# FreeIPMI Frequently Asked Questions

Free Intelligent Platform Management System
Version 1.6.15 updated on 5 April 2025

**by Albert Chu** chu11@llnl.gov

# Table of Contents

## 0.1 What is IPMI?

The IPMI specifications define standardized, abstracted interfaces to the platform management subsystem. IPMI includes the definition of interfaces for extending platform management between the board within the main chassis and between multiple chassis.

The term platform management is used to refer to the monitoring and control functions that are built in to the platform hardware and primarily used for the purpose of monitoring the health of the system hardware. This typically includes monitoring elements such as system temperatures, voltages, fans, power supplies, bus errors, system physical security, etc. It includes automatic and manually driven recovery capabilities such as local or remote system resets and power on/off operations. It includes the logging of abnormal or out-of-range conditions for later examination and alerting where the platform issues the alert without aid of run-time software. Lastly it includes inventory information that can help identify a failed hardware unit.

## 0.2 What is FreeIPMI?

FreeIPMI is a collection of Intelligent Platform Management IPMI system software. It provides in-band and out-of-band software and a development library conforming to the Intelligent Platform Management Interface (IPMI v1.5 and v2.0) standards. FreeIPMI also supports IPMI-related specifications such as the Data Center Management Interface (DCMI) and Intel Node Manager.

## 0.3 How did FreeIPMI start?

In October 2003, California Digital Corp. (CDC) was contracted by Lawrence Livermore National Laboratory (LLNL) for the assembly of Thunder, a 1024 node Itanium2 cluster. This led to software developers from CDC and LLNL merging the IPMI software developed by both organizations into FreeIPMI.

Anand Babu, Balamurugan and Ian Zimmerman at CDC contributed the in-band KCS driver, `ipmi-sensors`, `ipmi-sel`, `bmc-info`, core portions of `ipmi-config`, and portions of libfreeipmi. Albert Chu and Jim Garlick at LLNL contributed `ipmipower`, `bmc-watchdog`, `ipmiping`, `rmcpping`, portions of libfreeipmi, and IPMI support in Powerman. In October 2004, FreeIPMI 0.1.0 was officially released.

Since the 0.1.0 release, Z Research developers have contributed `ipmi-chassis`, `ipmi-raw`, `ipmi-locate`, and PEF portions of `ipmi-config`. LLNL has contributed IPMI 2.0 support, hostrange support, `ipmiconsole`, libipmiconsole, `ipmidetect`, `bmc-device`, `ipmi-oem`, `ipmi-dcmi`, libipmimonitoring, and the chassis and sensor portions of `ipmi-config`.

(Note: The original FreeIPMI developers from California Digital Corp. are now at Zresearch Inc.)

## 0.4 What operating systems does FreeIPMI run on?

FreeIPMI was originally developed on GNU/Linux. It has been confirmed to be built on most major GNU/Linux distributions such as Redhat, Fedora, Suse, and Debian. FreeIPMI has been ported and confirmed to work on atleast FreeBSD, OpenBSD, Solaris, OpenSolaris, and Windows via Cygwin. We imagine it would build cleanly on other operating systems.

If it doesn't, it should be easily portable to them. Please contact the maintainers on the freeipmi-devel@gnu.org mailing lists.

## 0.5 FreeIPMI vs OpenIPMI vs Ipmitool vs Ipmiutil

There are multiple implementations, APIs, interfaces, end user requirements, etc. that one can choose when developing IPMI drivers, libraries, and tools. FreeIPMI has taken some different approaches than other open-source projects.

The section below points out a number of the reasons why we feel FreeIPMI is particularly special compared to the other projects.

The Ipmiutil project has a good chart describing many of the differences between the projects: http://ipmiutil.sourceforge.net/docs/ipmisw-compare.htm.

## 0.6 What is special about FreeIPMI?

In our eyes, there are several reasons why FreeIPMI is particularly special.

1. Support for HPC, clusters, and large data centers

   A number of features have been added into the tools to support HPC, clusters, and/or large data centers. Much of this original support was added to support the large cluster environments at Lawrence Livermore National Laboratory (LLNL).

   Scalable parallel execution of many FreeIPMI tools (`ipmi-sensors`, `ipmi-sel`, `bmc-info`, etc.) across a cluster is supported through hostranged input and output. For example:

```
# > bmc-info -h "pwopr[0-5]" -u XXX -p XXX --get-device-id -C
----------------
pwopr[0-1,5]
----------------
Device ID              : 34
Device Revision        : 1
Device SDRs            : unsupported
Firmware Revision      : 1.0c
Device Available       : yes (normal operation)
IPMI Version           : 2.0
Sensor Device          : supported
SDR Repository Device  : supported
SEL Device             : supported
FRU Inventory Device   : supported
IPMB Event Receiver    : unsupported
IPMB Event Generator   : unsupported
Bridge                 : unsupported
Chassis Device         : supported
Manufacturer ID        : Peppercon AG (10437)
Product ID             : 4
Auxiliary Firmware Revision Information : 38420000h
----------------
pwopr[2-4]
```

```
----------------
Device ID              : 34
Device Revision        : 1
Device SDRs            : unsupported
Firmware Revision      : 1.17
Device Available       : yes (normal operation)
IPMI Version           : 2.0
Sensor Device          : supported
SDR Repository Device  : supported
SEL Device             : supported
FRU Inventory Device   : supported
IPMB Event Receiver    : unsupported
IPMB Event Generator   : unsupported
Bridge                 : unsupported
Chassis Device         : supported
Manufacturer ID        : Peppercon AG (10437)
Product ID             : 4
Auxiliary Firmware Revision Information : 38420000h
```

In the above example, its clear to see that pwopr[2-4] have different firmware than pwopr[0-1,5]. More information about hostrange support can be found in the document `freeipmi-hostrange.txt` (http://www.gnu.org/software/freeipmi/freeipmi-hostrange.txt).

`Ipmipower` is capable of scaling to large nodes for cluster support and is supported by Powerman (https://github.com/chaos/powerman) for scalable power management. At LLNL, in conjunction with Powerman, ipmipower is used for power control on clusters ranging from sizes of 4 to 2000. It has been used to determine power status or power control LLNL's largest clusters in under a second.

`libipmiconsole` is currently supported by Conman (https://github.com/dun/conman) and Conserver (http://www.conserver.com/) for scalable console management.

`Ipmi-sensors` and libipmimonitoring are capable of interpreting sensor readings as well as just reporting them. It can be used for host monitoring IPMI sensor severity on a cluster. By mapping sensor readings into NOMINAL, WARNING, or CRITICAL states, it makes monitoring sensors easier across large numbers of nodes. Skummee (http://sourceforge.net/projects/skummee) currently uses libipmimonitoring to monitoring sensors on LLNL clusters of up to 2000 nodes in size. FreeIPMI sensor monitoring plugins for Ganglia (http://ganglia.info/) and Nagios (http://www.nagios.org/) have also been developed and made available for download (http://www.gnu.org/software/freeipmi/download.html).

`Ipmi-sel` and libipmimonitoring are capable of interpreting system event log (SEL) entries as well as just reporting them. It can be used for host monitoring IPMI event severity on a cluster. By mapping events into NOMINAL, WARNING, or CRITICAL states, it makes monitoring system events easier across large numbers of nodes. Skummee (http://sourceforge.net/projects/skummee) currently uses libipmimonitoring to monitoring the SEL on LLNL clusters of up to 2000 nodes in size.

The `ipmi-config` configuration file and command-line interface are used to easily copy the BMC configuration from one node to every other node in a cluster quickly. It has been used to modify the BMC configuration across large LLNL clusters in a few minutes. They also have the capability to verify (via the diff option) that the desired configuration has been properly stored to firmware.

`Ipmidetect` can be used to enhance the efficiency of the hostranged input by eliminating those nodes in the cluster that have been temporarily removed for servicing.

FreeIPMI is supported within Slurm for energy consumption monitoring.

2. Additional OEM support

   FreeIPMI contains support for a number of OEM extensions and OEM sensors and/or events. `Ipmi-oem` currently supports OEM command extensions for motherboards made by Dell, Fujitsu, IBM, Intel, Inventec, Quanta, Sun Microsystems, Supermicro, and Wistron. `Ipmi-sensors` and `ipmi-sel` support OEM sensors and/or events for motherboards made from Dell, Fujitsu, HP, Intel, Inventec, Quanta, Sun Microsystems, Supermicro, and Wistron. (Some of the motherboards may have been rebranded by vendors, see manpages for official list of confirmed supported motherboards.)

3. Additional flexibility and features

   By implementing various IPMI sub-sections into multiple tools, each tool is capable of providing the user with more flexibility and ultimately more features in addition to those listed above. It may not be as easy (or architecturally possible) to do in an all-in-one tool.

4. Extra IPMI support

   In addition to the features listed above, FreeIPMI also supports specifications related to IPMI. The Data Center Management Interface, or DCMI, is supported via the FreeIPMI tool `ipmi-dcmi`. Some aspects of the Intel Power Node Manager are supported in `ipmi-oem`.

5. Easy setup

   By implementing drivers in userspace libraries, there is no need to build/setup/manage any kernel modules/drivers.

6. Portability

   Likewise, by implementing everything in userspace libraries and tools, portability to multiple operating systems and architectures should be easier.

## 0.7 Does my system support IPMI?

Unfortunately, there are no universally defined mechanisms for determining if a system supports IPMI via Inband communication. Assuming IPMI is set up correctly for over LAN communication, a fairly reliable mechanism exists out-of-band. Here are some suggestions.

1. FreeIPMI's `ipmi-locate` can be used to determine if IPMI can be found on your system. Users are cautioned though, the failure to discover IPMI via `ipmi-locate` is not sufficient to disprove that IPMI exists on your system. Your system may not publish such information or may expect clients to communicate at default locations.

2. `dmidecode` may be similarly used to probe for devices that support IPMI on your system. You may grep for IPMI or specify the IPMI DMI type on the command line.

```
# > dmidecode --type 38
# dmidecode 2.10
SMBIOS 2.5 present.

Handle 0x0049, 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: 0x0000000000000CA2 (I/O)
        Register Spacing: Successive Byte Boundaries
```

3. FreeIPMI's `ipmi-ping` can be used to see if a machine has an IPMI service at a specific host/IP address. For more wide scale IPMI discovery, the `ipmi-detectd` daemon and `ipmi-detect` tool can be used.

Again, the failure to find an IPMI supported device is not sufficient to show lack of IPMI support.

Ultimately, some amount of information from product documents or trial and error may be necessary to determine if IPMI is supported on your system.

## 0.8 How do I compile FreeIPMI?

Please see the README.build instructions provided with FreeIPMI or on the FreeIPMI website's documentation (http://www.gnu.org/software/freeipmi/README.build).

## 0.9 libgcrypt requirement

FreeIPMI requires the libgcrypt library to be installed for a variety of encryption requirements in IPMI 2.0. If you are building FreeIPMI and receive a 'libgcrypt required to build libfreeipmi' error, please install libgcrypt. For Linux users, this may require the install of the libgcrypt-devel package as well. For those who do not need IPMI 2.0 encryption, FreeIPMI may be built without it by specifying `--without-encryption` when executing configure.

## 0.10 x86-64 Compilation

By default, FreeIPMI's build autotools (e.g. configure) should detect if you are on a 64 bit system and should build against 64 bit libraries. However, some multi-architecture installs (e.g. you have 32 bit and 64 bit libraries installed) may lead to builds and installs of 32 bit instead of 64 bit. For those noticing this, pass libdir appropriately to the configure script to workaround this problem. (e.g. `--libdir=/usr/lib64`)

Example:

```
# ./configure  --prefix=/usr --libdir=/usr/lib64
```

## 0.11 Installing FreeIPMI on FreeBSD

You can install a binary package of freeipmi or use the port, located in ports/sysutils/freeipmi, to build it from the source. See ports(7) and 'Packages

and Ports' section (http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/ports.html) in The FreeBSD Handbook.

Please contact port maintainer (MAINTAINER line in the port's Makefile), if you have problems building from the port.

## 0.12 What are some IPMI terminology or acronyms I should be aware of?

Good question, here are some terms and acronyms with general definition you might want to know.

**BMC**

The **Baseboard Management Controller** is the management chip on the system that is responsible for IPMI. It is common to refer to configuring the "BMC" as synonymous for configuring IPMI.

**inband**

**inband** IPMI communication refers to communication on a system locally (i.e. not over a network).

**outofband**

**outofband** and **IPMI over LAN** refer to IPMI communication over a network, typically ethernet.

**SDR**

The **Sensor Data Repository** is a database of system information that is needed by many other IPMI functions. It is commonly read before some IPMI action can be taken. For example, it contains a list of all sensors on a system, so it must be downloaded before sensors on a system can be read. In FreeIPMI, the SDR is cached in a common location and can be used by a number of tools, such as `ipmi-sensors`, `ipmi-sel`, and `ipmi-fru`.

**SEL**

The **System Event Log** is a log of events stored on the system for later diagnostics. In FreeIPMI, `ipmi-sel` can be used to read the SEL.

**FRU**

The **Field Replaceable Unit** is a general computing term referring to a replaceable unit of electronics. In IPMI it is common to refer to the "FRU" as the database of all FRU components on a system. In FreeIPMI, `ipmi-fru` can be used to read the FRU components on a system.

**PEF**

**Platform Event Filtering** refers to the rules that determine when PETs are generated and where they are sent. In FreeIPMI, PEF can be configured via `ipmi-config`.

**PET**

**Platform Event Trap** refers to a trap that can be sent by a system to an SNMP agent to indicate an event has occurred on the system. In FreeIPMI, a PET trap can be interpreted via `ipmi-pet`.

**DCMI**

The **Data Center Management Interface** is a management interface defined by a group of vendors that use IPMI as the backend for their system management definition. In FreeIPMI, `ipmi-dcmi` can be used to read/configuring DCMI.

**SOL**

**Serial over LAN** refers to the forwarding of serial system traffic over a network, typically an ethernet network. It is typically used to access a remote system console. In FreeIPMI, ipmiconsole is used to access a remote console via SOL.

**RMCP**

The **Remote Management Control Protocol** protocol is another remote management protocol which IPMI is defined within for outofband communication. For most IPMI users, you will never need to know about RMCP.

## 0.13 What setup is needed for FreeIPMI to communicate over LAN?

Please see the ipmi-config.conf(5) manpage provided with FreeIPMI for details, or you can read it on the FreeIPMI website's documentation (http://www.gnu.org/software/freeipmi/manpages/man5/ipmi-config.conf.5.html).

## 0.14 What setup is needed for Serial over LAN (SOL) or Ipmiconsole?

The setup of Serial-over-LAN (SOL) and/or `Ipmiconsole` is highly dependent on your system. However, most motherboardss require the following:

1. Adjust the BIOS COM port for serial redirection over SOL instead of the normal serial port and set the appropriate baud rate. If you do not know which port is the SOL port, you may need to play around and guess. It is likely a non-default setting, since many manufacturers may still assume the default redirection is out of the normal serial port. If you do not have a serial port on your motherboard, this part can probably be skipped.
2. Configure IPMI on the motherboard to use SOL. Many motherboards may have this enabled by default, however you may wish to verify with FreeIPMI's `ipmi-config`. More information can be found in the ipmi-config.conf(5) manpage on the settings. However, the key settings are to enable SOL on the system, enable SOL for individual users, and select the appropriate baud. On many motherboards, the selected baud must match what is configured in the BIOS.
3. Adjust your operating systems serial console settings to use the appropriate COM port. For Linux, the following guide (http://www.vanemery.com/Linux/Serial/serial-console.html) provides a pretty good overview of setting of a serial console on Linux. The only difference for setting up a serial console with `Ipmiconsole` or SOL, is the ttySX terminal may need to be changed.

## 0.15 Do I need to install or configure a driver to perform IPMI inband?

For most people the answer is no.

FreeIPMI includes a userspace driver that works on most motherboards without any driver installation, loading, or configuration required. FreeIPMI also includes support of a Linux SSIF driver through the SSIF device (i.e. `/dev/i2c-0`), the OpenIPMI Linux kernel driver (i.e. `/dev/ipmi0`), the Sun/Solaris BMC driver (i.e. `/dev/bmc`), and the Intel DCMI/MEI driver (i.e. `/dev/dcmi`). If you communicate through one of these mechanisms, the appropriate drivers for them should be loaded. Most systems should automatically load the appropriate drivers you need.

Under most scenarios, the FreeIPMI tools should automatically discover which in-band interface to use and the proper settings to use. Some motherboards may require you to determine alternate configurations for addresses, paths, etc. on your own and pass them as command line options to the tools.

Every system is different and your situation may differ. Please see your manufacturer and operating system instructions.

Special note: At the time of this writing the Intel DCMI/MEI Linux device drivers are not distributed widely. Please work with your vendor to obtain the Intel MEI and DCMI device drivers.

There are some additional Linux OpenIPMI kernel driver notes here: http://www.gnu.org/software/freeipmi/README.openipmi.

## 0.16 SSIF Driver Configuration

FreeIPMI's SSIF driver works on top of kernel'2 i2c device interface.

Under GNU/Linux load these kernel modules: i2c-dev, i2c-i801, i2c-core before using FreeIPMI.

To identify SSIF device address:

Example:

```
$> lspci  (in the output look for this entry)
 00:1f.3 SMBus: Intel Corp. 6300ESB SMBus Controller (rev 01)
       Subsystem: Intel Corp.: Unknown device 342f
       Flags: medium devsel,  IRQ 17
       I/O ports at 0400 [size=32]
                    ----
$> cat /proc/bus/i2c
i2c-0   smbus    SMBus I801 adapter at 0400         Non-I2C SMBus adapter
                                          ----
    Make sure the "0400" above matches with the "0400" address under
    proc.  Also make sure "i2c-0" is not different. If it appears
    different then grep for "i2c-0" in this code "ipmitool.c" and
    change. "i2c-X" is the label assigned to each slave device attached on
    the i2c bus.

BMC address Locator:
    Refer to the SM BIOS IPMI Device Information Record
    Type 38,  record 06h and 08h. Use the value of record
    06h as the IPMBAddress and load the SMBus controller
```

```
driver at the address value read from record 08h.

Usual values for record 06h -> 0x42
Usual values for record 08h -> 0x400
```

## 0.17 How do you setup Powerman with ipmipower?

There are additional details in the Powerman (https://github.com/chaos/powerman) documentation, however here are the basics. In the powerman.conf file, you want to include the `ipmipower.dev` device file, setup an ipmipower device in co-process mode, then configure hosts to use that device.

```
include "/etc/powerman/ipmipower.dev"

device "ipmi0" "ipmipower" "/usr/sbin/ipmipower -h mynodes[0-10] |&"

node "mynodes[0-10]"  "ipmi0" "mynodes[0-10]"
```

You may wish to add some additional ipmipower configuration on the device line:

```
device "ipmi0" "ipmipower" "/usr/sbin/ipmipower --wait-until-on --wait-until-off -h my
```

although you will probably want to do some of this configuration (especially the username and password) in `freeipmi.conf`.

If you use an alternate set of hostnames for IPMI from the primary hostname, that can configured like this:

```
device "ipmi0" "ipmipower" "/usr/sbin/ipmipower -h altname[0-10] |&"

node "primaryname[0-10]"  "ipmi0" "altname[0-10]"
```

Configuration can be trickier if you want to configure Powerman to use ipmipower with an OEM extension specified through `--oem-power-type`. Many OEM extensions in ipmipower must include additional arguments, which can be passed in via the node argument. In addition, while `ipmipower` can take a host range as an additional argument, Powerman may not.

For example, the following would be suitable to configure OEM extension support the Dell Poweredge C410x. A specific node identifier is used to map to a specific node and additional argument (i.e. mynodes0-1 maps to mynodes0+1).

```
include "/etc/powerman/ipmipower.dev"

device "ipmi0" "ipmipower" "/usr/sbin/ipmipower -h mynodes[0-10]+[1-16] --oem-power-ty

node "mynodes[0-10]-1" "ipmi0" "mynodes[0-10]+1"
node "mynodes[0-10]-2" "ipmi0" "mynodes[0-10]+2"
node "mynodes[0-10]-3" "ipmi0" "mynodes[0-10]+3"
node "mynodes[0-10]-4" "ipmi0" "mynodes[0-10]+4"
node "mynodes[0-10]-5" "ipmi0" "mynodes[0-10]+5"
node "mynodes[0-10]-6" "ipmi0" "mynodes[0-10]+6"
node "mynodes[0-10]-7" "ipmi0" "mynodes[0-10]+7"
node "mynodes[0-10]-8" "ipmi0" "mynodes[0-10]+8"
```

```
node "mynodes[0-10]-9" "ipmi0" "mynodes[0-10]+9"
node "mynodes[0-10]-10" "ipmi0" "mynodes[0-10]+10"
node "mynodes[0-10]-11" "ipmi0" "mynodes[0-10]+11"
node "mynodes[0-10]-12" "ipmi0" "mynodes[0-10]+12"
node "mynodes[0-10]-13" "ipmi0" "mynodes[0-10]+13"
node "mynodes[0-10]-14" "ipmi0" "mynodes[0-10]+14"
node "mynodes[0-10]-15" "ipmi0" "mynodes[0-10]+15"
node "mynodes[0-10]-16" "ipmi0" "mynodes[0-10]+16"
```

As noted in the manpage, the Dell Poweredge C410x appears to have difficulty handling new slot power control requests until prior ones have completed. Users may wish to configure `ipmipower` with `--wait-until-on`, `--wait-until-off`, and consider using the `ipmipower-serial.dev` device file instead of `ipmipower.dev`.

## 0.18 How do you setup Conman with ipmiconsole or libipmiconsole?

There are additional details in the Conman (https://github.com/dun/conman) documentation and manpages, however here are some basics.

To configure Conman to connect via the ipmiconsole tool, Conman comes with an expect script named ipmiconsole.exp, typically installed into `/usr/lib/conman/exec/ipmiconsole.exp`. Consoles can be setup by adding lines to `conman.conf` like:

```
CONSOLE name="myserver"   dev="/usr/lib/conman/exec/ipmiconsole.exp myserver myusernam
```

One of the useful aspects of using the ipmiconsole.exp script is that the same configuration options you may have already configured into `freeipmi.conf`, may be loaded automatically when ipmiconsole is executed via this expect script.

However, as can be expected, scalability may be a problem as you must launch a process for every node in your cluster.

Conman is also capable of connecting to servers natively through the libipmiconsole library, so that no additional processes are launched. They can be configured as follows:

```
CONSOLE name="myserver" IPMIOPTS="U:myusername,P:mypassword" dev="ipmi:myserver"
```

on some older versions of Conman, you would instead use

```
CONSOLE name="myserver" IPMIOPTS="myusername,mypassword" dev="ipmi:myserver"
```

Please see the Conman documentation for current version options and additional configuration options available. Alternate defaults for libipmiconsole can also be set via the `libipmiconsole.conf` file.

One of the additional advantages of configuring Conman to use the libipmiconsole library natively is that Conman is able to detect and manage additional IPMI error cases.

## 0.19 How do you setup Conserver with libipmiconsole?

You can find more details in the conserver.cf (https://www.conserver.com/docs/conserver.cf.man.html) manpage.

Here is an example configuration entry from the conserver.cf file that should illustrate the basics:

```
break 3 { string '"\d\z"'; delay 250; }

default ipmisol {
  break 3;
  motd "Generic IPMI SOL. Use \"Ctrl+E c ?\" for help, \"Ctrl+E c l 0 <key>\" for SysR
  username myusername;
  type ipmi;
}

console node42 {
  include ipmisol;
  master conserver.example.com;
  password mypassword;
  host node42.bmc.example.com;
}
```

## 0.20 How do you setup Ganglia or Nagios to monitor IPMI sensors via FreeIPMI?

Scripts to monitor IPMI via FreeIPMI in Ganglia and Nagios have been developed and are downloadable on the FreeIPMI homepage (http://www.gnu.org/software/freeipmi/download.html). Instructions for setup can be found at the top of the scripts.

## 0.21 Why are times reported by FreeIPMI tools wrong?

Times reported by various FreeIPMI tools (such as `ipmi-sel`) are reported under the assumption that timestamps are written in localtime. This is by definition in the IPMI specification.

Whether or not a system truly stored the timestamps in localtime varies on many factors, such as the vendor, BIOS, and operating system.

If the times reported by the tool are off, there is a strong likelihood the time may be stored in GMT/UTC and needs to be converted into localtime. In FreeIPMI tools that have time outputs, the `--utc-to-localtime` option can be specified or the `utc-to-localtime` option can be specified in `freeipmi.conf`.

## 0.22 Why is the IPMI kernel driver faster than the KCS driver?

Internally the IPMI kernel driver chooses to spin while polling for a response from the base management controller (BMC) while the KCS driver elects to sleep between poll attempts. An operating system's scheduler granularity may be larger than the time it takes to perform a IPMI transaction, thus the wall clock time of the KCS driver is far worse than the IPMI kernel driver. FreeIPMI's KCS driver implements the sleep between poll attempts because it is believed to provide better overall system use. To force the KCS driver to have similar wall clock response time to the IPMI kernel driver, users can specify the 'spinpoll' workaround.

## 0.23 Why is the output from FreeIPMI different than another software?

Due to minor implementation differences and or incorrect IPMI firmware, the resulting output from FreeIPMI tools can differ from other software. Here are some of the more common inconsistencies that have been seen before. More inconsistencies can be seen/fixed by specifying a number of the workarounds available to many of the FreeIPMI tools.

- In FreeIPMI's `ipmi-sel` and `ipmi-sensors` there are options for FreeIPMI to interpret the SEL or sensor readings and give them a NOMINAL, WARNING, or CRITICAL status. Other IPMI software may have different interpretations for their sensors and/or SEL readings that map to NOMINAL, WARNING, or CRITICAL differently. These interpretations are configurable in FreeIPMI via the `freeipmi_interpret_sel.conf` and `freeipmi_interpret_sensor.conf` configuration files.
- In some IPMI software, sensor and/or FRU records are bridged by default and read off satellite controllers. In FreeIPMI they are not due to the discovery that many vendors do not implement their bridging correctly or publish invalid slave addresses in the SDR. In order to bridge sensors the `--bridge-sensors` option must be specified in `ipmi-sensors`. In order to bridge FRU records, the `--bridge-fru` must be specified in `ipmi-fru`.
- In some IPMI software, shared sensors may be read by default. In FreeIPMI's `ipmi-sensors`, they are not read by default due to discovery that too many systems implement this incorrectly. Shared sensors can be read by specifying the `--shared-sensors` option.
- In vendor provided IPMI software, OEM specific sensors, SEL records, or FRU records may be output correctly because the vendor is aware of how to properly read/output OEM specific information. FreeIPMI may not be able to do this by default. For the motherboards in which OEM information is known, it can be output using the `--interpret-oem-data`. This option is available in `ipmi-sensors`, `ipmi-sel`, and `ipmi-fru`.
- In some vendor IPMI software, sensor "names" are constructured through a combination of the multiple data in the SDR, rather than just the device name listed in the SDR. This can lead to different sensor names listed in tools like `ipmi-sensors` and `ipmi-sel`. In both tools, this can be alleviated through the use of the `--entity-sensor-names` option.
- On several HP systems (observed on HP Proliant DL380 G7 and HP ProLiant ML310 G5), the SDR lists sensors using inconsistent information. Some analog sensors are listed as discrete sensors or vice versa. This inconsistency, and implementation differences between `ipmi-sensors` and other IPMI software lead to different outputs. For example, this is one such sensor that was seen in `ipmi-sensors`:

  ```
  2  | Power Supply 1  | Power Supply | N/A        | N/A   | 'Presence detected'
  ```

  but this same sensor was see in `ipmitool` as

  ```
  Power Supply 1   | 120 Watts         | nc
  ```

  In this example, `ipmi-sensors` did not output a Watts reading but outputs the proper "Presence Detected" state. `Ipmitool` outputs the correct watts reading, but outputs the invalid non-critical "nc" state.

In FreeIPMI, this problem can be worked around using the 'discretereading' workaround flag.

- In FreeIPMI's `ipmi-fru`, all multirecord FRU entries are output by default. In `ipmitool` and perhaps other software, they are not. The resulting output from `ipmi-fru` is much larger than other software. To get similar output in `ipmitool`, the `-v` option must be set.
- In FreeIPMI's `ipmi-fru`, FRU record checksums are automatically checked and errors are output if a FRU record cannot assumed to be valid due to a failed checksum check. Other IPMI software has been shown to ignore the checksums and assume records are valid. If your system has invalid checksummed FRU entries, the 'skipchecks' workaround can be used to get around them.
- In FreeIPMI's `ipmi-sensors`, sensors may output an "OK" if no events are set. In other IPMI software (such as `ipmitool` and the command "sdr list"), "OK" means that a sensor was simply read correctly. The "OK" may not mean that the device behind the sensor is actually functioning properly. For example, here is an output from an `ipmitool` sdr list entry.

  ```
  PSU 1 Status      | 0x0b                | ok
  ```

  The 0x0b indicates that the power supply has errors (you normally want to see 0x00 or 0x01), however the sensor outputs "ok" because the sensor reading was read correctly. When using FreeIPMI's `ipmi-sensors`, the 0x0b is properly converted into the event messages indicating an error.

  ```
  54  | PSU 1 Status | Power Supply | N/A | N/A | 'Presence detected' 'Power Supply
  ```

## 0.24 Why are there so many IPMI compliance bugs?

The IPMI specification is very large. At last check, the IPMI specification was 601 pages. This does not count the various side specifications related to IPMI, including DCMI, PET, FRU, and the various OEM extension specifications (e.g. Intel Node Manager). Many sections of text can be ambiguous. Many components of IPMI are optional and aren't required to be implemented. There is some leeway for implementation interpretation as well. Ultimately, bugs will happen. In all fairness, FreeIPMI has had bugs too.

A number of the IPMI compliance bugs found by FreeIPMI are documented in the freeipmi-bugs-issues-and-workarounds.txt document (you can find it on the website here: http://www.gnu.org/software/freeipmi/freeipmi-bugs-issues-and-workarounds.txt).

## 0.25 How do I get around an IPMI compliance bug on my motherboard?

Most of the FreeIPMI tools and libraries have flags to workaround a large number of IPMI compliance bugs found on motherboards. Please see the appropriate tool manpages or library header files for details on the workarounds available and for what motherboards.

If you believe there is a compliance issue on your motherboard that has not yet been implemented, please contact the maintainers on the freeipmi-devel@gnu.org mailing list.

## 0.26 Why am I seeing so many 'internal IPMI error' or 'driver busy' messages?

In some Linux distributions (atleast with Redhat Enterprise Linux 6.4 / RHEL 6.4 and CentOS 6.4), the distributions began compiling the IPMI kernel driver (i.e. ipmi_msghandler and ipmi_si modules) into the kernel instead of as loadable modules. This was apparently due to a need for the IPMI kernel drivers to work with ACPI.

Due to compiling the IPMI kernel driver into the kernel, there is the potential for inband IPMI communication to occur in the kernel w/o any knowledge of it from outside software in userspace. Normally, the IPMI service (i.e. /etc/init.d/ipmi) is started to create a device file (i.e. /dev/ipmi0) so that userspace software will know to communicate through this device. However, some of these distros disable the ipmi service so that a device file is never created.

Because of this, multiple IPMI software can communicate inband to the BMC simultaneously, subsequently racing with each other. Ultimately, this can lead to communication problems. In FreeIPMI, this is most commonly seen through 'internal IPMI error' or 'driver busy' messages.

There are several possible solutions for this in FreeIPMI. If you start the ipmi service (i.e. /etc/init.d/ipmi start), a device file will be created which FreeIPMI will recognize. FreeIPMI will subsequently communicate via inband IPMI through this device file, thus eliminating racing with the IPMI occurring in the kernel.

The other option is to disable the IPMI kernel thread so that FreeIPMI can perform inband communication with the BMC through its own mechanisms. The following is from the Linux kernel documentation.

```
When compiled into the kernel, the parameters can be specified on the
kernel command line as:

  ipmi_si.type=<type1>,<type2>...
       ipmi_si.ports=<port1>,<port2>... ipmi_si.addrs=<addr1>,<addr2>...
       ipmi_si.irqs=<irq1>,<irq2>... ipmi_si.trydefaults=[0|1]
       ipmi_si.regspacings=<sp1>,<sp2>,...
       ipmi_si.regsizes=<size1>,<size2>,...
       ipmi_si.regshifts=<shift1>,<shift2>,...
       ipmi_si.slave_addrs=<addr1>,<addr2>,...
       ipmi_si.force_kipmid=<enable1>,<enable2>,...
       ipmi_si.kipmid_max_busy_us=<ustime1>,<ustime2>,...

...

If your IPMI interface does not support interrupts and is a KCS or
SMIC interface, the IPMI driver will start a kernel thread for the
interface to help speed things up.  This is a low-priority kernel
thread that constantly polls the IPMI driver while an IPMI operation
is in progress.  The force_kipmid module parameter will all the user to
force this thread on or off.  If you force it off and don't have
interrupts, the driver will run VERY slowly.
```

## 0.27 How do I program with the FreeIPMI libraries?

If you are looking for a high level library to do Serial-over-LAN (SOL) or IPMI sensor/SEL monitoring, you may wish to look at the libipmiconsole and libipmimonitoring libraries. These libraries attempt to abstract a large amount of the underlying IPMI detail from developers. The majority of the documentation can be found in the header files. Some examples can be found in the documentation and the FreeIPMI tools that use them.

The libfreeipmi library is the core library used by other FreeIPMI libraries and tools. However, it is quite detailed in regards to the IPMI specification and many components of the library will be quite confusing to those unfamiliar with the finer details of the IPMI specification. It is recommended most use the higher level libraries described above.

A more detailed description of the available FreeIPMI libraries can be found in the `freeipmi-libraries.txt` document (you can find it on the website here: http://www.gnu.org/software/freeipmi/freeipmi-libraries.txt).

## 0.28 Where can I get additional help or support?

For help, please email the freeipmi-users@gnu.org mailing list.