

```
ipmitool -I lan -H ipaddress
:ername raw netfn cmd evmrev
eventdir/eventtype eventdata
```

```
import os
load = open('/proc/loadavg').
loadAvg = [hex(ord(i)) for i
cmd = 'ipmitool raw 0x6 0x58
os.system(cmd)
os.system("ipmitool raw 0x6 0
```

USING IPMITOOL RAW COMMANDS FOR REMOTE MANAGEMENT OF DELL POWEREDGE SERVERS

BY SESHADRI N.
RAGHAVENDRA BABU

Although IPMItool provides a useful way to monitor and configure Intelligent Platform Management Interface (IPMI)-compliant devices, its limited number of command-line options restricts access to some capabilities of the baseboard management controllers in Dell™ PowerEdge™ servers. To help maximize these capabilities, administrators can instead access hardware functions using IPMItool raw commands.



The Intelligent Platform Management Interface (IPMI) specification defines standards for monitoring server hardware characteristics such as system temperatures, voltages, fans, power supplies, bus errors, and physical system security; logging abnormal conditions; generating platform alerts; and providing inventory information. IPMItool, an open source utility for managing and configuring IPMI-compliant devices, enables administrators to monitor, log, recover, inventory, and control hardware in Dell PowerEdge servers through a simple baseboard management controller (BMC) command-line interface (CLI). However, the number of CLI options limits the BMC features available through this interface. To help maximize the available BMC capabilities, administrators can instead access hardware functions using raw commands.

This article provides step-by-step guidance on configuring IPMI, installing and using IPMItool, enabling and using Serial Over LAN (SOL), and using IPMItool raw commands. The instructions in this article are based on the IPMI 2.0 specification available at developer.intel.com/design/servers/ipmi/spec.htm, and are designed to apply to Dell PowerEdge 1950 and PowerEdge 2950 servers running the Red Hat® Enterprise Linux® OS. Other configurations may require administrators to modify these instructions.

Configuring Intelligent Platform Management Interface

Administrators can use IPMItool to manage IPMI-based servers either locally (within the servers themselves) or remotely (from another computer on which IPMItool is installed). Local management requires that OpenIPMI device drivers be installed on the servers. Remote management does not require these drivers, but does require that IPMI Over LAN be set up using the Remote Access Configuration Utility on Dell PowerEdge servers.

Installing OpenIPMI drivers for local management

The OpenIPMI drivers provide an open source IPMI library for Linux operating systems, and are typically included in major Linux distributions. If the drivers are not installed, administrators can download the latest versions from sourceforge.net/projects/openipmi or linux.dell.com/files/openipmi. Before installation, they should first flash the latest versions of the BIOS and BMC firmware (available from support.dell.com). Then, if they are using a .tar.gz driver file, they can extract the file contents using the command `tar zxvf filename.tar.gz`, and install it by executing `install.sh` or by using the following commands:

Related Categories:

Baseboard management controller (BMC)

Dell PowerEdge servers

Intelligent Platform Management Interface (IPMI)

Systems management

Visit DELL.COM/PowerSolutions for the complete category index.

```
sh ./configure
make
make install
```

If they are using a Red Hat Package Manager (RPM™) driver file, they can install it using the command `rpm -ivh openipmi.rpm`.

Administrators can then enable the OpenIPMI drivers as follows:

1. Add the IPMI modules to the Linux kernel using the following commands:

```
modprobe ipmi_msghandler
modprobe ipmi_devintf
```

2. Add the additional necessary modules using the command `modprobe ipmi_kcs_drv` (for kernel version 2.4) or `modprobe ipmi_si` (for kernel version 2.6).¹
3. Create the IPMI character device (if it has not already been created) using the following commands:

```
majorNo=`cat /proc/devices | grep ipmi | cut
-f 1 -d ' '`
mknod -m 0600 /dev/ipmi0 c ${majorNo} 0
ln -sf /dev/ipmi0 /dev/ipmi
```

4. Check that the required modules are loaded by using the command `lsmod | grep ipmi` to verify the presence of the IPMI drivers. IPMI functionality can also be verified using the command `dmidecode` and searching for “IPMI.” The output should be similar to the following:²

```
Handle 0x2600, DMI type 38, 18 bytes.
IPMI Device Information
  Interface Type: KCS (Keyboard Control Style)
  Specification Version: 2.0
  I2C Slave Address: 0x10
  NV Storage Device: Not Present
  Base Address: 0x0000000000000CA8 (I/O)
  Register Spacing: 32-bit Boundaries
```

Administrators can later disable the OpenIPMI drivers, if desired, as follows:

1. Remove the IPMI modules from the Linux kernel using the following commands:

```
rmmod ipmi_msghandler
rmmod ipmi_devintf
```

2. Remove the other modules using the command `rmmod ipmi_kcs_drv` (for kernel version 2.4) or `rmmod ipmi_si` (for kernel version 2.6).
3. Remove the IPMI character device using the command `rm -f /dev/ipmi0`.
4. Check that the modules have been removed by using the command `lsmod | grep ipmi` and verifying that the IPMI drivers do not appear.

Setting up IPMI Over LAN for remote management

Administrators can set up IPMI Over LAN using the Remote Access Configuration Utility, which resides in the BIOS of Dell PowerEdge servers and provides a simple user interface engine for configuring BMCs and Dell Remote Access Controllers (DRACs).³ To do so, they can first launch this utility during boot by pressing Ctrl+E when prompted, then perform the following steps:

1. Set the IPMI Over LAN option to On and the NIC Selection option to Shared, Failover, or Dedicated (see Figure 1).
2. In the LAN Parameters options, set the IP Address Source parameter to either Static or DHCP, and set the VLAN Enable parameter to Off (see Figure 2).

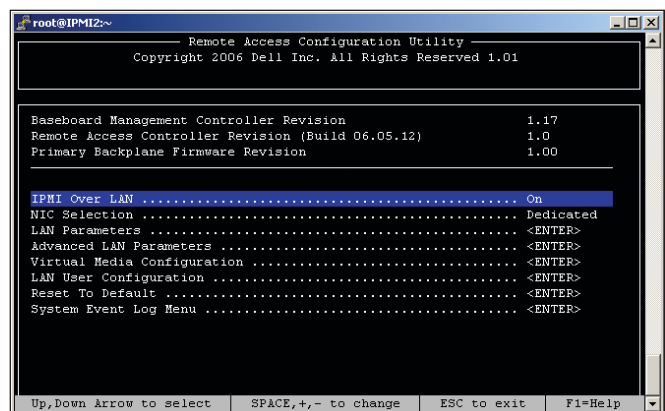


Figure 1. Remote Access Configuration Utility for ninth-generation Dell PowerEdge servers

¹For more details on these modules, see the IPMITool man page at ipmitool.sourceforge.net/manpage.html.

²If using the `dmidecode` command on eighth-generation Dell PowerEdge servers (such as the PowerEdge 1800 or PowerEdge 1850), the specification version would be 1.5. For previous-generation servers, the specification version would be 1.0 and the interface type would be shown as SMIC (Server Management Interface Chip).

³For more information on this utility, see “Exploring the Remote Access Configuration Utility in Ninth-Generation Dell PowerEdge Servers,” by Kalyani Khobragade, in *Dell Power Solutions*, February 2007, DELL.COM/downloads/global/power/ps1q07-20060359-Khobragade.pdf.

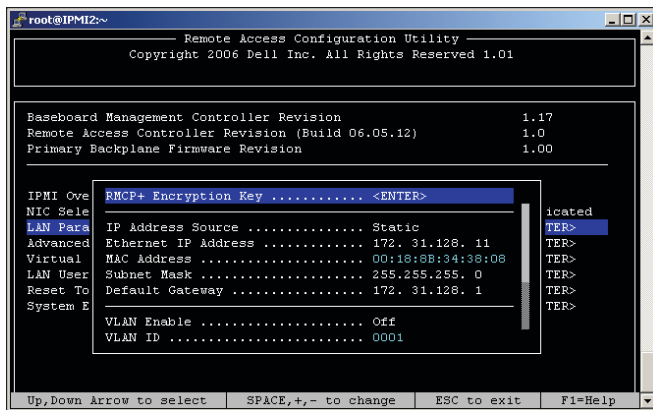


Figure 2. LAN Parameters options in the Remote Access Configuration Utility

- In the LAN User Configuration options, set the Account Access parameter to Enabled and the Account Privilege parameter to Admin, and enter a password (see Figure 3).
- Press the Esc key, save changes, and exit the utility.

Installing and using IPMITool

Administrators can download the latest version of IPMITool from sourceforge.net/projects/ipmitool or linux.dell.com/files/openipmi/ipmitool. They can then install it using the command `rpm -ivh packagename`.

IPMITool provides interfaces for both local and remote server management. For local management, it uses the Keyboard Controller Style (KCS) interface, often called the open interface, which requires the OpenIPMI kernel driver. Administrators can set IPMITool to use this interface by including the `-I open` option in their commands:

```
ipmitool -I open command
```

For remote management, IPMITool uses either the lan or lanplus interface. These interfaces are used to communicate with BMCs over an Ethernet LAN connection using User Datagram Protocol (UDP) under IP version 4 (IPv4). These UDP packets are formatted to contain IPMI request/response messages with IPMI session headers and Remote Management and Control Protocol (RMCP) headers, which support management in environments without an OS. The difference between the lan and lanplus interfaces is that lanplus uses RMCP+, providing enhanced authentication and data integration checks.

The format for using commands with these interfaces is as follows:

```
ipmitool -I interface -H ipaddress -U username
-P password command
```

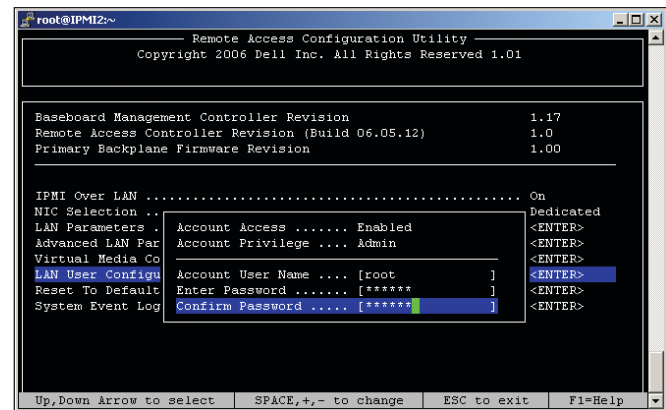


Figure 3. LAN User Configuration options in the Remote Access Configuration Utility

In this command, *interface* is `lan` or `lanplus`, *ipaddress* is the IP address of the remote BMC, *username* and *password* are the administrative username and password, and *command* is the command the administrator wants to run. If the password is set as the `IPMI_PASSWORD` environment variable, the command format changes to the following:

```
ipmitool -I interface -H ipaddress -U username
-E command
```

Administrators can get help with a specific command using the following format:

```
ipmitool -I interface -H ipaddress -U username
-P password command help
```

For example, if during the Remote Access Configuration Utility setup the username was defined as “root” and the password was defined as “calvin” (see Figure 3), administrators could display the system event log using the following command:

```
ipmitool -I lanplus -H 172.31.128.11 -U root
-P calvin sel list
```

By default, the IPMITool open interface works only with root privileges because the device `/dev/ipmi0` is set with root ownership. Administrators can make IPMITool accessible for ordinary users as follows:

- Create a user group using the command `groupadd groupname` (where *groupname* is the name of the new group).
- Change the group for `/dev/ipmi0` using the following command:

```
chgrp groupname /dev/ipmi0
```

3. Change the file permissions using the following command:

```
chmod 664 /dev/ipmi0
```

4. Check the group ID of the new group using the following command:

```
cat /etc/group | grep groupname
```

5. Replace the group ID of the ordinary user in /etc/passwd with this number.

Enabling and using Serial Over LAN

The SOL protocol can transmit serial data over a LAN using IPMI Over LAN packets. It works by establishing an IPMI Over LAN session between the BMC and the remote management applications; once the session is established, the remote console can request that SOL be activated. The BMC can then assemble the serial packets and send them to the remote console over the LAN. IPMI 1.5 required the separate SOL Proxy daemon to enable console redirection. However, in IPMI 2.0, the lanplus interface uses RMCP+ to connect to the BMC, the terminal is set to raw mode, and user input is sent to the serial console on the remote server. IPMItool itself can act as a serial console without requiring an additional proxy service. On exit, the SOL payload mode is deactivated, returning the terminal to its original settings.

Enabling SOL functionality

Administrators can enable SOL functionality in the BIOS utility by first rebooting the server and pressing F2 to launch the utility. In the Serial Communication options, they should then set the Serial Communication setting to On with Console Redirection via COM2, the External Serial Connector setting to Remote Access Device, the Failsafe Baud Rate setting to any suitable value (the BIOS attempts to determine this value automatically, and uses this baud rate only if that attempt fails), the Remote Terminal Type setting to VT100/VT220, and the Redirection After Boot setting to Enabled (see Figure 4).

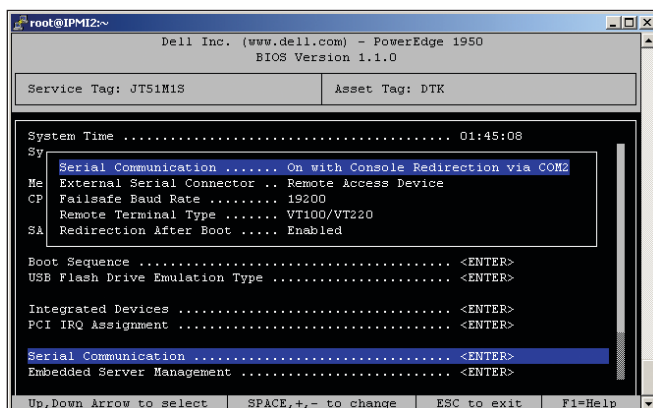


Figure 4. Serial Communication options in the Dell BIOS utility

Keyboard mapping for console redirection or session task	Escape sequence
Terminate connection	~+.
Suspend IPMItool	~+^+Z
Send break	~+B
Print escape sequence help	~+?
F1	Esc+1
F2	Esc+2
F3	Esc+3
F9	Esc+9
F10	Esc+0
F11	Esc+!
F12	Esc+@
Home	Esc+h
End	Esc+k
Insert	Esc++
Delete	Esc+-
Page Up	Esc+?
Page Down	Esc+/ Esc+>
Ctrl+M	Esc+Ctrl+M
Ctrl+H	Esc+Ctrl+H
Ctrl+I	Esc+Ctrl+I
Ctrl+J	Esc+Ctrl+J
Alt+x (where x is any letter)	Esc+X+x (where x is any letter, and X is the uppercase of that letter)
Ctrl+Alt+Del	Esc+R+Esc+r+Esc+R

Note: In these sequences, + denotes combinations of keys and is not a part of the sequence. For example, Ctrl+M denotes combining the Ctrl and M keys.

Figure 5. SOL session escape sequences

Using SOL

Administrators can activate a SOL session using the following command:

```
ipmitool -I lanplus -H ipaddress -U username  
-P password sol activate
```

This command should produce the following output:

```
[SOL Session operational. Use ~? for help]
```

This message indicates that the SOL session is active, and that there is currently no console redirection activity. Only one SOL session can be active at a time. Administrators can use various escape sequences within a SOL session, as shown in Figure 5.

The escape sequence ~+. terminates the session and resets the terminal settings. However, if SOL mode exits unintentionally and the BMC

must be reset, administrators can also terminate the current session from another console using the following command:

```
ipmitool -I lanplus -H ipaddress -U username -P password sol deactivate
```

Using IPMITool raw commands

The IPMI specification defines many more management capabilities than the IPMITool command-line options can provide. Administrators can take advantage of these capabilities by framing raw command requests and sending them to BMCs.

IPMITool raw commands have the following format:

```
ipmitool -I interface options raw netfn cmd data
```

In this format, *interface* can be `open`, `lan`, or `lanplus`. If `open` is used, *options* is not included; otherwise, *options* is `-H ipaddress -U username -P password`. The *netfn* element is the network function, which identifies the functional message class and clusters IPMI commands into different sets.⁴ The *cmd* element represents a unique one-byte command value within a given network function. Finally, the *data* element provides additional parameters for a request or response, if any.

Three examples can help demonstrate using raw commands: changing the power cycle interval, creating custom LCD messages, and generating platform event messages.

Changing the power cycle interval

The power cycle interval is the time a system is powered down during a power cycle operation initiated by either a chassis control command or a watchdog timer. IPMITool does not include an option for setting this interval, but administrators can do so using raw commands.

```
import os
load = open('/proc/loadavg').readline().
    split()[2]
loadAvg = [hex(ord(i)) for i in 'LoadAvg
'+str(load)]
cmd = 'ipmitool raw 0x6 0x58 193 0 0 %d
%s'%(len(loadAvg), ''.join(loadAvg))
os.system(cmd)
os.system('ipmitool raw 0x6 0x58 194 0')
```

Figure 6. Python script to display average server load on an LCD

Appendix G of the IPMI 2.0 specification provides the command to change the power cycle interval in the “Chassis Device Commands” section as “Set Power Cycle Interval,” denoting the network function as “Chassis” and the command value as 0Bh. Table 5-1 in the specification shows that “Chassis” resolves to 00h (or 0x00). Appendix G also refers to section 28.9, where Table 28-10 provides the format for the request data—a one-byte value representing the power cycle interval in seconds.

For example, administrators can set the power cycle interval to 15 seconds by constructing the following raw command:

```
ipmitool -v -I open raw 0x0 0x0B 0x0F
```

In this command, 0x0 is the network function, 0x0B is the command value, and 0x0F is the power cycle interval (the one-byte value representing 15 seconds). Because the command included the `-v` option for verbose output, the output would be as follows:

```
RAW REQ (channel=0x0 netfn=0x0 lun=0x0 cmd=0xb
data_len=1)
RAW REQUEST (1 bytes)
0f
RAW RSP (0 bytes)
```

As denoted in Table 28-10 of the IPMI 2.0 specification, the output also includes one byte of response data—in this case, 0, indicating a successful response. If the command had been framed incorrectly, then the response data might be 0xc7 and return the following output:

```
Unable to send RAW command (channel=0x0
netfn=0x0 lun=0x0 cmd=0xb rsp=0xc7): Request
data length invalid
```

To verify that the settings have been successfully applied, administrators could run the command `ipmitool chassis power cycle` and check that the server takes 15 seconds to power on again following the reboot.

Creating custom LCD messages

Server LCDs display details such as server type and platform events. Administrators can use them, for example, to help identify a particular server by using the `ipmitool -identify` command, which blinks the LCD. In addition, LCDs can display custom messages, which is typically done through Dell OpenManage™ Server Administrator commands. However, administrators can also use raw commands to display these messages.

The Python script shown in Figure 6 displays the server load average on the LCD. Administrators could run this script from the command line

⁴For more information, see section 5.1 of the IPMI 2.0 specification.

or as a cron job at five-minute intervals to update the LCD with the latest load values. For example, if the load average is 0.09, then the server would display “LoadAvg 0.09” on its LCD.

This script incorporates two raw commands based on the “Set System Info Parameters” row in the “BMC Device and Messaging Commands” section of Table G-1 of the IPMI 2.0 specification:

```
ipmitool raw 0x6 0x58 193 0 0 length
  ASCII_hex_values
ipmitool raw 0x6 0x58 194 0
```

The first command sets the LCD text, and the second enables the BMC to show this text instead of the default text. In these commands, 0x6 is the network function for “Set System Info Parameters” and 0x58 is the command value, 193 is the data header specifying the text assignment, 194 specifies which string to show on the LCD, the first 0 after 193 specifies which chunk of 16 bytes of text is being edited, the second 0 after 193 specifies the type of text encoding (in this case, ASCII), *length* specifies the number of characters shown on the LCD, and *ASCII_hex_values* is the hexadecimal values of the ASCII characters to be shown on the LCD (which is equal to the *length* value).

Generating platform event messages

Platform event message commands serve as requests for the BMC to process the event data that a command contains. This data is typically stored in the system event log. Depending on the implementation, the data may also go to the event message buffer and be processed by a Platform Event Filter.

Based on these events, administrators may want to write monitoring scripts to help them counter system failures and take appropriate action. To help test system monitoring scripts, they may also need to generate test event messages, which they can do using raw commands.

The format for generating a platform event message is as follows:

```
ipmitool -I lan -H ipaddress -U username raw
  netfn cmd evmrev sensortype sensorid
  eventdir/eventtype eventdata
```

In this format, *evmrev* is the one-byte event message revision, which for IPMI 2.0 would be 04h. The *sensortype* element is the one-byte event class or type of sensor that generates the event message, and the *sensorid* element is the one-byte unique number representing the sensor within the management controller. The *eventdir/eventtype* element combines one bit indicating the transition direction (0 for an assertion event, 1 for a de-assertion event) with seven bits indicating the type of threshold crossing or state transition that produced the event (encoded using the event/reading type code). Finally, the *eventdata* element consists of one to three bytes

“When combined with the guidelines in the IPMI 2.0 specification and the BMCs of Dell PowerEdge servers, IPMITool can become an indispensable tool in enterprise data centers.”

containing the sensor-specific offset from Table 42-3 of the IPMI 2.0 specification, which is based on the class of the event type for the sensor (threshold, discrete, or OEM).

For example, the command to generate a “processor disabled” event is as follows:


```
ipmitool -I lanplus -H ipaddress -U username raw
  0x04 0x02 0x04 0x07 0x61 0x6F 0x08 00 00
```

Appendix G of the IPMI 2.0 specification lists the command number assignments, which include “Platform Event (a.k.a. ‘Event Message’)” listed in the “Event Commands” section. This table also denotes the network function as “Sensor/Event.” The “processor disabled” raw command was generated as follows:

- 0x04 (*netfn*): Table 5-1 of the IPMI 2.0 specification lists 04h as the command/request code for “Sensor/Event” (with 05h being a response).
- 0x02 (*cmd*): Appendix G of the IPMI 2.0 specification lists the command value for “Platform Event (a.k.a. ‘Event Message’)” in the “Event Commands” section as 02h.
- 0x04 (*evmrev*): As noted, the event message revision in IPMI 2.0 is 04h.
- 0x07 (*sensortype*): Table 42-3 of the IPMI 2.0 specification lists the sensor-type command for “Processor” as 07h.
- 0x61 (*sensorid*): The sensor ID value for the processor can be obtained using the *sensor* command with the *-v* option and searching for “Processor/CPU.”
- 0x6F (*eventdir/eventtype*): Table 29-5 of the IPMI 2.0 specification lists the event direction for an assertion event as 0, and Table 42-1 lists the event type for a sensor-specific event (such as “processor disabled”) as 6Fh. Combining the event direction, 0h (0), with the event type, 6Fh (1101111), produces a final binary value of 01101111, which is equivalent to 0x6F.
- 0x08 00 00 (*eventdata*): Table 42-3 of the IPMI 2.0 specification lists the sensor-specific offset for “processor disabled” as 08h. The second two bytes are left blank.

Figure 7 lists other example raw commands for generating platform event messages. The sensor IDs are platform specific. Once the Platform Event Trap events are generated, administrators can verify them using the command `ipmitool -I open sel list`.

Expanding IPMITool remote management with raw commands


Raw commands enable administrators to expand the functionality of IPMITool and gain powerful remote management capabilities. When combined with the guidelines in the IPMI 2.0 specification and the BMCs of Dell PowerEdge servers, IPMITool can become an indispensable tool in enterprise data centers. 

Seshadri N. is a Product Group senior engineering analyst in the Dell Tools and Automation Framework Group. He has a B.E. in Computer Science from Visvesvaraya Technological University and an M.S. in Software Systems from the Birla Institute of Technology and Science, Pilani.

Raghavendra Babu is a Product Group test engineer senior analyst on the Test Engineering team at the Dell Bangalore Development Center. He has a B.E. in Computer Science and Engineering from Visvesvaraya Technological University.

Platform event message	Raw command
Over temperature	raw 0x04 0x02 0x04 0x01 0x01 0x01 0x09 00 00
Over voltage	raw 0x04 0x02 0x04 0x02 0x11 0x01 0x09 00 00
Chassis intrusion	raw 0x04 0x02 0x04 0x05 0x52 0x6f 0x00 00 00
Fan redundancy lost	raw 0x04 0x02 0x04 0x04 0x30 0x0b 0x01 00 00
Power redundancy lost	raw 0x04 0x02 0x04 0x08 0x44 0x0b 0x01 00 00
Power supply failure	raw 0x04 0x02 0x04 0x08 0x44 0x6f 0x01 00 00
Power supply lost	raw 0x04 0x02 0x04 0x08 0x44 0x6f 0x03 00 00
Memory redundancy lost	raw 0x04 0x02 0x04 0x0c 0x13 0x0b 0x01 00 00
System event log full	raw 0x04 0x02 0x04 0x10 0x51 0x6f 0x04 00 00

Figure 7. Example raw commands for generating platform event messages



DELL.COM/PowerSolutions

QUICK LINKS

IPMI specification:
developer.intel.com/design/servers/ipmi/spec.htm

IPMITool man page:
ipmitool.sourceforge.net/manpage.html

Khobragade, Kalyani. "Exploring the Remote Access Configuration Utility in Ninth-Generation Dell PowerEdge Servers." *Dell Power Solutions*, February 2007. DELL.COM/downloads/global/power/ps1q07-20060359-Khobragade.pdf