain	1	Power Supplies	PS_Stat_G01
75	0x01 Threshold	0x01 Temperature	0x0A Power Supply	1	XX-P/S 1	PS_Temp_01
76	0x01 Threshold	0x01 Temperature	0x0A Power Supply	2	XX-P/S 2	PS_Temp_02
77	0x01 Threshold	0x01 Temperature	0x0A Power Supply	3	XX-P/S 3	PS_Temp_03
78	0x01 Threshold	0x01 Temperature	0x0A Power Supply	4	XX-P/S 4	PS_Temp_04
79	0x01 Threshold	0x01 Temperature	0x0A Power Supply	5	XX-P/S 5	PS_Temp_05
80	0x01 Threshold	0x01 Temperature	0x0A Power Supply	6	XX-P/S 6	PS_Temp_06
81	0x01 Threshold	0x01 Temperature	0x0A Power Supply	7	XX-P/S 7	PS_Temp_07
82	0x01 Threshold	0x01 Temperature	0x0A Power Supply	8	XX-P/S 8	PS_Temp_08
83	0x01 Threshold	0x01 Temperature	0x0A Power Supply	1	XX-P/S 1 Inlet	PSInlet_Temp_01
84	0x01 Threshold	0x01 Temperature	0x0A Power Supply	2	XX-P/S 2 Inlet	PSInlet_Temp_02
85	0x01 Threshold	0x01 Temperature	0x0A Power Supply	3	XX-P/S 3 Inlet	PSInlet_Temp_03
86	0x01 Threshold	0x01 Temperature	0x0A Power Supply	4	XX-P/S 4 Inlet	PSInlet_Temp_04

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Old Name	New Name
87	0x01 Threshold	0x01 Temperature	0x0A Power Supply	5	XX-P/S 5 Inlet	PSInlet_Temp_05
88	0x01 Threshold	0x01 Temperature	0x0A Power Supply	6	XX-P/S 6 Inlet	PSInlet_Temp_06
89	0x01 Threshold	0x01 Temperature	0x0A Power Supply	7	XX-P/S 7 Inlet	PSInlet_Temp_07
90	0x01 Threshold	0x01 Temperature	0x0A Power Supply	8	XX-P/S 8 Inlet	PSInlet_Temp_08
91	0x01 Threshold	0x08 Power Supply	0x0A Power Supply	1	PS 1 Input	PS_Watt_In_01[3, 4]
92	0x01 Threshold	0x08 Power Supply	0x0A Power Supply	2	PS 2 Input	PS_Watt_In_02[3, 4]
93	0x01 Threshold	0x08 Power Supply	0x0A Power Supply	3	PS 3 Input	PS_Watt_In_03[3, 4]
94	0x01 Threshold	0x08 Power Supply	0x0A Power Supply	4	PS 4 Input	PS_Watt_In_04[3, 4]
95	0x01 Threshold	0x08 Power Supply	0x0A Power Supply	5	PS 5 Input	PS_Watt_In_05[3, 4]
96	0x01 Threshold	0x08 Power Supply	0x0A Power Supply	6	PS 6 Input	PS_Watt_In_06[3, 4]
97	0x01 Threshold	0x08 Power Supply	0x0A Power Supply	7	PS 7 Input	PS_Watt_In_07[3, 4]
98	0x01 Threshold	0x08 Power Supply	0x0A Power Supply	8	PS 8 Input	PS_Watt_In_08[3, 4]
99	0x01 Threshold	0x02 Voltage	0x0A Power Supply	1		PS_Volt_Out_01[3, 4]
100	0x01 Threshold	0x02 Voltage	0x0A Power Supply	2		PS_Volt_Out_02[3, 4]
101	0x01 Threshold	0x02 Voltage	0x0A Power Supply	3		PS_Volt_Out_03[3, 4]
102	0x01 Threshold	0x02 Voltage	0x0A Power Supply	4		PS_Volt_Out_04[3, 4]
103	0x01 Threshold	0x02 Voltage	0x0A Power Supply	5		PS_Volt_Out_05[3, 4]
104	0x01 Threshold	0x02 Voltage	0x0A Power Supply	6		PS_Volt_Out_06[3, 4]
105	0x01 Threshold	0x02 Voltage	0x0A Power Supply	7		PS_Volt_Out_07[3, 4]
106	0x01 Threshold	0x02 Voltage	0x0A Power Supply	8		PS_Volt_Out_08[3, 4]
107	0x01 Threshold	0x02 Voltage	0x0A Power Supply	1		PS_Volt_In_01[3, 4]
108	0x01 Threshold	0x02 Voltage	0x0A Power Supply	2		PS_Volt_In_02[3, 4]
109	0x01 Threshold	0x02 Voltage	0x0A Power Supply	3		PS_Volt_In_03[3, 4]
110	0x01 Threshold	0x02 Voltage	0x0A Power Supply	4		PS_Volt_In_04[3, 4]
111	0x01 Threshold	0x02 Voltage	0x0A Power Supply	5		PS_Volt_In_05[3, 4]
112	0x01 Threshold	0x02 Voltage	0x0A Power Supply	6		PS_Volt_In_06[3, 4]
113	0x01 Threshold	0x02 Voltage	0x0A Power Supply	7		PS_Volt_In_07[3, 4]
114	0x01 Threshold	0x02 Voltage	0x0A Power Supply	8		PS_Volt_In_08[3, 4]
115	0x01 Threshold	0x03 Current	0x0A Power Supply	1		PS_Curr_Out_01 [3, 4]
116	0x01 Threshold	0x03 Current	0x0A Power Supply	2		PS_Curr_Out_02[3, 4]

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Old Name	New Name
117	0x01 Threshold	0x03 Current	0x0A Power Supply	3		PS_Curr_Out_03[3, 4]
118	0x01 Threshold	0x03 Current	0x0A Power Supply	4		PS_Curr_Out_04[3, 4]
119	0x01 Threshold	0x03 Current	0x0A Power Supply	5		PS_Curr_Out_05[3, 4]
120	0x01 Threshold	0x03 Current	0x0A Power Supply	6		PS_Curr_Out_06[3, 4]
121	0x01 Threshold	0x03 Current	0x0A Power Supply	7		PS_Curr_Out_07[3, 4]
122	0x01 Threshold	0x03 Current	0x0A Power Supply	8		PS_Curr_Out_08[3, 4]
123	0x01 Threshold	0x03 Current	0x0A Power Supply	1		PS_Curr_In_01[3, 4]
124	0x01 Threshold	0x03 Current	0x0A Power Supply	2		PS_Curr_In_02[3, 4]
125	0x01 Threshold	0x03 Current	0x0A Power Supply	3		PS_Curr_In_03[3, 4]
126	0x01 Threshold	0x03 Current	0x0A Power Supply	4		PS_Curr_In_04[3, 4]
127	0x01 Threshold	0x03 Current	0x0A Power Supply	5		PS_Curr_In_05[3, 4]
128	0x01 Threshold	0x03 Current	0x0A Power Supply	6		PS_Curr_In_06[3, 4]
129	0x01 Threshold	0x03 Current	0x0A Power Supply	7		PS_Curr_In_07[3, 4]
130	0x01 Threshold	0x03 Current	0x0A Power Supply	8		PS_Curr_In_08[3, 4]
131-254	-	-	-	-	-	-

[1] Old ERC: 0x0A Availability

[2] This sensors is supported with iLO 5 2.40 and later for Gen10 plus Apollo 2000, Apollo 6500 and DL360 systems. Where 'N' may vary depending on number of CPU and GPU for a specific platform.

[3] Readings are displayed only if the server is connected to a Flex Slot power supply.

[4] This sensor is supported with iLO 5 2.31 and later.

RAIDMC sensor LUN 0

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Old Name	New Name
2	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	96	CN PX Bay Z	Dr_Stat_XXY_BZZZ
3	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	97	CN PX Bay Z	Dr_Stat_XXY_BZZZ
4	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	98	CN PX Bay Z	Dr_Stat_XXY_BZZZ
5	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	99	CN PX Bay Z	Dr_Stat_XXY_BZZZ
6	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	100	CN PX Bay Z	Dr_Stat_XXY_BZZZ
7	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	101	CN PX Bay Z	Dr_Stat_XXY_BZZZ

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Old Name	New Name
8	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	102	CN PX Bay Z	Dr_Stat_XXY_BZZZ
9	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	103	CN PX Bay Z	Dr_Stat_XXY_BZZZ
10	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	104	CN PX Bay Z	Dr_Stat_XXY_BZZZ
11	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	105	CN PX Bay Z	Dr_Stat_XXY_BZZZ
12	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	106	CN PX Bay Z	Dr_Stat_XXY_BZZZ
13	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	107	CN PX Bay Z	Dr_Stat_XXY_BZZZ
14	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	108	CN PX Bay Z	Dr_Stat_XXY_BZZZ
15	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	109	CN PX Bay Z	Dr_Stat_XXY_BZZZ
16	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	110	CN PX Bay Z	Dr_Stat_XXY_BZZZ
17	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	111	CN PX Bay Z	Dr_Stat_XXY_BZZZ
18	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	112	CN PX Bay Z	Dr_Stat_XXY_BZZZ
19	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	113	CN PX Bay Z	Dr_Stat_XXY_BZZZ
20	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	114	CN PX Bay Z	Dr_Stat_XXY_BZZZ
21	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	115	CN PX Bay Z	Dr_Stat_XXY_BZZZ
22	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	116	CN PX Bay Z	Dr_Stat_XXY_BZZZ
23	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	117	CN PX Bay Z	Dr_Stat_XXY_BZZZ
24	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	118	CN PX Bay Z	Dr_Stat_XXY_BZZZ
25	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	119	CN PX Bay Z	Dr_Stat_XXY_BZZZ
26	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	120	CN PX Bay Z	Dr_Stat_XXY_BZZZ
27	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	121	CN PX Bay Z	Dr_Stat_XXY_BZZZ

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Old Name	New Name
28	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	122	CN PX Bay Z	Dr_Stat_XXY_BZZZ
29	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	123	CN PX Bay Z	Dr_Stat_XXY_BZZZ
30	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	124	CN PX Bay Z	Dr_Stat_XXY_BZZZ
31	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	125	CN PX Bay Z	Dr_Stat_XXY_BZZZ
32	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	126	CN PX Bay Z	Dr_Stat_XXY_BZZZ
33	0x6F Sensor-specific	0x0D Drive Slot	0x04 Disk	127	CN PX Bay Z	Dr_Stat_XXY_BZZZ
34	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	96	CN PX Bay Z	Dr_Stat_XXY_BZZZ
35	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	97	CN PX Bay Z	Dr_Stat_XXY_BZZZ
36	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	98	CN PX Bay Z	Dr_Stat_XXY_BZZZ
37	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	99	CN PX Bay Z	Dr_Stat_XXY_BZZZ
38	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	100	CN PX Bay Z	Dr_Stat_XXY_BZZZ
39	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	101	CN PX Bay Z	Dr_Stat_XXY_BZZZ
40	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	102	CN PX Bay Z	Dr_Stat_XXY_BZZZ
41	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	103	CN PX Bay Z	Dr_Stat_XXY_BZZZ
42	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	104	CN PX Bay Z	Dr_Stat_XXY_BZZZ
43	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	105	CN PX Bay Z	Dr_Stat_XXY_BZZZ
44	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	106	CN PX Bay Z	Dr_Stat_XXY_BZZZ
45	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	107	CN PX Bay Z	Dr_Stat_XXY_BZZZ
46	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	108	CN PX Bay Z	Dr_Stat_XXY_BZZZ
47	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	109	CN PX Bay Z	Dr_Stat_XXY_BZZZ

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Old Name	New Name
48	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	110	CN PX Bay Z	Dr_Stat_XXY_BZZZ
49	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	111	CN PX Bay Z	Dr_Stat_XXY_BZZZ
50	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	112	CN PX Bay Z	Dr_Stat_XXY_BZZZ
51	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	113	CN PX Bay Z	Dr_Stat_XXY_BZZZ
52	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	114	CN PX Bay Z	Dr_Stat_XXY_BZZZ
53	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	115	CN PX Bay Z	Dr_Stat_XXY_BZZZ
54	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	116	CN PX Bay Z	Dr_Stat_XXY_BZZZ
55	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	117	CN PX Bay Z	Dr_Stat_XXY_BZZZ
56	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	118	CN PX Bay Z	Dr_Stat_XXY_BZZZ
57	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	119	CN PX Bay Z	Dr_Stat_XXY_BZZZ
58	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	120	CN PX Bay Z	Dr_Stat_XXY_BZZZ
59	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	121	CN PX Bay Z	Dr_Stat_XXY_BZZZ
60	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	122	CN PX Bay Z	Dr_Stat_XXY_BZZZ
61	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	123	CN PX Bay Z	Dr_Stat_XXY_BZZZ
62	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	124	CN PX Bay Z	Dr_Stat_XXY_BZZZ
63	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	125	CN PX Bay Z	Dr_Stat_XXY_BZZZ
64	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	126	CN PX Bay Z	Dr_Stat_XXY_BZZZ
65	0x6F Sensor-specific	0x0D Drive Slot	0xD0 OEM Disk 1	127	CN PX Bay Z	Dr_Stat_XXY_BZZZ
66	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	96	CN PX Bay Z	Dr_Stat_XXY_BZZZ
67	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	97	CN PX Bay Z	Dr_Stat_XXY_BZZZ

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Old Name	New Name
68	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	98	CN PX Bay Z	Dr_Stat_XXY_BZZZ
69	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	99	CN PX Bay Z	Dr_Stat_XXY_BZZZ
70	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	100	CN PX Bay Z	Dr_Stat_XXY_BZZZ
71	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	101	CN PX Bay Z	Dr_Stat_XXY_BZZZ
72	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	102	CN PX Bay Z	Dr_Stat_XXY_BZZZ
73	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	103	CN PX Bay Z	Dr_Stat_XXY_BZZZ
74	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	104	CN PX Bay Z	Dr_Stat_XXY_BZZZ
75	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	105	CN PX Bay Z	Dr_Stat_XXY_BZZZ
76	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	106	CN PX Bay Z	Dr_Stat_XXY_BZZZ
77	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	107	CN PX Bay Z	Dr_Stat_XXY_BZZZ
78	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	108	CN PX Bay Z	Dr_Stat_XXY_BZZZ
79	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	109	CN PX Bay Z	Dr_Stat_XXY_BZZZ
80	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	110	CN PX Bay Z	Dr_Stat_XXY_BZZZ
81	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	111	CN PX Bay Z	Dr_Stat_XXY_BZZZ
82	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	112	CN PX Bay Z	Dr_Stat_XXY_BZZZ
83	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	113	CN PX Bay Z	Dr_Stat_XXY_BZZZ
84	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	114	CN PX Bay Z	Dr_Stat_XXY_BZZZ
85	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	115	CN PX Bay Z	Dr_Stat_XXY_BZZZ
86	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	116	CN PX Bay Z	Dr_Stat_XXY_BZZZ
87	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	117	CN PX Bay Z	Dr_Stat_XXY_BZZZ

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Old Name	New Name
88	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	118	CN PX Bay Z	Dr_Stat_XXY_BZZZ
89	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	119	CN PX Bay Z	Dr_Stat_XXY_BZZZ
90	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	120	CN PX Bay Z	Dr_Stat_XXY_BZZZ
91	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	121	CN PX Bay Z	Dr_Stat_XXY_BZZZ
92	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	122	CN PX Bay Z	Dr_Stat_XXY_BZZZ
93	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	123	CN PX Bay Z	Dr_Stat_XXY_BZZZ
94	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	124	CN PX Bay Z	Dr_Stat_XXY_BZZZ
95	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	125	CN PX Bay Z	Dr_Stat_XXY_BZZZ
96	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	126	CN PX Bay Z	Dr_Stat_XXY_BZZZ
97	0x6F Sensor-specific	0x0D Drive Slot	0xD1 OEM Disk 2	127	CN PX Bay Z	Dr_Stat_XXY_BZZZ
98	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	96	CN PX Bay Z	Dr_Stat_XXY_BZZZ
99	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	97	CN PX Bay Z	Dr_Stat_XXY_BZZZ
100	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	98	CN PX Bay Z	Dr_Stat_XXY_BZZZ
101	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	99	CN PX Bay Z	Dr_Stat_XXY_BZZZ
102	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	100	CN PX Bay Z	Dr_Stat_XXY_BZZZ
103	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	101	CN PX Bay Z	Dr_Stat_XXY_BZZZ
104	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	102	CN PX Bay Z	Dr_Stat_XXY_BZZZ
105	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	103	CN PX Bay Z	Dr_Stat_XXY_BZZZ
106	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	104	CN PX Bay Z	Dr_Stat_XXY_BZZZ
107	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	105	CN PX Bay Z	Dr_Stat_XXY_BZZZ

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Old Name	New Name
108	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	106	CN PX Bay Z	Dr_Stat_XXY_BZZZ
109	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	107	CN PX Bay Z	Dr_Stat_XXY_BZZZ
110	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	108	CN PX Bay Z	Dr_Stat_XXY_BZZZ
111	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	109	CN PX Bay Z	Dr_Stat_XXY_BZZZ
112	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	110	CN PX Bay Z	Dr_Stat_XXY_BZZZ
113	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	111	CN PX Bay Z	Dr_Stat_XXY_BZZZ
114	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	112	CN PX Bay Z	Dr_Stat_XXY_BZZZ
115	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	113	CN PX Bay Z	Dr_Stat_XXY_BZZZ
116	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	114	CN PX Bay Z	Dr_Stat_XXY_BZZZ
117	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	115	CN PX Bay Z	Dr_Stat_XXY_BZZZ
118	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	116	CN PX Bay Z	Dr_Stat_XXY_BZZZ
119	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	117	CN PX Bay Z	Dr_Stat_XXY_BZZZ
120	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	118	CN PX Bay Z	Dr_Stat_XXY_BZZZ
121	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	119	CN PX Bay Z	Dr_Stat_XXY_BZZZ
122	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	120	CN PX Bay Z	Dr_Stat_XXY_BZZZ
123	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	121	CN PX Bay Z	Dr_Stat_XXY_BZZZ
124	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	122	CN PX Bay Z	Dr_Stat_XXY_BZZZ
125	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	123	CN PX Bay Z	Dr_Stat_XXY_BZZZ
126	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	124	CN PX Bay Z	Dr_Stat_XXY_BZZZ
127	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	125	CN PX Bay Z	Dr_Stat_XXY_BZZZ

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Old Name	New Name
128	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	126	CN PX Bay Z	Dr_Stat_XXY_BZZZ
129	0x6F Sensor-specific	0x0D Drive Slot	0xD2 OEM Disk 3	127	CN PX Bay Z	Dr_Stat_XXY_BZZZ
130	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	96	CN PX Bay Z	Dr_Stat_XXY_BZZZ
131	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	97	CN PX Bay Z	Dr_Stat_XXY_BZZZ
132	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	98	CN PX Bay Z	Dr_Stat_XXY_BZZZ
133	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	99	CN PX Bay Z	Dr_Stat_XXY_BZZZ
134	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	100	CN PX Bay Z	Dr_Stat_XXY_BZZZ
135	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	101	CN PX Bay Z	Dr_Stat_XXY_BZZZ
136	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	102	CN PX Bay Z	Dr_Stat_XXY_BZZZ
137	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	103	CN PX Bay Z	Dr_Stat_XXY_BZZZ
138	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	104	CN PX Bay Z	Dr_Stat_XXY_BZZZ
139	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	105	CN PX Bay Z	Dr_Stat_XXY_BZZZ
140	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	106	CN PX Bay Z	Dr_Stat_XXY_BZZZ
141	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	107	CN PX Bay Z	Dr_Stat_XXY_BZZZ
142	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	108	CN PX Bay Z	Dr_Stat_XXY_BZZZ
143	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	109	CN PX Bay Z	Dr_Stat_XXY_BZZZ
144	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	110	CN PX Bay Z	Dr_Stat_XXY_BZZZ
145	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	111	CN PX Bay Z	Dr_Stat_XXY_BZZZ
146	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	112	CN PX Bay Z	Dr_Stat_XXY_BZZZ
147	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	113	CN PX Bay Z	Dr_Stat_XXY_BZZZ

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Old Name	New Name
148	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	114	CN PX Bay Z	Dr_Stat_XXY_BZZZ
149	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	115	CN PX Bay Z	Dr_Stat_XXY_BZZZ
150	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	116	CN PX Bay Z	Dr_Stat_XXY_BZZZ
151	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	117	CN PX Bay Z	Dr_Stat_XXY_BZZZ
152	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	118	CN PX Bay Z	Dr_Stat_XXY_BZZZ
153	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	119	CN PX Bay Z	Dr_Stat_XXY_BZZZ
154	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	120	CN PX Bay Z	Dr_Stat_XXY_BZZZ
155	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	121	CN PX Bay Z	Dr_Stat_XXY_BZZZ
156	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	122	CN PX Bay Z	Dr_Stat_XXY_BZZZ
157	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	123	CN PX Bay Z	Dr_Stat_XXY_BZZZ
158	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	124	CN PX Bay Z	Dr_Stat_XXY_BZZZ
159	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	125	CN PX Bay Z	Dr_Stat_XXY_BZZZ
160	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	126	CN PX Bay Z	Dr_Stat_XXY_BZZZ
161	0x6F Sensor-specific	0x0D Drive Slot	0xD3 OEM Disk 4	127	CN PX Bay Z	Dr_Stat_XXY_BZZZ
162	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	96	CN PX Bay Z	Dr_Stat_XXY_BZZZ
163	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	97	CN PX Bay Z	Dr_Stat_XXY_BZZZ
164	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	98	CN PX Bay Z	Dr_Stat_XXY_BZZZ
165	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	99	CN PX Bay Z	Dr_Stat_XXY_BZZZ
166	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	100	CN PX Bay Z	Dr_Stat_XXY_BZZZ
167	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	101	CN PX Bay Z	Dr_Stat_XXY_BZZZ

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Old Name	New Name
168	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	102	CN PX Bay Z	Dr_Stat_XXY_BZZZ
169	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	103	CN PX Bay Z	Dr_Stat_XXY_BZZZ
170	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	104	CN PX Bay Z	Dr_Stat_XXY_BZZZ
171	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	105	CN PX Bay Z	Dr_Stat_XXY_BZZZ
172	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	106	CN PX Bay Z	Dr_Stat_XXY_BZZZ
173	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	107	CN PX Bay Z	Dr_Stat_XXY_BZZZ
174	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	108	CN PX Bay Z	Dr_Stat_XXY_BZZZ
175	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	109	CN PX Bay Z	Dr_Stat_XXY_BZZZ
176	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	110	CN PX Bay Z	Dr_Stat_XXY_BZZZ
177	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	111	CN PX Bay Z	Dr_Stat_XXY_BZZZ
178	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	112	CN PX Bay Z	Dr_Stat_XXY_BZZZ
179	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	113	CN PX Bay Z	Dr_Stat_XXY_BZZZ
180	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	114	CN PX Bay Z	Dr_Stat_XXY_BZZZ
181	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	115	CN PX Bay Z	Dr_Stat_XXY_BZZZ
182	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	116	CN PX Bay Z	Dr_Stat_XXY_BZZZ
183	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	117	CN PX Bay Z	Dr_Stat_XXY_BZZZ
184	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	118	CN PX Bay Z	Dr_Stat_XXY_BZZZ
185	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	119	CN PX Bay Z	Dr_Stat_XXY_BZZZ
186	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	120	CN PX Bay Z	Dr_Stat_XXY_BZZZ
187	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	121	CN PX Bay Z	Dr_Stat_XXY_BZZZ

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Old Name	New Name
188	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	122	CN PX Bay Z	Dr_Stat_XXY_BZZZ
189	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	123	CN PX Bay Z	Dr_Stat_XXY_BZZZ
190	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	124	CN PX Bay Z	Dr_Stat_XXY_BZZZ
191	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	125	CN PX Bay Z	Dr_Stat_XXY_BZZZ
192	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	126	CN PX Bay Z	Dr_Stat_XXY_BZZZ
193	0x6F Sensor-specific	0x0D Drive Slot	0xD4 OEM Disk 5	127	CN PX Bay Z	Dr_Stat_XXY_BZZZ
194	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	96	CN PX Bay Z	Dr_Stat_XXY_BZZZ
195	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	97	CN PX Bay Z	Dr_Stat_XXY_BZZZ
196	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	98	CN PX Bay Z	Dr_Stat_XXY_BZZZ
197	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	99	CN PX Bay Z	Dr_Stat_XXY_BZZZ
198	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	100	CN PX Bay Z	Dr_Stat_XXY_BZZZ
199	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	101	CN PX Bay Z	Dr_Stat_XXY_BZZZ
200	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	102	CN PX Bay Z	Dr_Stat_XXY_BZZZ
201	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	103	CN PX Bay Z	Dr_Stat_XXY_BZZZ
202	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	104	CN PX Bay Z	Dr_Stat_XXY_BZZZ
203	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	105	CN PX Bay Z	Dr_Stat_XXY_BZZZ
204	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	106	CN PX Bay Z	Dr_Stat_XXY_BZZZ
205	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	107	CN PX Bay Z	Dr_Stat_XXY_BZZZ
206	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	108	CN PX Bay Z	Dr_Stat_XXY_BZZZ
207	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	109	CN PX Bay Z	Dr_Stat_XXY_BZZZ

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Old Name	New Name
208	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	110	CN PX Bay Z	Dr_Stat_XXY_BZZZ
209	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	111	CN PX Bay Z	Dr_Stat_XXY_BZZZ
210	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	112	CN PX Bay Z	Dr_Stat_XXY_BZZZ
211	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	113	CN PX Bay Z	Dr_Stat_XXY_BZZZ
212	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	114	CN PX Bay Z	Dr_Stat_XXY_BZZZ
213	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	115	CN PX Bay Z	Dr_Stat_XXY_BZZZ
214	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	116	CN PX Bay Z	Dr_Stat_XXY_BZZZ
215	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	117	CN PX Bay Z	Dr_Stat_XXY_BZZZ
216	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	118	CN PX Bay Z	Dr_Stat_XXY_BZZZ
217	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	119	CN PX Bay Z	Dr_Stat_XXY_BZZZ
218	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	120	CN PX Bay Z	Dr_Stat_XXY_BZZZ
219	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	121	CN PX Bay Z	Dr_Stat_XXY_BZZZ
220	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	122	CN PX Bay Z	Dr_Stat_XXY_BZZZ
221	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	123	CN PX Bay Z	Dr_Stat_XXY_BZZZ
222	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	124	CN PX Bay Z	Dr_Stat_XXY_BZZZ
223	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	125	CN PX Bay Z	Dr_Stat_XXY_BZZZ
224	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	126	CN PX Bay Z	Dr_Stat_XXY_BZZZ
225	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	127	CN PX Bay Z	Dr_Stat_XXY_BZZZ
226	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	96	CN PX Bay Z	Dr_Stat_XXY_BZZZ
227	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	97	CN PX Bay Z	Dr_Stat_XXY_BZZZ

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Old Name	New Name
228	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	98	CN PX Bay Z	Dr_Stat_XXY_BZZZ
229	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	99	CN PX Bay Z	Dr_Stat_XXY_BZZZ
230	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	100	CN PX Bay Z	Dr_Stat_XXY_BZZZ
231	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	101	CN PX Bay Z	Dr_Stat_XXY_BZZZ
232	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	102	CN PX Bay Z	Dr_Stat_XXY_BZZZ
233	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	103	CN PX Bay Z	Dr_Stat_XXY_BZZZ
234	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	104	CN PX Bay Z	Dr_Stat_XXY_BZZZ
235	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	105	CN PX Bay Z	Dr_Stat_XXY_BZZZ
236	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	106	CN PX Bay Z	Dr_Stat_XXY_BZZZ
237	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	107	CN PX Bay Z	Dr_Stat_XXY_BZZZ
238	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	108	CN PX Bay Z	Dr_Stat_XXY_BZZZ
239	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	109	CN PX Bay Z	Dr_Stat_XXY_BZZZ
240	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	110	CN PX Bay Z	Dr_Stat_XXY_BZZZ
241	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	111	CN PX Bay Z	Dr_Stat_XXY_BZZZ
242	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	112	CN PX Bay Z	Dr_Stat_XXY_BZZZ
243	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	113	CN PX Bay Z	Dr_Stat_XXY_BZZZ
244	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	114	CN PX Bay Z	Dr_Stat_XXY_BZZZ
245	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	115	CN PX Bay Z	Dr_Stat_XXY_BZZZ
246	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	116	CN PX Bay Z	Dr_Stat_XXY_BZZZ
247	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	117	CN PX Bay Z	Dr_Stat_XXY_BZZZ

Table Continued

Sl. No	ERC	Sensor Type	Entity ID	Entity Instance	Old Name	New Name
248	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	118	CN PX Bay Z	Dr_Stat_XXY_BZZZ
249	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	119	CN PX Bay Z	Dr_Stat_XXY_BZZZ
250	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	120	CN PX Bay Z	Dr_Stat_XXY_BZZZ
251	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	121	CN PX Bay Z	Dr_Stat_XXY_BZZZ
252	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	122	CN PX Bay Z	Dr_Stat_XXY_BZZZ
253	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	123	CN PX Bay Z	Dr_Stat_XXY_BZZZ
254	0x6F Sensor-specific	0x0D Drive Slot	0xD5 OEM Disk 6	124	CN PX Bay Z	Dr_Stat_XXY_BZZZ

Intrusion Sensor

Intrusion detection kits are introduced with Gen 10 servers. For servers prior to Gen 10, intrusion sensor information are not displayed. Intrusion sensor will not be listed for Blade and Synergy platforms.

Supported system without intrusion detection kit

When the intrusion detection kit is not installed on a supported system, the status of the intrusion sensor in the output of the ipmitool command with `sdr list all` will display the status as **Not Readable** and the `sdr elist all` will display the status as **Disabled**. When invoked with `sensor list`, the 2nd and 4th column will display **na**.

For example:

`ipmitool sdr list all`

```
Intrusion        | Not Readable      | ns
```

`ipmitool sdr elist all`

```
Intrusion        | ABh | ns  | 23.1 | Disabled
```

`ipmitool sensor list`

```
Intrusion        | na        | discrete   | na    | na        | na        | na        | na        | na        | na
```

Supported system with intrusion detection kit

For supported platforms, if the intrusion sensor is installed and the ipmitool is invoked with the `sdr elist all`, the status for the intrusion will be blank, that is, the 5th column does not display any value when no Intrusion is detected.

When invoked with the `sensor list`, the status **0x0080** indicates **OK** and the status **0x0180** indicates **General Chassis Intrusion**, (which means the access panel is open) in the 4th column.

For example:

Normal scenario:

`ipmitool sdr list all`

```
Intrusion     | 0x00              | ok
```

ipmitool sdr elist all

```
# Intrusion         | ABh | ok   | 23.1 |
```

ipmitool sensor list

```
# Intrusion        | 0x0        | discrete   | 0x0080| na        | na        | na        | na        | na        | na
```

Intrusion detected scenario:

ipmitool sdr list all

```
Intrusion     | 0x00              | ok
```

ipmitool sdr elist all

```
# Intrusion         | ABh | ok   | 23.1 | General Chassis intrusion
```

ipmitool sensor list

```
# Intrusion        | 0x0        | discrete   | 0x0180| na        | na        | na        | na        | na        | na
```

Events

Certain HPE server events may need to contain more information than originally provided for by the IPMI specification. This is specifically true for MCE and PCIe events which are also reported through the HPE iLO IML. As a solution, HPE provides multiple OEM IPMI event logs in sequence for a single event.

BMC LUN 0 SDRR

Sensor Name	Description	Owner ID	ERC	Sensor Type	Notes
XX-CPU X	CPU upper critical Temps - 0.25 DTS Temps above this threshold for >=60S will cause soft shutdown.	0x20	0x01 Threshold	0x01 Temperature	
XX-DIMM PX X-X	Memory upper critical Temps 87C Temps above/= this threshold for >=60S will cause soft shutdown.	0x20	0x01 Threshold	0x01 Temperature	
XX-XXXXX	Various components have both upper critical and upper non-recoverable thresholds. Temps above upper critical threshold >= 60S will have soft shutdown. Temps above upper-nonrecoverable have forced shutdown.	0x20	0x01 Threshold	0x01 Threshold	
CPU_THERM_TRIP[1]	Thermal Trip event of CPU	0x20	0x6F Sensor-specific	0x07 Processor	
SYS_THERM_TRIP[1]	Thermal Trip event of other system components	0x20	0x07 severity	0x15 Board	
Memory_Error	BIOS memory events	0x01	0x6F Sensor-specific	0x0C Memory	Includes slot number in event data byte 3 if obtained from BIOS
CPUX_ERROR[1]	BIOS processor events	0x01	0x6F Sensor-specific	0x07 Processor	
	OEM MCE Event[2]	0x01			Machine check errors

Table Continued

Sensor Name	Description	Owner ID	ERC	Sensor Type	Notes
PCI-E Bus Error[1]	BIOS PCIe error event	0x01	0x6F Sensor-specific	0x13 Critical Interrupt	
	OEM PCIe error event[3]	0x01	0x6F Sensor-specific	0x13 Critical Interrupt	PCI express errors
SYS_PWR_FAULT[1]	System Power Fault	0x20	0x6F Sensor-specific	0x09 Power Unit	
OS_STOP_SHUTDN[1]	OS system crash event	0x01	0x6F Sensor-specific	0x20 OS Stop/ Shutdown	
OS_NMI[1]	NMI events	0x01	0x6F Sensor-specific	0x13 Critical Interrupt	
ACPI[1]	ACPI events	0x20	0x6F Sensor-specific	0x22 ACPI Power State	
CPU_IERR[1]	CATERR hold assertion	0x20	0x6F Sensor-specific	0x07 Processor	
	IPMI watchdog event	0x20	0x6F Sensor-specific	0x12 System Event	
	IPMI watchdog event	0x20	0x6F Sensor-specific	0x23 Watchdog 2	

1 Event only

2 **MCE Event Logs**

3 **PCI-E Event Logs**

Chassis MC LUN 0 SDRR

Sensor Name	Description	Owner ID	Event Only	ERC	Sensor Type
Fan X	Discrete fan sensors	0x44		0x07 Severity	0x04 Fan
Fans	Redundancy fan sensor	0x44		0x0B Redundancy	0x04 Fan
Power Supply X	Power supply status events	0x44		0x06 Sensor-specific	0x08 Power Supply
Power Supplies	Power supply redundancy status	0x44		0x0B Redundancy	0x08 Power Supply

MCE event logs

MCE event logs are broken into 10 separate IPMI log files. Use the complete log set to generate any one event report.

- Bytes 1-11: These bytes are the same for one event. They provide identifying information.
- Byte 12: Provides the sequence number of the event log, 1-10, that when combined, make up one event.
- Bytes 13-16: Provide specific information for the event.

Table 165: Shared Bytes

Byte	Field	Description
1:2	Record ID	SEL Record ID
3	Record Type	OEM System Event Record: 0xC1 (IML)
4:7	Timestamp	Time stamp
8:10	ManufacturerID	IANA Enterprise ID
11	IML Event Number	Wrapping counter for OEM IML Type events

Table 166: Unique IDs - Event Log 1

Byte	Field	Description
12	Sequence Number	Wrapping counter for sequence within event number = 1
13:14	Data Length	Total length of data
15:16	IML Class	0x05 CPU

Table 167: Unique IDs - Event Log 2

Byte	Field	Description
12	Sequence Number	Wrapping counter for sequence within event number = 2
13:14	IML Code	0x03 = Uncorrectable Machine Check Exception
15	Data (0)	Board Number
16	Data (1)	Processor Number

Table 168: Unique IDs - Event Log 3

Byte	Field	Description
12	Sequence Number	Wrapping counter for sequence within event number = 3
13:16	Data [2:5]	APIC ID

Table 169: Unique IDs - Event Log 4

Byte	Field	Description
12	Sequence Number	Wrapping counter for sequence within event number = 4
13:16	Data [6:9]	MCA bank

Table 170: Unique IDs - Event Log 5

Byte	Field	Description
12	Sequence Number	Wrapping counter for sequence within event number = 5
13:16	Data [10:13]	Mci_STATUS_MSR (lower 4 bytes)

Table 171: Unique IDs - Event Log 6

Byte	Field	Description
12	Sequence Number	Wrapping counter for sequence within event number = 6
13:16	Data [14:17]	Mci_STATUS_MSR (upper 4 bytes)

Table 172: Unique IDs - Event Log 7

Byte	Field	Description
12	Sequence Number	Wrapping counter for sequence within event number = 7
13:16	Data [18:21]	Mci_ADDR MSR (lower 4 bytes)

Table 173: Unique IDs - Event Log 8

Byte	Field	Description
12	Sequence Number	Wrapping counter for sequence within event number = 8
13:16	Data [22:25]	Mci_ADDR MSR (upper 4 bytes)

Table 174: Unique IDs - Event Log 9

Byte	Field	Description
12	Sequence Number	Wrapping counter for sequence within event number = 9
13:16	Data [26:29]	Mci_MISC MSR (lower 4 bytes)

Table 175: Unique IDs - Event Log 10

Byte	Field	Description
12	Sequence Number	Wrapping counter for sequence within event number = 10
13:16	Data [26:29]	Mci_MISC MSR (upper 4 bytes)

PCI-E event logs

PCI-E event logs are broken into four separate IPMI log files. Use the complete log set to generate any one event report.

- Bytes 1-11: These bytes are the same for every event. They provide identifying information.
- Byte 12: Provides the sequence number of the IPMI event logs, 1-4, that when combined, make up one event.
- Bytes 13-16: Provide specific information for the event.

Table 176: Shared Bytes

Byte	Field	Description
1:2	Record ID	SEL Record ID
3	Record Type	OEM System Event Record: 0xC1 (IML)
4:7	Timestamp	Time stamp
8:10	ManufacturerID	IANA Enterprise ID
11	IML Event Number	Wrapping counter for OEM IML Type events

Table 177: Unique IDs - Event Log 1

Byte	Field	Description
12	Sequence Number	Wrapping counter for sequence within event number = 1
13:14	Data Length	Total length of data
15:16	IML Class	0x08 = PCI Bus

Table 178: Unique IDs - Event Log 2

Byte	Field	Description
12	Sequence Number	Wrapping counter for sequence within event number = 2
13:14	IML Code	0x02 = PCI Express Error
15	Data (0)	Slot Number
16	Data (1)	Bus Number

Table 179: Unique IDs - Event Log 3

Byte	Field	Description
12	Sequence Number	Wrapping counter for sequence within event number = 3
13	Data [2]	Device Number
14	Data [3]	Function Number
15:16	Data [4:5]	Error Status Register (lower 2 bytes)

Table 180: Unique IDs - Event Log 4

Byte	Field	Description
12	Sequence Number	Wrapping counter for sequence within event number = 4
13:14	Data [6:7]	Error Status Register (upper 2 bytes)
15:16	Data [8:9]	Padding (0s [zeros])

Glossary

ACPI

Advanced Configuration and Power Interface Specification

BCD

Binary-coded Decimal

BMC

Baseboard Management Controller

BT

Block Transfer

ChMC

Chassis Management Controller

CMOS

The PC-AT compatible region of battery-backed 128 bytes of memory, which normally resides on the baseboard.

CTS

Clear to send

DCD

Data Carrier Detect

DCMI

Data Center Manageability Interface

DSR

Data Set Ready

DTR

Data Transfer Request

EvMRev

Event message revision

FPGA

Field-Programmable Gate Array

FRB

Fault-resilient booting

FRU

Field replaceable unit

GUID

Globally Unique ID

HA

High availability

I²C

Inter-Integrated Circuit

IANA

Internet Assigned Numbers Authority

ICMB

Intelligent Chassis Management Bus

IERR

Internal error

IPMB

Intelligent Platform Management Bus

PICMG

PCI Industrial Computer Manufacturers Group

IPMI

Intelligent Platform Management Interface

KCS

Keyboard Controller Style

LPC

Low Pin Count

LS

Least significant byte

MC

Management Controller

MS

Most significant byte

mux

Multiplexing

NetFn

Network Function

NACK

Negative acknowledgment, Not acknowledged

NAK

Negative-acknowledge character

NMI

Nonmaskable Interrupt

PCI

Peripheral Component Interconnect

PEF

Platform Event Filtering

POH

Power-On Hours

PSMC

Power Supply Management Controller

RAKP

Remote Authenticated Key-Exchange Protocol

RMCP

Remote Management Control Protocol

RQ

Received request

RS

Received response

rsSA

Random Single Switch Algorithm

RTS

Request to send

SCI

Software Configuration Identification

SDR

Sensor Data Record

SEL

System Event Log

SMB

System Management Bus

SMI

System Management Interrupt

SMIC

System Management Interface Chip

SMM

System Management Mode.

SMS

System Management Software

SOL

Serial Over LAN

SSI

Server System Infrastructure

SSIF

SMBus System Interface

SWID

Software ID

TAP

Telocator Access Protocol

UUID

Universally Unique IDentifier

VSO

VITA Standards Organization

Websites

iLO

https://www.hpe.com/info/ilo

iLO 5 documentation

https://www.hpe.com/support/ilo-docs

iLO support

https://www.hpe.com/support/ilo5

iLO helpful links and resources

https://www.hpe.com/support/ilo-resource-ref-en

HPE iLO Free Online Training

https://www.hpe.com/ww/iloBundle

HPE ProLiant Gen10 servers

https://www.hpe.com/info/proliantgen10-docs

HPE ProLiant Gen10 Plus servers

https://www.hpe.com/info/proliantgen10plus-docs

HPE ProLiant training

https://www.hpe.com/ww/learnproliant

HPE Synergy

https://www.hpe.com/info/synergy-docs

Apollo systems and APM

Navigate to **https://www.hpe.com/info/docs** and then select an Apollo System in the **Products & Solutions** section.

UEFI System Utilities

https://www.hpe.com/info/ProLiantUEFI/docs

SUM

https://www.hpe.com/info/sum-docs

SPP

https://www.hpe.com/info/spp/documentation

Intelligent Provisioning

https://www.hpe.com/info/intelligentprovisioning/docs

iLO RESTful API and RESTful Interface Tool

https://www.hpe.com/support/restfulinterface/docs

Remote Support

https://www.hpe.com/info/insightremotesupport/docs

HPE InfoSight for Servers

https://www.hpe.com/servers/infosight

iLO Amplifier Pack

https://www.hpe.com/servers/iloamplifierpack

HPE OneView

https://www.hpe.com/info/oneview/docs

HPE SIM

https://www.hpe.com/info/insightmanagement/sim/docs

OA

http://www.hpe.com/support/BladeSystem/docs

Support and other resources

Accessing Hewlett Packard Enterprise Support

- For live assistance, go to the Contact Hewlett Packard Enterprise Worldwide website:

  https://www.hpe.com/info/assistance

- To access documentation and support services, go to the Hewlett Packard Enterprise Support Center website:

  https://www.hpe.com/support/hpesc

Information to collect

- Technical support registration number (if applicable)
- Product name, model or version, and serial number
- Operating system name and version
- Firmware version
- Error messages
- Product-specific reports and logs
- Add-on products or components
- Third-party products or components

Accessing updates

- Some software products provide a mechanism for accessing software updates through the product interface. Review your product documentation to identify the recommended software update method.
- To download product updates:

  Hewlett Packard Enterprise Support Center

  https://www.hpe.com/support/hpesc

  Hewlett Packard Enterprise Support Center: Software downloads

  https://www.hpe.com/support/downloads

  My HPE Software Center

  https://www.hpe.com/software/hpesoftwarecenter

- To subscribe to eNewsletters and alerts:

  https://www.hpe.com/support/e-updates

- To view and update your entitlements, and to link your contracts and warranties with your profile, go to the Hewlett Packard Enterprise Support Center **More Information on Access to Support Materials** page:

  https://www.hpe.com/support/AccessToSupportMaterials

IMPORTANT: Access to some updates might require product entitlement when accessed through the Hewlett Packard Enterprise Support Center. You must have an HPE Passport set up with relevant entitlements.

Remote support

Remote support is available with supported devices as part of your warranty or contractual support agreement. It provides intelligent event diagnosis, and automatic, secure submission of hardware event notifications to Hewlett Packard Enterprise, which initiates a fast and accurate resolution based on the service level of your product. Hewlett Packard Enterprise strongly recommends that you register your device for remote support.

If your product includes additional remote support details, use search to locate that information.

HPE Get Connected

https://www.hpe.com/services/getconnected

HPE Pointnext Tech Care

https://www.hpe.com/services/techcare

HPE Complete Care

https://www.hpe.com/services/completecare

Warranty information

To view the warranty information for your product, see the links provided below:

HPE ProLiant and IA-32 Servers and Options

https://www.hpe.com/support/ProLiantServers-Warranties

HPE Enterprise and Cloudline Servers

https://www.hpe.com/support/EnterpriseServers-Warranties

HPE Storage Products

https://www.hpe.com/support/Storage-Warranties

HPE Networking Products

https://www.hpe.com/support/Networking-Warranties

Regulatory information

To view the regulatory information for your product, view the *Safety and Compliance Information for Server, Storage, Power, Networking, and Rack Products*, available at the Hewlett Packard Enterprise Support Center:

https://www.hpe.com/support/Safety-Compliance-EnterpriseProducts

Additional regulatory information

Hewlett Packard Enterprise is committed to providing our customers with information about the chemical substances in our products as needed to comply with legal requirements such as REACH (Regulation EC No 1907/2006 of the European Parliament and the Council). A chemical information report for this product can be found at:

https://www.hpe.com/info/reach

For Hewlett Packard Enterprise product environmental and safety information and compliance data, including RoHS and REACH, see:

https://www.hpe.com/info/ecodata

For Hewlett Packard Enterprise environmental information, including company programs, product recycling, and energy efficiency, see:

https://www.hpe.com/info/environment

Documentation feedback

Hewlett Packard Enterprise is committed to providing documentation that meets your needs. To help us improve the documentation, use the **Feedback** button and icons (located at the bottom of an opened document) on the Hewlett Packard Enterprise Support Center portal (**https://www.hpe.com/support/hpesc**) to send any errors, suggestions, or comments. All document information is captured by the process.