

# IPMItool

## Command-line Management of Intelligent Devices (<http://ipmitool.sourceforge.net>)

**Duncan Laurie**  
Sun Microsystems  
Linux Software Engineering

[duncan@sun.com](mailto:duncan@sun.com)

### Intelligent Platform Management Interface

Platform Management refers to the monitoring, logging, recovery and inventory control features implemented in hardware and firmware. The key differentiator of *Intelligent* Platform Management is that these functions are independent of the main CPU, BIOS, and OS.

There are two major components of platform management: the Service Processor or Baseboard Management Controller (BMC) and System Management Software (SMS). The service processor is the brain behind platform management and its primary purpose is to provide autonomous sensor monitoring and event logging features. Typical sensor-related events are out-of-range temperature or voltage and fan failure. When an event occurs it is noted in the system event log and made available to SMS. The service processor can even be wired to a standby power source and function when the server is powered down or the operating system has crashed. This allows platform status to be obtained and recovery initiated under situations where in-band delivery mechanisms are unavailable.

In modern systems, the Intelligent Platform Management Interface provides a hardware level interface specification for monitoring and control functions. It defines a standard, abstract, message-based interface between the BMC and SMS and a common set of commands for operations such as accessing sensor values, setting thresholds, logging events, and controlling a watchdog timer. IPMI messages can be used to communicate with the BMC over serial and LAN interfaces, so software designed for in-band (local) management can be re-used for out-of-band (remote) management simply by changing the low-level communications layer.

### Introduction to IPMItool

IPMItool is a simple command-line interface to systems that support the Intelligent Platform Management Interface v1.5 specification. It provides the ability to read the sensor data repository and print sensor values, display the contents of the system event log, print field replaceable unit information, read and set LAN configuration parameters, and perform remote chassis power control. It was originally written to take advantage of IPMI-over-LAN interfaces but is also capable of using the system interface as provided by a kernel device driver such as OpenIPMI. IPMItool is available under a BSD-compatible license.

System Management Software is generally 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 and systems that can be micro-managed. IPMItool takes a different approach to SMS and provides a completely command-line oriented tool. Therefore, it is not designed to replace the OpenIPMI 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.

## Interfaces

IPMItool supports dynamic loading of *interfaces* that correspond to low-level communication methods for accessing IPMI systems. The most common of these 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, but they are all similar enough to enable a single driver like OpenIPMI to support them all. They can be connected to any system bus such as ISA or X-bus that allows the main processor to access I/O mapped locations and meet the timing specifications. The varieties of system interfaces include Keyboard Controller Style (KCS), System Management Interface Chip (SMIC), and Block Transfer (BT). All of these 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.

### LAN Interface

This is often referred to as “IPMI-over-LAN” and defines how IPMI messages can be sent to and from the BMC encapsulated in Remote Management Control Protocol (RMCP) packets when are then transferred as UDP datagrams. Using the RMCP packet format increases the compatibility between management applications in heterogeneous environments. IPMI-over-LAN is only supported with version 1.5 and higher of the IPMI specification, and IPMItool was written specifically to take advantage of this interface. To use it provide the “-I lan” option to IPMItool and specify a hostname and password.

## Features

### Sensors

Instead of directly accessing the monitoring hardware, 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 termination voltage sensors. The amount of data available for each sensor can be overwhelming, so by default IPMItool only displays 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 Sensor Data Records (SDR) that are kept in a single centralized non-volatile storage area. These records include software information such as how many sensors are present, what type they are, their events, threshold info and more. This allows software to interpret and present sensor data without any prior knowledge about the platform. There is also support for writing your own SDR records

for OEM extensions, however any non-standard sensor records will not be able to be parsed by IPMI compliant software without specific knowledge of the data format.

### Example 1. Output from the *sdr list* command

```
Baseboard 1.25V | 1.245 Volts | ok
Baseboard 2.5V | 2.489 Volts | ok
Baseboard 3.3V | 3.320 Volts | ok
Baseboard Temp | 40 degrees C | ok
FntPnl Amb Temp | 25 degrees C | ok
Processor1 Temp | 46 degrees C | ok
Processor2 Temp | 45 degrees C | ok
```

### Example 2. Verbose output from the *sdr list* command

```
Sensor ID           : Baseboard 1.25V (0x11)
Entity ID           : 7.1 (System Board)
Sensor Type (Analog) : Voltage
Sensor Reading       : 1.245 Volts (+/- 0.039)
Status              : ok
Nominal Reading      : 1.245
Normal Minimum       : 1.078
Normal Maximum       : 1.411
Upper critical       : 1.490
Upper non-critical   : 1.450
Lower critical        : 1.019
Lower non-critical   : 1.049
```

## Events

Events are special messages sent by management controller when they detect system management events. Some examples of events are “temperature threshold exceeded”, “voltage threshold exceeded”, “correctable ECC memory error”, etc. These events are processed and usually logged in the System Event Log or SEL. This is similar to the SDR in that it provides a centralized non-volatile storage area for platform events that are logged autonomously by the BMC 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 the event and the sensor that triggered it. Detailed information is available with the verbose option.

### Example 3. Output from the *sel list* command

```
123 | 12/09/2003 | 19:36:03 | Fan #0x41 | Lower Critical - going low
124 | 12/09/2003 | 20:17:38 | Memory #0x08 | Correctable ECC
125 | 12/14/2003 | 06:04:03 | Physical Security #0x05 | System unplugged from LAN
126 | 01/06/2004 | 09:44:07 | Button #0x84 | Power Button pressed
```

## Inventory

IPMI supports multiple sets of non-volatile Field Replaceable Unit (FRU) information for different parts in the system. This provides access to data such as serial number, part number, asset tag, and other info for major modules in the system including the baseboard, chassis, processors, memory, power supplies, and even the management controller itself. This information is even available when the system is powered down or non-operational, facilitating 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.

### Example 4. Output from the *fru print* command

```
Chassis Type       : Rack Mount Chassis
Chassis Part      : F540-5746-01
Chassis Serial    : AZCW3110041
Board Mfg        : Sun Microsystems
Board Product     : SE7501WV2
Board Serial      : 0007E905C1AA103IMWE
Board Part       : A99386-109
Product Mfg      : Sun Microsystems
Product Name     : Sun Fire(tm) V60
```

## Chassis Management

This feature provides standardized chassis status and control functions that allow a remote system to be turned on/off or rebooted without manual intervention. It also provides commands for causing the chassis to physically identify itself with an implementation-dependant mechanism such as turning on visible lights, displaying message on an LCD, emitting beeps through a speaker, etc. IPMItool fully supports the available chassis management commands and can eliminate trips to the data center or server room to reset a frozen machine or help identify the single system in a rack that must be removed.

### Example 5. Sample *chassis power* commands

```
$ ipmitool -I lan -H sunfire -P password chassis power status
Chassis Power is on
```

```
$ ipmitool -I lan -H sunfire -P password chassis power off
Chassis Power Control: Down/Off
```

## Conclusion

This document highlights some of the more interesting features of IPMItool, but in reality it barely scratches the surface of what can be done. In all of the above examples only a portion of the available output has been shown, the full output is much richer and tells a full story about the system health and status. IPMItool is an actively developed project with multiple contributing developers who are working to increase its usefulness and functionality. Please refer to the IPMItool homepage (<http://ipmitool.sourceforge.net>) for more information about this program and to download source or pre-built binaries.