NAME

ipmitool − utility for controlling IPMI−enabled devices

SYNOPSIS

ipmitool [**−c**|**−h**|**−d** *N*|**−v**|**−V**] **−I** *open* <*command*>

ipmitool [**−c**|**−h**|**−v**|**−V**] **−I** *lan* **−H** <*hostname*>
  [**−p** <*port*>]
  [**−U** <*username*>]
  [**−A** <*authtype*>]
  [**−L** <*privlvl*>]
  [**−a**|**−E**|**−P**|**−f** <*password*>]
  [**−o** <*oemtype*>]
  [**−O** <*sel oem*>]
  [**−e** <*esc_char*>]
  [**−N** <*sec*>]
  [**−R** <*count*>]
  <*command*>

ipmitool [**−c**|**−h**|**−v**|**−V**] **−I** *lanplus* **−H** <*hostname*>
  [**−p** <*port*>]
  [**−U** <*username*>]
  [**−L** <*privlvl*>]
  [**−a**|**−E**|**−P**|**−f** <*password*>]
  [**−o** <*oemtype*>]
  [**−O** <*sel oem*>]
  [**−C** <*ciphersuite*>]
  [**−Y**|[**−K**|**−k** <*kg_key*>]
  [**−y** <*hex_kg_key*>]
  [**−e** <*esc_char*>]
  [**−N** <*sec*>]
  [**−R** <*count*>]
  <*command*>

DESCRIPTION

This program lets you manage Intelligent Platform Management Interface (IPMI) functions of either the local system, via a kernel device driver, or a remote system, using IPMI V1.5 and IPMI v2.0. These functions include printing FRU information, LAN configuration, sensor readings, and remote chassis power control.

IPMI management of a local system interface requires a compatible IPMI kernel driver to be installed and configured. On Linux this driver is called *OpenIPMI* and it is included in standard distributions. On Solaris this driver is called *BMC* and is inclued in Solaris 10. Management of a remote station requires the IPMI−over−LAN interface to be enabled and configured. Depending on the particular requirements of each system it may be possible to enable the LAN interface using ipmitool over the system interface.

OPTIONS

−a Prompt for the remote server password.

−A <*authtype*>
  Specify an authentication type to use during IPMIv1.5 *lan* session activation. Supported types are NONE, PASSWORD, MD2, MD5, or OEM.

−c Present output in CSV (comma separated variable) format. This is not available with all commands.

−e <*sol_escape_char*>
: Use supplied character for SOL session escape character. The default is to use ˜ but this can conflict with ssh sessions.

−K Read Kg key from IPMI_KGKEY environment variable.

−k <*key*>
: Use supplied Kg key for IPMIv2 authentication. The default is not to use any Kg key.

−y <*hex key*>
: Use supplied Kg key for IPMIv2 authentication. The key is expected in hexadecimal format and can be used to specify keys with non-printable characters. E.g. '-k PASSWORD' and '-y 50415353574F5244' are equivalent. The default is not to use any Kg key.

−Y Prompt for the Kg key for IPMIv2 authentication.

−C <*ciphersuite*>
: The remote server authentication, integrity, and encryption algorithms to use for IPMIv2 *lanplus* connections. See table 22−19 in the IPMIv2 specification. The default is 3 which specifies RAKP−HMAC−SHA1 authentication, HMAC−SHA1−96 integrity, and AES−CBC−128 encryption algorightms.

−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 password_file is empty, the password will default to NULL.

−h Get basic usage help from the command line.

−H <*address*>
: Remote server address, can be IP address or hostname. This option is required for *lan* and *lanplus* interfaces.

−I <*interface*>
: Selects IPMI interface to use. Supported interfaces that are compiled in are visible in the usage help output.

−L <*privlvl*>
: Force session privilege level. Can be CALLBACK, USER, OPERATOR, ADMINISTRATOR. Default is ADMINISTRATOR.

−m <*local_address*>
: Set the local IPMB address. The default is 0x20 and there should be no need to change it for normal operation.

−N <*sec*>
: Specify nr. of seconds between retransmissions of lan/lanplus messages. Default are 2 seconds for lan and 4 seconds for lanplus interfaces.

−o <*oemtype*>
: Select OEM type to support. This usually involves minor hacks in place in the code to work around quirks in various BMCs from various manufacturers. Use *−o list* to see a list of current supported OEM types.

−O <*sel oem*>
: Open selected file and read OEM SEL event descriptions to be used during SEL listings. See examples in contrib dir for file format.

−p <*port*>
: Remote server UDP port to connect to. Default is 623.

−P <*password*>
: Remote server password is specified on the command line. If supported it will be obscured in the process list. **Note!** Specifying the password as a command line option is not recommended.

−R <*count*>
Set the number of retries for lan/lanplus interface (default=4).

−S <*sdr_cache_file*>
Use local file for remote SDR cache. Using a local SDR cache can drastically increase performance for commands that require knowledge of the entire SDR to perform their function. Local SDR cache from a remote system can be created with the *sdr dump* command.

−t <*target_address*>
Bridge IPMI requests to the remote target address.

−U <*username*>
Remote server username, default is NULL user.

−d *N* Use device number N to specify the /dev/ipmiN (or /dev/ipmi/N or /dev/ipmidev/N) device to use for in-band BMC communication. Used to target a specific BMC on a multi-node, multi-BMC system through the ipmi device driver interface. Default is 0.

−v Increase verbose output level. This option may be specified multiple times to increase the level of debug output. If given three times you will get hexdumps of all incoming and outgoing packets.

−V Display version information.

If no password method is specified then ipmitool will prompt the user for a password. If no password is entered at the prompt, the remote server password will default to NULL.

SECURITY

There are several security issues be be considered before enabling the IPMI LAN interface. A remote station has the ability to control a system’s power state as well as being able to gather certain platform information. To reduce vulnerability it is strongly advised that the IPMI LAN interface only be enabled in ’trusted’ environments where system security is not an issue or where there is a dedicated secure ’management network’.

Further it is strongly advised that you should not enable IPMI for remote access without setting a password, and that that password should not be the same as any other password on that system.

When an IPMI password is changed on a remote machine with the IPMIv1.5 *lan* interface the new password is sent across the network as clear text. This could be observed and then used to attack the remote system. It is thus recommended that IPMI password management only be done over IPMIv2.0 *lanplus* interface or the system interface on the local station.

For IPMI v1.5, the maximum password length is 16 characters. Passwords longer than 16 characters will be truncated.

For IPMI v2.0, the maximum password length is 20 characters; longer passwords are truncated.

COMMANDS

help This can be used to get command−line help on ipmitool commands. It may also be placed at the end of commands to get option usage help.

```
ipmitool help
Commands:
       raw          Send a RAW IPMI request and print response
       i2c          Send an I2C Master Write-Read command and print response
       spd          Print SPD info from remote I2C device
       lan          Configure LAN Channels
       chassis      Get chassis status and set power state
       power        Shortcut to chassis power commands
       event        Send events to MC
```

mc	Management Controller status and global enables
sdr	Print Sensor Data Repository entries and readings
sensor	Print detailed sensor information
fru	Print built−in FRU and scan for FRU locators
sel	Print System Event Log (SEL)
pef	Configure Platform Event Filtering (PEF)
sol	Configure and connect IPMIv2.0 Serial−over−LAN
tsol	Configure and connect Tyan IPMIv1.5 Serial−over−LAN
isol	Configure and connect Intel IPMIv1.5 Serial−over−LAN
user	Configure Management Controller users
channel	Configure Management Controller channels
session	Print session information
sunoem	Manage Sun OEM Extensions
kontronoem	Manage Kontron OEM Extensions
picmg	Run a PICMG/ATA extended command
firewall	Configure Firmware Firewall
shell	Launch interactive IPMI shell
exec	Run list of commands from file
set	Set runtime variable for shell and exec
delloem	Manage Dell OEM Extensions
echo	Used to echo lines to stdout in scripts
ekanalyzer	run FRU-Ekeying analyzer using FRU files

ipmitool chassis help
Chassis Commands: status, power, identify, policy, restart_cause, poh, bootdev, bootparam, self-test

ipmitool chassis power help
chassis power Commands: status, on, off, cycle, reset, diag, soft

bmc|mc

reset <**warm**|**cold**>

Instructs the BMC to perform a warm or cold reset.

guid Display the Management Controller Globally Unique IDentifier.

info

Displays information about the BMC hardware, including device revision, firmware revision, IPMI version supported, manufacturer ID, and information on additional device support.

watchdog

These commands allow a user to view and change the current state of the watchdog timer.

get

Show current Watchdog Timer settings and countdown state.

reset

Reset the Watchdog Timer to its most recent state and restart the countdown timer.

off

Turn off a currently running Watchdog countdown timer.

selftest

Check on the basic health of the BMC by executing the Get Self Test results command and report the results.

getenables

Displays a list of the currently enabled options for the BMC.

setenables <**option**>=[*on*|*off*]

Enables or disables the given *option*. This command is only supported over the system interface according to the IPMI specification. Currently supported values for *option* include:

recv_msg_intr

Receive Message Queue Interrupt

event_msg_intr

Event Message Buffer Full Interrupt

event_msg

Event Message Buffer

system_event_log

System Event Logging

oem0

OEM−Defined option #0

oem1

OEM−Defined option #1

oem2

OEM−Defined option #2

channel

authcap <**channel number**> <**max priv**>

Displays information about the authentication capabilities of the selected channel at the specified privilege level.

Possible privilege levels are:

1	Callback level
2	User level
3	Operator level
4	Administrator level
5	OEM Proprietary level

info [**channel number**]

Displays information about the selected channel. If no channel is given it will display information about the currently used channel.

```
> ipmitool channel info
Channel 0xf info:
  Channel Medium Type   : System Interface
  Channel Protocol Type : KCS
  Session Support       : session−less
  Active Session Count  : 0
  Protocol Vendor ID    : 7154
```

getaccess <**channel number**> [<**userid**>]

Configure the given userid as the default on the given channel number. When the given channel is subsequently used, the user is identified implicitly by the given userid.

setaccess <**channel number**> <**userid**> [<*callin*=**on**|**off**>]
[<*ipmi*=**on**|**off**>] [<*link*=**on**|**off**>] [<*privilege*=**level**>]

Configure user access information on the given channel for the given userid.

getciphers <*ipmi*|*sol*> [<**channel**>]

Displays the list of cipher suites supported for the given application (ipmi or sol) on the given channel.

setkg <*hex*|*plain*> <**key**> [<**channel**>]

Sets K_g key to given value. Use *plain* to specify **key** as simple ASCII string. Use *hex* to specify **key** as sequence of hexadecimal codes of ASCII charactes. I.e. following two examples are equivalent:

```
ipmitool channel setkg plain PASSWORD

ipmitool channel setkg hex 50415353574F5244
```

chassis

status

Displays information regarding the high−level status of the system chassis and main power subsystem.

poh

This command will return the Power−On Hours counter.

identify <**interval**>

Control the front panel identify light. Default interval is 15 seconds. Use 0 to turn off. Use "force" to turn on indefinitely.

restart_cause

Query the chassis for the cause of the last system restart.

selftest

Check on the basic health of the BMC by executing the Get Self Test results command and report the results.

policy

Set the chassis power policy in the event power failure.

list

Return supported policies.

always−on

Turn on when power is restored.

previous

Returned to previous state when power is restored.

always−off

Stay off after power is restored.

power

Performs a chassis control command to view and change the power state.

status

Show current chassis power status.

on

Power up chassis.

off

Power down chassis into soft off (S4/S5 state). **WARNING**: This command does not initiate a clean shutdown of the operating system prior to powering down the system.

cycle

Provides a power off interval of at least 1 second. No action should occur if chassis power is in S4/S5 state, but it is recommended to check power state first and only issue a power cycle command if the system power is on or in lower sleep state than S4/S5.

reset

This command will perform a hard reset.

diag

Pulse a diagnostic interrupt (NMI) directly to the processor(s).

soft

Initiate a soft−shutdown of OS via ACPI. This can be done in a number of ways, commonly by simulating an overtemperture or by simulating a power button press. It is necessary for there to be Operating System support for ACPI and some sort of daemon watching for events for this soft power to work.

bootdev <**device**> [<*clear−cmos*=**yes**|**no**>] [<*options*=**help,...**>]

Request the system to boot from an alternate boot device on next reboot. The *clear−cmos* option, if supplied, will instruct the BIOS to clear its CMOS on the next reboot. Various options may be used to modify the boot device settings. Run *"bootdev none options=help"* for a list of available boot device modifiers/options.

Currently supported values for <device> are:

none

Do not change boot device

pxe

Force PXE boot

disk

Force boot from BIOS default boot device

safe

Force boot from BIOS default boot device, request Safe Mode

diag

Force boot from diagnostic partition

cdrom

Force boot from CD/DVD

bios

Force boot into BIOS setup

floppy

Force boot from Floppy/primary removable media

bootparam

Get or set various system boot option parameters.

get <**param #**>

Get boot parameter. Currently supported values for <**param #**> are:

0 - Set In Progress

1 - Service Partition Selector

2 - Service Partition Scan

3 - BMC Boot Flag Valid Bit Clearing

4 - Boot Info Acknowledge

5 - Boot Flags

6 - Boot Initiator Info

7 - Boot Initiator Mailbox

set <**option**> [**value ...**]

Set boot parameter.

Currently supported values for **<option>** are:

force_pxe

Force PXE boot

force_disk

Force boot from default hard-drive

force_safe

Force boot from default hard-drive, request Safe Mode

force_diag

Force boot from diagnostic partition

force_cdrom

Force boot from CD/DVD

force_bios

Force boot into BIOS setup

ekanalyzer <**command**> <**xx=filename1**> <**xx=filename2**> [<**rc=filename3**>] **...**

NOTE : This command can support a maximum of 8 files per command line

filename1 : binary file that stores FRU data of a Carrier or an AMC module

filename2 : binary file that stores FRU data of an AMC module.
These binary files can be generated from command:
ipmitool fru read <id> <filename>

filename3 : configuration file used for configuring On-Carrier Device ID
or OEM GUID. This file is optional.

xx : indicates the type of the file. It can take the following value:

oc : On-Carrier device

a1 : AMC slot A1

a2 : AMC slot A2

a3 : AMC slot A3

a4 : AMC slot A4

b1 : AMC slot B1

b2 : AMC slot B2

b3 : AMC slot B3

b4 : AMC slot B4

sm : Shelf Manager

The available commands for ekanalyzer are:

print [<**carrier** | **power** | **all**>]

carrier (default) <**oc=filename1**> <**oc=filename2**> **...**

Display point to point physical connectivity between carriers and AMC modules.

```
 Example:
   > ipmitool ekanalyzer print carrier oc=fru oc=carrierfru
   From Carrier file: fru
      Number of AMC bays supported by Carrier: 2
      AMC slot B1 topology:
         Port 0 =====> On Carrier Device ID 0, Port 16
         Port 1 =====> On Carrier Device ID 0, Port 12
         Port 2 =====> AMC slot B2, Port 2
      AMC slot B2 topology:
         Port 0 =====> On Carrier Device ID 0, Port 3
         Port 2 =====> AMC slot B1, Port 2
   *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
   From Carrier file: carrierfru
      On Carrier Device ID 0 topology:
         Port 0 =====> AMC slot B1, Port 4
         Port 1 =====> AMC slot B1, Port 5
         Port 2 =====> AMC slot B2, Port 6
         Port 3 =====> AMC slot B2, Port 7
      AMC slot B1 topology:
         Port 0 =====> AMC slot B2, Port 0
      AMC slot B1 topology:
         Port 1 =====> AMC slot B2, Port 1
      Number of AMC bays supported by Carrier: 2
```

power <**xx=filename1**> <**xx=filename2**> **...**

Display power supply informations between carrier and AMC modules.

all <**xx=filename**> <**xx=filename**> **...**

Display both physical connectivity and power supply of each carrier and AMC modules.

frushow <**xx=filename**>

Convert a binary FRU file into human readable text format. Use -v option to get more display information.

summary [<**match** | **unmatch** | **all**>]

match (default) <**xx=filename**> <**xx=filename**> **...**

Display only matched results of Ekeying match between an On-Carrier device and an AMC module or between 2 AMC modules. Example:

```
 > ipmitool ekanalyzer summary match oc=fru b1=amcB1 a2=amcA2
 On-Carrier Device vs AMC slot B1
  AMC slot B1 port 0 ==> On-Carrier Device 0 port 16
   Matching Result
   - From On-Carrier Device ID 0
   -Channel ID 11 || Lane 0: enable
   -Link Type: AMC.2 Ethernet
   -Link Type extension: 1000BASE-BX (SerDES Gigabit) Ethernet link
   -Link Group ID: 0 || Link Asym. Match: exact match
  - To AMC slot B1
   -Channel ID 0 || Lane 0: enable
   -Link Type: AMC.2 Ethernet
   -Link Type extension: 1000BASE-BX (SerDES Gigabit) Ethernet link
   -Link Group ID: 0 || Link Asym. Match: exact match
  *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
 AMC slot B1 port 1 ==> On-Carrier Device 0 port 12
  Matching Result
  - From On-Carrier Device ID 0
   -Channel ID 6 || Lane 0: enable
   -Link Type: AMC.2 Ethernet
   -Link Type extension: 1000BASE-BX (SerDES Gigabit) Ethernet link
   -Link Group ID: 0 || Link Asym. Match: exact match
  - To AMC slot B1
   -Channel ID 1 || Lane 0: enable
   -Link Type: AMC.2 Ethernet
   -Link Type extension: 1000BASE-BX (SerDES Gigabit) Ethernet link
   -Link Group ID: 0 || Link Asym. Match: exact match
  *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
On-Carrier Device vs AMC slot A2
 AMC slot A2 port 0 ==> On-Carrier Device 0 port 3
  Matching Result
  - From On-Carrier Device ID 0
   -Channel ID 9 || Lane 0: enable
   -Link Type: AMC.2 Ethernet
   -Link Type extension: 1000BASE-BX (SerDES Gigabit) Ethernet link
   -Link Group ID: 0 || Link Asym. Match: exact match
  - To AMC slot A2
   -Channel ID 0 || Lane 0: enable
   -Link Type: AMC.2 Ethernet
   -Link Type extension: 1000BASE-BX (SerDES Gigabit) Ethernet link
   -Link Group ID: 0 || Link Asym. Match: exact match
  *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
AMC slot B1 vs AMC slot A2
 AMC slot A2 port 2 ==> AMC slot B1 port 2
  Matching Result
  - From AMC slot B1
   -Channel ID 2 || Lane 0: enable
   -Link Type: AMC.3 Storage
   -Link Type extension: Serial Attached SCSI (SAS/SATA)
   -Link Group ID: 0 || Link Asym. Match: FC or SAS interface {exact match}
  - To AMC slot A2
   -Channel ID 2 || Lane 0: enable
   -Link Type: AMC.3 Storage
   -Link Type extension: Serial Attached SCSI (SAS/SATA)
   -Link Group ID: 0 || Link Asym. Match: FC or SAS interface {exact match}
*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
```

unmatch <**xx=filename**> <**xx=filename**> **...**

Display the unmatched results of Ekeying match between an On-Carrier device and an AMC module or between 2 AMC modules

all <**xx=filename**> <**xx=filename**> **...**

Display both matched result and unmatched results of Ekeying match between two cards or two modules.

delloem

The delloem commands provide information on Dell-specific features.

setled {b:d.f} {state..}

Sets the drive backplane LEDs for a device.
{b:d.f} = PCI Address of device (eg. 06:00.0)
{state} = one or more of the following:
online | *present* | *hotspare* | *identify* | *rebuilding* | *fault* | *predict* | *critical* | *failed*

lcd

set {mode}|{lcdqualifier}|{errordisplay}

Allows you to set the LCD mode and user-defined string.

lcd set mode
{none}|{modelname}|{ipv4address}|{macaddress}|
{systemname}|{servicetag}|{ipv6address}|
{ambienttemp}|{systemwatt}|{assettag}|
{userdefined}<text>

Allows you to set the LCD display mode to any of the preceding parameters.

lcd set lcdqualifier
{watt}|{btuphr}|
{celsius}|{fahrenheit}

Allows you to set the unit for the system ambient temperature mode.

lcd set errordisplay
{sel}|{simple}

Allows you to set the error display.

lcd info

Displays the LCD screen information.

lcd set vkvm
{active}|{inactive}

Allows you to set the vKVM status to active or inactive. When it is active and session is in progress, a message appears on LCD.

lcd status

Displays the LCD status for vKVM display active or inactive and Front Panel access mode (viewandmodify, view-only or disabled).

setled **<b:d.f> <state> [state...]**

Allows to set backplane LED state.

<b:d.f>

PCI Bus:Device.Function of drive (lspci format).

<state>

Sets the LED state (present, online, hotspare, identify, rebuilding, fault, predict, critical or failed). More than one state can be specified, the final state is composed as logical OR of all specified states.

mac

Displays the information about the system NICs.

mac list

Displays the NIC MAC address and status of all NICs. It also displays the DRAC/iDRAC MAC address.

mac get

<NIC number>

Displays the selected NICs MAC address and status.

lan

Displays the information of Lan.

lan set

<Mode>

Sets the NIC selection mode (dedicated, shared with lom1, shared with lom2,shared with lom3,shared with lom4,shared with failover lom1,shared with failover lom2,shared with failover lom3,shared with failover lom4,shared with Failover all loms, shared with Failover None).

lan get

Returns the current NIC selection mode (dedicated, shared with lom1, shared with lom2, shared with lom3, shared with lom4,shared with failover lom1, shared with failover lom2,shared with failover lom3,shared with failover lom4,shared with Failover all loms,shared with Failover None).

lan get active

Returns the current active NIC (dedicated, LOM1, LOM2, LOM3 or LOM4).

powermonitor

Displays power tracking statistics.

powermonitor clear cumulativepower

Reset cumulative power reading.

powermonitor clear peakpower

Reset peak power reading.

powermonitor powerconsumption
<watt>|<btuphr>
Displays the power consumption in watt or btuphr.

powermonitor powerconsumptionhistory
<watt>|<btuphr>
Displays the power consumption history in watt or btuphr.

powermonitor getpowerbudget
<watt>|<btuphr>
Displays the power cap in watt or btuphr.

powermonitor setpowerbudget
<val><watt|btuphr|percent>
Allows you to set the power cap in watt, BTU/hr or percentage.

powermonitor enablepowercap
Enables set power cap.

powermonitor disablepowercap

Disables set power cap.

windbg
windbg start
Starts the windbg session (Cold Reset & SOL Activation)
windbg end
Ends the windbg session (SOL Deactivation)

vFlash info Card

Shows Extended SD Card information

event

<**predefined event number** *N*>

Send a pre−defined test event to the System Event Log. The following events are included as a means to test the functionality of the System Event Log component of the BMC (an entry will be added each time the event *N* command is executed).

Currently supported values for *N* are:

1	Temperature: Upper Critical: Going High
2	Voltage Threshold: Lower Critical: Going Low
3	Memory: Correctable ECC

NOTE: These pre−defined events will likely not produce "accurate" SEL records for a particular system because they will not be correctly tied to a valid sensor number, but they are sufficient to verify correct operation of the SEL.

file <**filename**>

Event log records specified in <**filename**> will be added to the System Event Log.

The format of each line in the file is as follows:

<{*EvM Revision*} {*Sensor Type*} {*Sensor Num*} {*Event Dir/Type*} {*Event Data 0*} {*Event Data 1*} {*Event Data 2*}>[*# COMMENT*]

e.g.: 0x4 0x2 0x60 0x1 0x52 0x0 0x0 # Voltage threshold: Lower Critical: Going Low

EvM Revision - The "Event Message Revision" is 0x04 for messages that comply with the IPMI 2.0 Specification and 0x03 for messages that comply with the IPMI 1.0 Specification.

Sensor Type - Indicates the Event Type or Class.

Sensor Num - Represents the ’sensor’ within the management controller that generated the Event Message.

Event Dir/Type - This field is encoded with the event direction as the high bit (bit 7) and the event type as the low 7 bits. Event direction is 0 for an assertion event and 1 for a deassertion event.

See the IPMI 2.0 specification for further details on the definitions for each field.

<**sensorid**> <**list**>

Get a list of all the possible Sensor States and pre-defined Sensor State Shortcuts available for a particular sensor. **sensorid** is the character string representation of the sensor and must be enclosed in double quotes if it includes white space. Several different commands including *ipmitool sensor list* may be used to obtain a list that includes the **sensorid** strings representing the sensors on a given system.

```
> ipmitool −I open event "PS 2T Fan Fault" list
Finding sensor PS 2T Fan Fault... ok
Sensor States:
  State Deasserted
  State Asserted
Sensor State Shortcuts:
  present    absent
  assert     deassert
  limit      nolimit
  fail       nofail
  yes        no
  on         off
  up         down
```

<**sensorid**> <**sensor state**> [<**direction**>]

Generate a custom event based on existing sensor information. The optional event **direction can be either** *assert* **(the default) or** *deassert***.**

```
> ipmitool event "PS 2T Fan Fault" "State Asserted"
Finding sensor PS 2T Fan Fault... ok
   0 | Pre-Init Time-stamp   | Fan PS 2T Fan Fault | State Asserted
> ipmitool event "PS 2T Fan Fault" "State Deasserted"
Finding sensor PS 2T Fan Fault... ok
   0 | Pre-Init Time-stamp   | Fan PS 2T Fan Fault | State Desserted
```

exec <**filename**>

Execute ipmitool commands from *filename*. Each line is a complete command. The syntax of the commands are defined by the COMMANDS section in this manpage. Each line may have an optional comment at the end of the line, delimited with a '#' symbol.

e.g., a command file with two lines:

```
sdr list # get a list of sdr records
sel list # get a list of sel records
```

fru

print

Read all Field Replaceable Unit (FRU) inventory data and extract such information as serial number, part number, asset tags, and short strings describing the chassis, board, or product.

read <**fru id**> <**fru file**>

fru id is the digit ID of the FRU (see output of 'fru print'). **fru file** is the absolute pathname of a file in which to dump the binary FRU data pertaining to the specified FRU entity.

write <**fru id**> <**fru file**>

fru id is the digit ID of the FRU (see output of 'fru print'). **fru file** is the absolute pathname of a file from which to pull the binary FRU data before uploading it to the specified FRU.

upgEkey <**fru id**> <**fru file**>

Update a multirecord FRU location. **fru id** is the digit ID of the FRU (see output of 'fru print'). **fru file** is the absolute pathname of a file from which to pull the binary FRU data to upload into the specified multirecord FRU entity.

edit <**fru id**>

This command provides interactive editing of some supported records, namely PICMG Carrier Activation Record. **fru id** is the digit ID of the FRU (see output of 'fru print'); default is 0.

edit <**fru id**> **field** <**section**> <**index**> <**string**>

This command may be used to set a field string to a new value. It replaces the FRU data found at **index** in the specified **section** with the supplied **string**.

fru id is the digit ID of the FRU (see output of 'fru print').

<**section**> is a string which refers to FRU Inventory Information Storage Areas and may be refer to:

c FRU Inventory Chassis Info Area

b FRU Inventory Board Info Area

p FRU Inventory Product Info Area

<**index**> specifies the field number. Field numbering starts on the first 'english text' field type. For instance in the <**board**> info area field '0' is <**Board Manufacturer**> and field '2' is <**Board Serial Number**>; see IPMI Platform Management FRU Information Storage Definition v1.0 R1.1 for field locations.

<**string**> must be the same length as the string being replaced and must be 8-bit ASCII (0xCx).

edit <**fru id**> **oem iana** <**record**> <**format**> [<**args**>]

This command edits the data found in the multirecord area. Support for OEM specific records is limited.

firewall

This command supports the Firmware Firewall capability. It may be used to add or remove security-based restrictions on certain commands/command sub-functions or to list the current firmware firewall restrictions set on any commands. For each firmware firewall command listed below, parameters may be included to cause the command to be executed with increasing granularity on a specific LUN, for a specific NetFn, for a specific IPMI Command, and finally for a specific command's sub-function (see Appendix H in the IPMI 2.0 Specification for a listing of any sub-function numbers that may be associated with a particular command).

Parameter syntax and dependencies are as follows:

[<*channel* **H**>] [<*lun* **L**> [<*netfn* **N**> [<*command* **C** [<*subfn* **S**>]]]]

Note that if "netfn <**N**>" is specified, then "lun <**L**>" must also be specified; if "command <**C**>" is specified, then "netfn <**N**>" (and therefore "lun <**L**>") must also be specified, and so forth.

"channel <**H**>" is an optional and standalone parameter. If not specified, the requested operation will be performed on the current channel. Note that command support may vary from channel to channel.

Firmware firewall commands:

info [<**Parms as described above**>]

List firmware firewall information for the specified LUN, NetFn, and Command (if supplied) on the current or specified channel. Listed information includes the support, configurable, and enabled bits for the specified command or commands.

Some usage examples:

info [<**channel H**>] [<**lun L**>]

This command will list firmware firewall information for all NetFns for the specified LUN on either the current or the specified channel.

info [<**channel H**>] [<**lun L**> [<**netfn N**>]

This command will print out all command information for a single LUN/NetFn pair.

info [<**channel H**>] [<**lun L**> [<**netfn N**> [<**command C**]]]

This prints out detailed, human-readable information showing the support, configurable, and enabled bits for the specified command on the specified LUN/NetFn pair. Information will be printed about each of the command subfunctions.

info [<**channel H**>] [<**lun L**> [<**netfn N**> [<**command C** [<**subfn S**>]]]]

Print out information for a specific sub-function.

enable [<**Parms as described above**>]

This command is used to enable commands for a given NetFn/LUN combination on the specified channel.

disable [<**Parms as described above**>] [**force**]

This command is used to disable commands for a given NetFn/LUN combination on the specified channel. Great care should be taken if using the "force" option so as not to disable the "Set Command Enables" command.

reset [<**Parms as described above**>]

This command may be used to reset the firmware firewall back to a state where all commands and command sub-functions are enabled.

i2c <**i2caddr**> <**read bytes**> [<**write data**>]

This command may be used to execute raw I2C commands with the Master Write−Read IPMI command.

isol

info

Retrieve information about the Intel IPMI v1.5 Serial−Over−LAN configuration.

set <**parameter**> <**value**>

Configure parameters for Intel IPMI v1.5 Serial−over−LAN.

Valid parameters and values are:

enabled
true, false.

privilege−level
user, operator, admin, oem.

bit−rate
9.6, 19.2, 38.4, 57.6, 115.2.

activate

Causes ipmitool to enter Intel IPMI v1.5 Serial Over LAN mode. An RMCP+ connection is made to the BMC, the terminal is set to raw mode, and user input is sent to the serial console on the remote server. On exit, the the SOL payload mode is deactivated and the terminal is reset to its original settings.

Special escape sequences are provided to control the SOL session:

~.	Terminate connection
~^Z	Suspend ipmitool
~^X	Suspend ipmitool, but don’t restore tty on restart
~B	Send break
~~	Send the escape character by typing it twice
~?	Print the supported escape sequences

Note that escapes are only recognized immediately after newline.

kontronoem

OEM commands specific to Kontron devices.

setsn

Set FRU serial number.

setmfgdate

Set FRU manufacturing date.

nextboot <**boot device**>

Select the next boot order on the Kontron CP6012.

lan

These commands will allow you to configure IPMI LAN channels with network information so they can be used with the ipmitool *lan* and *lanplus* interfaces. *NOTE*: To determine on which channel the LAN interface is located, issue the ‘channel info *number*’ command until you come across a valid 802.3 LAN channel. For example:

```
> ipmitool −I open channel info 1
Channel 0x1 info:
  Channel Medium Type   : 802.3 LAN
  Channel Protocol Type : IPMB−1.0
  Session Support       : session−based
  Active Session Count  : 8
  Protocol Vendor ID    : 7154
```

print [<**channel**>]

Print the current configuration for the given channel. The default will print information on the first found LAN channel.

set <**channel number**> <**command**> <**parameter**>

Set the given command and parameter on the specified channel. Valid command/parameter options are:

ipaddr <**x.x.x.x**>

Set the IP address for this channel.

netmask <**x.x.x.x**>

Set the netmask for this channel.

macaddr <**xx:xx:xx:xx:xx:xx**>

Set the MAC address for this channel.

defgw ipaddr <**x.x.x.x**>

Set the default gateway IP address.

defgw macaddr <**xx:xx:xx:xx:xx:xx**>

Set the default gateway MAC address.

bakgw ipaddr <**x.x.x.x**>

Set the backup gateway IP address.

bakgw macaddr <**xx:xx:xx:xx:xx:xx**>

Set the backup gateway MAC address.

password <**pass**>

Set the null user password.

snmp <**community string**>

Set the SNMP community string.

user

Enable user access mode for userid 1 (issue the ‘user’ command to display information about userids for a given channel).

access <**on|off**>

Set LAN channel access mode.

alert <**on|off**>

Enable or disable PEF alerting for this channel.

ipsrc <**source**>

Set the IP address source:

none	unspecified

static	manually configured static IP address
dhcp	address obtained by BMC running DHCP
bios	address loaded by BIOS or system software

arp respond <**on**|**off**>

Set BMC generated ARP responses.

arp generate <**on**|**off**>

Set BMC generated gratuitous ARPs.

arp interval <**seconds**>

Set BMC generated gratuitous ARP interval.

vlan id <**off**|**id**>

Disable VLAN operation or enable VLAN and set the ID.
ID: value of the virtual lan identifier between 1 and 4094 inclusive.

vlan priority <**priority**>

Set the priority associated with VLAN frames.
ID: priority of the virtual lan frames between 0 and 7 inclusive.

auth <**level**,**...**> <**type**,**...**>

Set the valid authtypes for a given auth level.
Levels: callback, user, operator, admin
Types: none, md2, md5, password, oem

cipher_privs <**privlist**>

Correlates cipher suite numbers with the maximum privilege level that is allowed to use it. In this way, cipher suites can restricted to users with a given privilege level, so that, for example, administrators are required to use a stronger cipher suite than normal users.

The format of *privlist* is as follows. Each character represents a privilege level and the character position identifies the cipher suite number. For example, the first character represents cipher suite 1 (cipher suite 0 is reserved), the second represents cipher suite 2, and so on. *privlist* must be 15 characters in length.

Characters used in *privlist* and their associated privilege levels are:

X	Cipher Suite Unused
c	CALLBACK
u	USER
o	OPERATOR
a	ADMIN
O	OEM

So, to set the maximum privilege for cipher suite 1 to USER and suite 2 to ADMIN, issue the following command:

> ipmitool −I *interface* lan set *channel* cipher_privs uaXXXXXXXXXXXXX

alert print [<**channel**>] [<**alert destination**>]

Print alert information for the specified channel and destination. The default will print all alerts for all alert destinations on the first found LAN channel.

alert set <**channel number**> <**alert destination**> <**command**> <**parameter**>

Set an alert on the given LAN channel and destination. Alert Destinations are listed via the '*lan alert print*' command. Valid command/parameter options are:

ipaddr <**x.x.x.x**>

Set alert IP address.

macaddr <**xx:xx:xx:xx:xx:xx**>

Set alert MAC address.

gateway <**default | backup**>

Set the channel gateway to use for alerts.

ack <**on | off**>

Set Alert Acknowledge on or off.

type <**pet | oem1 | oem2**>

Set the destination type as PET or OEM.

time <**seconds**>

Set ack timeout or unack retry interval.

retry <**number**>

Set the number of alert retries.

stats get [<**channel number**>]

Retrieve information about the IP connections on the specified channel. The default will retrieve statistics on the first found LAN channel.

stats clear [<**channel number**>]

Clear all IP/UDP/RMCP Statistics to 0 on the specified channel. The default will clear statistics on the first found LAN channel.

pef

info

This command will query the BMC and print information about the PEF supported features.

status

This command prints the current PEF status (the last SEL entry processed by the BMC, etc).

policy

This command lists the PEF policy table entries. Each policy entry describes an alert destination. A policy set is a collection of table entries. PEF alert actions reference policy sets.

list

This command lists the PEF table entries. Each PEF entry relates a sensor event to an action. When PEF is active, each platform event causes the BMC to scan this table for entries matching the event, and possible actions to be taken. Actions are performed in priority order (higher criticality first).

picmg <**properties**>

Run a PICMG/ATA extended command. Get PICMG properties may be used to obtain and print Extension major version information, PICMG identifier, FRU Device ID and Max FRU Device ID.

addrinfo

Get address information. This command may return information on the Hardware address, IPMB-0 Address, FRU ID, Site/Entity ID, and Site/Entity Type.

frucontrol <**fru id**> <**options**>

Set various control options:

0x00 - Cold Reset

0x01 - Warm Reset

0x02 - Graceful Reboot

0x03 - Issue Diagnostic Interrupt

0x04 - Quiesce [AMC only]

0x05-0xFF - Cold Reset

activate <**fru id**>

Activate the specified FRU.

deactivate <**fru id**>

Deactivate the specified FRU.

policy get <**fru id**>

Get FRU activation policy.

policy set <**fru id**> <**lockmask**> <**lock**>

Set FRU activation policy. **lockmask** is 1 or 0 to indicate action on the deactivation or activation locked bit respectively. **lock** is 1 or 0 to set/clear locked bit.

portstate **set|getall|getgranted|getdenied** <**parameters**>

Get or set various port states. See usage for parameter details.

power <**chassis power command**>

Shortcut to the *chassis power* commands. See the *chassis power* commands for usage information.

raw <**netfn**> <**cmd**> [<**data**>]

This will allow you to execute raw IPMI commands. For example to query the POH counter with a raw command:

```
> ipmitool −v raw 0x0 0xf
RAW REQ (netfn=0x0 cmd=0xf data_len=0)
RAW RSP (5 bytes)
3c 72 0c 00 00
```

sdr

get <**id**> ... [<**id**>]

Prints information for sensor data records specified by sensor id.

info

This command will query the BMC for Sensor Data Record (SDR) Repository information.

type [<**sensor type**>]

This command will display all records from the SDR Repository of a specific type. Run with type *list* (or simply with no type) to see the list of available types. For example to query for all Temperature sensors:

```
> ipmitool sdr type Temperature
Baseboard Temp   | 30h | ok  |  7.1 | 28 degrees C
FntPnl Amb Temp  | 32h | ok  | 12.1 | 24 degrees C
Processor1 Temp  | 98h | ok  |  3.1 | 57 degrees C
Processor2 Temp  | 99h | ok  |  3.2 | 53 degrees C
```

list | *elist* [<**all**|**full**|**compact**|**event**|**mcloc**|**fru**|**generic**>]

This command will read the Sensor Data Records (SDR) and extract sensor information of a given type, then query each sensor and print its name, reading, and status. If invoked as *elist* then it will also print sensor number, entity id and instance, and asserted discrete states.

The default output will only display *full* and *compact* sensor types, to see all sensors use the *all* type with this command.

Valid types are:

all

All SDR records (Sensor and Locator)

full

Full Sensor Record

compact

Compact Sensor Record

event

Event−Only Sensor Record

mcloc

Management Controller Locator Record

fru

FRU Locator Record

generic

Generic SDR records

entity <**id**>[.<**instance**>]

Displays all sensors associated with an entity. Get a list of valid entity ids on the target system by issuing the *sdr elist* command. A list of all entity ids can be found in the IPMI specifications.

dump <**file**>

Dumps raw SDR data to a file. This data file can then be used as a local SDR cache of the remote managed system with the *−S <file>* option on the ipmitool command line. This can greatly improve performance over system interface or remote LAN.

fill sensors

Create the SDR Repository for the current configuration. Will perform a ’Clear SDR Repository’ command so be careful.

fill file <**filename**>

Fill the SDR Repository using records stored in a binary data file. Will perform a ’Clear SDR Repository’ command so be careful.

sel

NOTE: System Event Log (SEL) entry−times are displayed as ‘Pre−Init Time−stamp’ if the SEL clock needs to be set. Ensure that the SEL clock is accurate by invoking the *sel time get* and *sel time set <time string>* commands.

info

This command will query the BMC for information about the System Event Log (SEL) and its contents.

clear

This command will clear the contents of the SEL. It cannot be undone so be careful.

list | elist

When this command is invoked without arguments, the entire contents of the System Event Log are displayed. If invoked as *elist* (extended list) it will also use the Sensor Data Record entries to display the sensor ID for the sensor that caused each event. **Note** this can take a long time over the system interface.

<**count**> | *first* <**count**>

Displays the first *count* (least−recent) entries in the SEL. If *count* is zero, all entries are displayed.

last <**count**>

Displays the last *count* (most−recent) entries in the SEL. If *count* is zero, all entries are displayed.

delete <**SEL Record ID**> ... <**SEL Record ID**>

Delete one or more SEL event records.

add <**filename ID**>

Read event entries from a file and add them to the SEL. New SEL entries area added onto the SEL after the last record in the SEL. Record added is of type 2 and is automatically timestamped.

get <**SEL Record ID**>

Print information on the specified SEL Record entry.

save <**file**>

Save SEL records to a text file that can be fed back into the *event file* ipmitool command. This can be useful for testing Event generation by building an appropriate Platform Event Message file based on existing events. Please see the available help for the ’event file ...’ command for a description of the format of this file.

writeraw <**file**>

Save SEL records to a file in raw, binary format. This file can be fed back to the *sel readraw* ipmitool command for viewing.

readraw <**file**>

Read and display SEL records from a binary file. Such a file can be created using the *sel writeraw* ipmitool command.

time

get

Displays the SEL clock’s current time.

set <**time string**>

Sets the SEL clock. Future SEL entries will use the time set by this command. <**time string**> is of the form "MM/DD/YYYY HH:MM:SS". Note that hours are in 24−hour form. It is recommended that the SEL be cleared before setting the time.

sensor

list

Lists sensors and thresholds in a wide table format.

get <**id**> ... [<**id**>]

Prints information for sensors specified by name.

thresh <**id**> <**threshold**> <**setting**>

This allows you to set a particular sensor threshold value. The sensor is specified by name.

Valid *thresholds* are:

unr	Upper Non−Recoverable
ucr	Upper Critical
unc	Upper Non−Critical
lnc	Lower Non−Critical
lcr	Lower Critical
lnr	Lower Non−Recoverable

thresh <**id**> *lower* <**lnr**> <**lcr**> <**lnc**>

This allows you to set all lower thresholds for a sensor at the same time. The sensor is specified by name and the thresholds are listed in order of Lower Non−Recoverable, Lower Critical, and Lower Non−Critical.

thresh <**id**> *upper* <**unc**> <**ucr**> <**unr**>

This allows you to set all upper thresholds for a sensor at the same time. The sensor is specified by name and the thresholds are listed in order of Upper Non−Critical, Upper Critical, and Upper Non−Recoverable.

session

info <**active|all|id 0xnnnnnnnn|handle 0xnn**>

Get information about the specified session(s). You may identify sessions by their id, by their handle number, by their active status, or by using the keyword ‘all’ to specify all sessions.

shell

This command will launch an interactive shell which you can use to send multiple ipmitool commands to a BMC and see the responses. This can be useful instead of running the full ipmitool command each time. Some commands will make use of a Sensor Data Record cache and you will see marked improvement in speed if these commands are able to reuse the same cache in a shell session. LAN sessions will send a periodic keepalive command to keep the IPMI session from timing out.

sol

info [<**channel number**>]

Retrieve information about the Serial−Over−LAN configuration on the specified channel. If no channel is given, it will display SOL configuration data for the currently used channel.

payload <*enable* | *disable* | *status*> <**channel number**> <**userid**>

Enable, disable or show status of SOL payload for the user on the specified channel.

set <**parameter**> <**value**> [<**channel**>]

Configure parameters for Serial Over Lan. If no channel is given, it will display

SOL configuration data for the currently used channel. Configuration parameter updates are automatically guarded with the updates to the set−in−progress parameter.

Valid parameters and values are:

set−in−progress
set−complete set−in−progress commit−write

enabled
true false

force−encryption
true false

force−authentication
true false

privilege−level
user operator admin oem

character−accumulate−level
Decimal number given in 5 milliseconds increments

character−send−threshold
Decimal number

retry−count
Decimal number. 0 indicates no retries after packet is transmitted.

retry−interval
Decimal number in 10 millisend increments. 0 indicates that retries should be sent back to back.

non−volatile−bit−rate
serial, 19.2, 38.4, 57.6, 115.2. Setting this value to serial indicates that the BMC should use the setting used by the IPMI over serial channel.

volatile−bit−rate
serial, 19.2, 38.4, 57.6, 115.2. Setting this value to serial indiates that the BMC should use the setting used by the IPMI over serial channel.

activate [*usesolkeepalive* | *nokeepalive*]

Causes ipmitool to enter Serial Over LAN mode, and is only available when using the lanplus interface. An RMCP+ connection is made to the BMC, the terminal is set to raw mode, and user input is sent to the serial console on the remote server. On exit,the the SOL payload mode is deactivated and the terminal is reset to its original settings.

Special escape sequences are provided to control the SOL session:

~.	Terminate connection
~^Z	Suspend ipmitool
~^X	Suspend ipmitool, but don't restore tty on restart
~B	Send break
~~	Send the escape character by typing it twice
~?	Print the supported escape sequences

Note that escapes are only recognized immediately after newline.

deactivate

Deactivates Serial Over LAN mode on the BMC. Exiting Serial Over LAN mode should automatically cause this command to be sent to the BMC, but in the case of an unintentional exit from SOL mode, this command may be necessary to reset the state of the BMC.

spd <**i2cbus**> <**i2caddr**> [<**channel**>] [<axread>]

This command may be used to read SPD (Serial Presence Detect) data using the I2C Master Write−Read IPMI command.

sunoem

led

These commands provide a way to get and set the status of LEDs on a Sun Microsystems server. Use 'sdr list generic' to get a list of devices that are controllable LEDs. The *ledtype* parameter is optional and not necessary to provide on the command line unless it is required by hardware.

get <**sensorid**> [<**ledtype**>]

Get status of a particular LED described by a Generic Device Locator record in the SDR. A sensorid of *all* will get the status of all available LEDS.

set <**sensorid**> <**ledmode**> [<**ledtype**>]

Set status of a particular LED described by a Generic Device Locator record in the SDR. A sensorid of *all* will set the status of all available LEDS to the specified *ledmode* and *ledtype*.

LED Mode is required for set operations:

OFF	Off
ON	Steady On
STANDBY	100ms on 2900ms off blink rate
SLOW	1HZ blink rate
FAST	4HZ blink rate

LED Type is optional:

OK2RM	Ok to Remove
SERVICE	Service Required
ACT	Activity
LOCATE	Locate

fan speed <0-100>

Set system fan speed (PWM duty cycle).

sshkey

set <**userid**> <**keyfile**>

This command will allow you to specify an SSH key to use for a particular user on the Service Processor. This key will be used for CLI logins to the SP and not for IPMI sessions. View available users and their userids with the 'user list' command.

del <**userid**>

This command will delete the SSH key for a specified userid.

tsol

This command allows Serial-over-LAN sessions to be established with Tyan IPMIv1.5 SMDC such as the M3289 or M3290. The default command run with no arguments will establish default SOL session back to local IP address. Optional arguments may be supplied in any order.

<ipaddr>

Send receiver IP address to SMDC which it will use to send serial traffic to. By default this detects the local IP address and establishes two-way session. Format of ipaddr is XX.XX.XX.XX

port=NUM

Configure UDP port to receive serial traffic on. By default this is 6230.

ro|rw

Confiure SOL session as read-only or read-write. Sessions are read-write by default.

user

summary

Displays a summary of userid information, including maximum number of userids, the number of enabled users, and the number of fixed names defined.

list

Displays a list of user information for all defined userids.

set

name <**userid**> <**username**>

Sets the username associated with the given userid.

password <**userid**> [<**password**>]

Sets the password for the given userid. If no password is given, the password is cleared (set to the NULL password). Be careful when removing passwords from administrator−level accounts.

disable <**userid**>

Disables access to the BMC by the given userid.

enable <**userid**>

Enables access to the BMC by the given userid.

priv <**userid**> <**privilege level**> [<**channel number**>]

Set user privilege level on the specified channel. If the channel is not specified, the current channel will be used.

test <**userid**> <**16**|**20**> [<**password**>]

Determine whether a password has been stored as 16 or 20 bytes.

OPEN INTERFACE

The ipmitool *open* interface utilizes the OpenIPMI kernel device driver. This driver is present in all modern 2.4 and all 2.6 kernels and it should be present in recent Linux distribution kernels. There are also IPMI driver kernel patches for different kernel versions available from the OpenIPMI homepage.

The required kernel modules is different for 2.4 and 2.6 kernels. The following kernel modules must be loaded on a 2.4−based kernel in order for ipmitool to work:

ipmi_msghandler
Incoming and outgoing message handler for IPMI interfaces.

ipmi_kcs_drv
An IPMI Keyboard Controler Style (KCS) interface driver for the message handler.

ipmi_devintf
Linux character device interface for the message handler.

The following kernel modules must be loaded on a 2.6−based kernel in order for ipmitool to work:

ipmi_msghandler
Incoming and outgoing message handler for IPMI interfaces.

ipmi_si
An IPMI system interface driver for the message handler. This module supports various IPMI system interfaces such as KCS, BT, SMIC, and even SMBus in 2.6 kernels.

ipmi_devintf
Linux character device interface for the message handler.

Once the required modules are loaded there will be a dynamic character device entry that must exist at **/dev/ipmi0**. For systems that use devfs or udev this will appear at **/dev/ipmi/0**.

To create the device node first determine what dynamic major number it was assigned by the kernel by looking in **/proc/devices** and checking for the *ipmidev* entry. Usually if this is the first dynamic device it will be major number **254** and the minor number for the first system interface is **0** so you would create the device entry with:

mknod /dev/ipmi0 c 254 0

ipmitool includes some sample initialization scripts that can perform this task automatically at start−up.

In order to have ipmitool use the OpenIPMI device interface you can specifiy it on the command line:

ipmitool **−I** *open <command>*

BMC INTERFACE

The ipmitool bmc interface utilizes the *bmc* device driver as provided by Solaris 10 and higher. In order to force ipmitool to make use of this interface you can specify it on the command line:

ipmitool **−I** *bmc* <*command*>

The following files are associated with the bmc driver:

/platform/i86pc/kernel/drv/bmc
32−bit **ELF** kernel module for the bmc driver.

/platform/i86pc/kernel/drv/amd64/bmc
64−bit **ELF** kernel module for the bmc driver.

/dev/bmc
Character device node used to communicate with the bmc driver.

LIPMI INTERFACE

The ipmitool *lipmi* interface uses the Solaris 9 IPMI kernel device driver. It has been superceeded by the *bmc* interface on Solaris 10. You can tell ipmitool to use this interface by specifying it on the command line.

ipmitool **−I** *lipmi* <*expression*>

LAN INTERFACE

The ipmitool *lan* interface communicates with the BMC over an Ethernet LAN connection using UDP under IPv4. UDP datagrams are formatted to contain IPMI request/response messages with a IPMI session headers and RMCP headers.

IPMI−over−LAN uses version 1 of the Remote Management Control Protocol (RMCP) to support pre−OS and OS−absent management. RMCP is a request−response protocol delivered using UDP datagrams to port 623.

The LAN interface is an authenticatiod multi−session connection; messages delivered to the BMC can (and should) be authenticated with a challenge/response protocol with either straight password/key or MD5 message−digest algorithm. ipmitool will attempt to connect with administrator privilege level as this is required to perform chassis power functions.

You can tell ipmitool to use the lan interface with the **−I** *lan* option:

ipmitool **−I** *lan* **−H** <*hostname*> [**−U** <*username*>] [**−P** <*password*>] <*command*>

A hostname must be given on the command line in order to use the lan interface with ipmitool. The password field is optional; if you do not provide a password on the command line, ipmitool will attempt to connect without authentication. If you specify a password it will use MD5 authentication if supported by the BMC and straight password/key otherwise, unless overridden with a command line option.

LANPLUS INTERFACE

Like the *lan* interface, the *lanplus* interface communicates with the BMC over an Ethernet LAN connection using UDP under IPv4. The difference is that the *lanplus* interface uses the RMCP+ protocol as described in the IPMI v2.0 specification. RMCP+ allows for improved authentication and data integrity checks, as well as encryption and the ability to carry multiple types of payloads. Generic Serial Over LAN support requires RMCP+, so the ipmitool *sol activate* command requires the use of the *lanplus* interface.

RMCP+ session establishment uses a symmetric challenge−response protocol called RAKP (**Remote Authenticated Key−Exchange Protocol**) which allows the negotiation of many options. ipmitool does not yet allow the user to specify the value of every option, defaulting to the most obvious settings marked as required in the v2.0 specification. Authentication and integrity HMACS are produced with SHA1, and encryption is performed with AES−CBC−128. Role−level logins are not yet supported.

ipmitool must be linked with the *OpenSSL* library in order to perform the encryption functions and support the *lanplus* interface. If the required packages are not found it will not be compiled in and supported.

You can tell ipmitool to use the lanplus interface with the **−I** *lanplus* option:

ipmitool **−I** *lanplus* **−H** <*hostname*> [**−U** <*username*>] [**−P** <*password*>] <*command*>

A hostname must be given on the command line in order to use the lan interface with ipmitool. With the exception of the **−A** and **−C** options the rest of the command line options are identical to those available for the *lan* interface.

The **−C** option allows you specify the authentication, integrity, and encryption algorithms to use for for *lanplus* session based on the cipher suite ID found in the IPMIv2.0 specification in table 22−19. The default cipher suite is *3* which specifies RAKP−HMAC−SHA1 authentication, HMAC−SHA1−96 integrity, and AES−CBC−128 encryption algorightms.

FREE INTERFACE

The ipmitool *free* interface utilizes the FreeIPMI libfreeipmi drivers.

You can tell ipmitool to use the FreeIPMI interface with the -I option:

ipmitool **−I** *free* <*command*>

IMB INTERFACE

The ipmitool *imb* interface supports the Intel IMB (Intel Inter-module Bus) Interface through the /dev/imb device.

You can tell ipmitool to use the IMB interface with the -I option:

ipmitool **−I** *imb* <*command*>

EXAMPLES

Example 1: Listing remote sensors

```
> ipmitool −I lan −H 1.2.3.4 −f passfile sdr list
Baseboard 1.25V  | 1.24 Volts      | ok
Baseboard 2.5V   | 2.49 Volts      | ok
Baseboard 3.3V   | 3.32 Volts      | ok
```

Example 2: Displaying status of a remote sensor

```
> ipmitool −I lan −H 1.2.3.4 −f passfile sensor get "Baseboard 1.25V"
Locating sensor record...
Sensor ID              : Baseboard 1.25V (0x10)
Sensor Type (Analog)   : Voltage
Sensor Reading         : 1.245 (+/− 0.039) Volts
Status                 : ok
Lower Non−Recoverable  : na
Lower Critical         : 1.078
Lower Non−Critical     : 1.107
Upper Non−Critical     : 1.382
Upper Critical         : 1.431
Upper Non−Recoverable  : na
```

Example 3: Displaying the power status of a remote chassis

```
> ipmitool −I lan −H 1.2.3.4 −f passfile chassis power status
Chassis Power is on
```

Example 4: Controlling the power on a remote chassis

```
> ipmitool −I lan −H 1.2.3.4 −f passfile chassis power on
Chassis Power Control: Up/On
```

AUTHOR

Duncan Laurie <duncan@iceblink.org>

SEE ALSO

IPMItool Homepage
http://ipmitool.sourceforge.net

Intelligent Platform Management Interface Specification
http://www.intel.com/design/servers/ipmi

OpenIPMI Homepage
http://openipmi.sourceforge.net

FreeIPMI Homepage
http://www.gnu.org/software/freeipmi/