skiboot Documentation

Release 5.3.0-rc1

Stewart Smith, IBM, others

Sep 15, 2016

CONTENTS

Contents:

CHAPTER

ONE

SKIBOOT OVERVIEW

Skiboot is firmware, loaded by the FSP. Along with loading the bootloader, it provides some runtime services to the OS (typically Linux).

1.1 Source layout

Directory	Content
asm/	small amount, mainly entry points
ccan/	bits from CCAN
core/	common code among machines.
doc/	not enough here
external/	tools to run external of sapphire.
hdata/	all stuff going to/from FSP
hw/	drivers for things & fsp things.
include/	headers!
libc/	tiny libc, from SLOF
libfdt/	straight device tree lib
libpore/	to manipulate PORE engine.

We have a spinlock implementation in asm/lock.S Entry points are detailed in asm/head.S The main C entry point is in core/init.c: main_cpu_entry()

1.2 Binaries

The following binaries are built:

File	Purpose
skiboot.lid	is the actual lid. objdump out
skiboot.elf	is the elf binary of it, lid comes from this
skiboot.map	plain map of symbols

1.3 Booting

On boot, every thread of execution jumps to a single entry point in skiboot so we need to do some magic to ensure we init things properly and don't stomp on each other. We choose a master thread, putting everybody else into a spinloop.

Essentially, we do this by doing an atomic fetch and inc and whoever gets 0 gets to be the master.

When we enter skiboot we also get a memory location in a register which is the location of a device tree for the system. We flatten out the device tree, turning offsets into real pointers and manipulating it where needed. We re-flatten the device tree before booting the OS (Linux).

The main entry point is main_cpu_entry() in core/init.c, this is a carefully ordered init of things. The sequence is relatively well documented there.

1.4 OS interface

Skiboot maintains its own stack for each CPU. We do not have an ABI like "may use X stack on OS stack", we entirely keep to our own stack space. The OS (Linux) calling skiboot will never use any OS stack space and the OS does not need to call skiboot with a valid stack.

We define an array of stacks, one for each CPU. On entry to skiboot, we can find out stack by multiplying our CPU number by the stack size and adding that to the address of the stack area.

At the bottom of each stack area is a per CPU data structure, which we can get to by chopping off the LSBs of the stack pointer.

The OPAL interface is a generic message queue. The Linux side of things can be found in linux/arch/powerpc/platform/powernv/

1.5 Interrupts

We don't handle interrupts in skiboot.

In the future we may have to change to process machine check interrupts during boot.

We do not have timer interrupts.

1.6 Memory

We initially occupy a chunk of memory, "heap". We pass to the OS (Linux) a reservation of what we occupy (including stacks).

In the source file include/config.h we include a memory map. This is manually generated, not automatically generated.

We use CCAN for a bunch of helper code, turning on things like DEBUG_LOCKS and DEBUG_MALLOC as these are not a performance issue for us, and we like to be careful.

In include/config.h there are defines for turning on extra tracing. OPAL is what we name the interface from skiboot to OS (Linux).

Each CPU gets a 16k stack, which is probably more than enough. Stack should be used sparingly though.

Important memory locations:

Location	What's there
SKIBOOT_BASE	where skiboot lives, of SKIBOOT_SIZE
HEAP_BASE	Where skiboot heap starts, of HEAP_SIZE

There is also SKIBOOT_SIZE (manually calculated) and DEVICE_TREE_MAX_SIZE, which is largely historical.

1.7 Skiboot log

There is a circular log buffer that skiboot maintains. This can be accessed either from the FSP or through /dev/mem or through a debugfs patch that's currently floating around.

CHAPTER

TWO

OPAL SPECIFICATION

DRAFT - VERSION 0.0.1 AT BEST.

COMMENTS ARE WELCOME - and indeed, needed.

If you are reading this, congratulations: you're now reviewing it!

This document aims to define what it means to be OPAL compliant.

While skiboot is the reference implementation, this documentation should be complete enough that (given hardware documentation) create another implementation. It is not recommended that you do this though.

2.1 Authors

Stewart Smith <stewart@linux.vnet.ibm.com> : OPAL Architect, IBM

2.2 Definitions

Host processor - the main POWER CPU (e.g. the POWER8 CPU) Host OS - the operating system running on the host processor. OPAL - OpenPOWER Abstraction Layer.

2.3 What is OPAL?

The OpenPower Abstraction Layer (OPAL) is boot and runtime firmware for POWER systems. There are several components to what makes up a firmware image for OpenPower machines.

For example, there may be:

- BMC firmware
  - Firmware that runs purely on the BMC.
  - On IBM systems that have an FSP rather than a BMC, there is FSP firmware
  - While essential to having the machine function, this firmware is not part of the OPAL Specification.
- HostBoot
  - HostBoot (https://github.com/open-power/hostboot) performs all processor, bus and memory initialization within IBM POWER based systems.
- OCC Firmware

  - On Chip Controller (Firmware for OCC - a PPC405 core inside the IBM POWER8 in charge of keeping the system thermally and power safe).

- SkiBoot
  - Boot and runtime services.
- A linux kernel and initramfs incorporating petitboot
  - The bootloader. This is where a user chooses what OS to boot, and petitboot will use kexec to switch to the host Operating System (for example, PowerKVM).

While all of these components may be absolutely essential to power on, boot and operate a specific OpenPower POWER8 system, the majority of the code mentioned above can be thought of as implementation details and not something that should form part of an OPAL Specification.

For an OPAL system, we assume that the hardware is functioning and any hardware management that is specific to a platform is performed by OPAL firmware transparently to the host OS.

The OPAL Specification focus on the interface between firmware and the Operating System. It does not dictate that any specific pieces of firmware code be used, although re-inventing the wheel is strongly discouraged.

The OPAL Specification explicitly allows for:

- A conforming implementation to not use any of the reference implementation code.
- A conforming implementation to use any 64bit POWER ISA conforming processor, and not be limited to the IBM POWER8.
- A conforming implementation to be a simulator, emulator or virtual environment
- A host OS other than Linux

Explicitly not covered in this specification:

- A 32bit OPAL Specification There is no reason this couldn't exist but the current specification is for 64bit POWER systems only.

2.4 Boot Services

An OPAL compliant firmware implementation will load and execute a payload capable of booting a Host Operating System.

The reference implementation loads a Linux kernel with an initramfs with a minimal userspace and the petitboot boot loader - collectively referred to as skiroot.

The OPAL Specification explicitly allows variation in this payload.

A requirement of the payload is that it MUST support loading and booting an uncompressed vmlinux Linux kernel. [**TODO**: expand on what this actually means]

An OPAL system MUST pass a device tree to the host kernel. [**TODO**: expand the details, add device-tree section and spec]

An OPAL system MUST provide the host kernel with enough information to know how to call OPAL runtime services. [**TODO**: expand on this.]

Explicitly not covered by the OPAL Specification:

- Kernel module ABI for skiroot kernel
- Userspace environment of skiroot
- That skiroot is Linux.

Explicitly allowed:

- Replacing the payload with something of equal/similar functionality (whether replacing skiroot with an implementation of Zork would be compliant is left as an exercise for the reader)

2.5 Payload Environment

The payload is started with:

Register	Value
r3	address of flattened device-tree (fdt)
r8	OPAL base
r9	OPAL entry

2.6 Runtime Services

An OPAL Specification compliant system provides runtime services to the host Operating System via a standard interface.

An OPAL call is made by calling opal_entry with:

- r0: OPAL Token
- r2: OPAL Base
- r3..r10: Args (up to 8)

The OPAL API is defined in skiboot/doc/opal-api/

Not all OPAL APIs must be supported for a system to be compliant. When called with an unsupported token, a compliant firmware implementation MUST fail gracefully and not crash. Reporting a warning that an unsupported token was called is okay, as compliant host Operating Systems should use OPAL_CHECK_TOKEN to test for optional functionality.

All parameters to OPAL calls are big endian. Little endian hosts MUST appropriately convert parameters before passing them to OPAL.

Machine state across OPAL calls:

- r1 is preserved
- r12 is scratch
- r13 - 31 preserved
- 64bit HV real mode
- big endian
- external interrupts disabled

2.7 Detecting OPAL Support

A Host OS may need to detect the presence of OPAL as it may support booting under other platforms. For example, a single Linux kernel can be built to boot under OPAL and under PowerVM or qemu pseries machine type.

The root node of the device tree MUST have compatible = "ibm,powernv". See doc/device-tree.rst for more details [TODO: make doc/device-tree.rst better]

The presence of the "/ibm,opal" entry in the device tree signifies running under OPAL. Additionally, the "/ibm,opal" node MUST have a compatibile property listing "ibm,opal-v3".

The "/ibm,opal" node MUST have the following properties:

```
ibm,opal {
          compatible = "ibm,opal-v3";
          opal-base-address = <>;
          opal-entry-address = <>;
          opal-runtime-size = <>;
}
```

The compatible property MAY have other strings, such as a future "ibm,opal-v4". These are reserved for future use.

Some releases of the reference implementation (skiboot) have had compatible contain "ibm,opal-v2" as well as "ibm,opal-v3". Host operating systems MUST NOT rely on "ibm,opal-v2", this is a relic from early OPAL history.

The "ibm,opal" node MUST have a child node named "firmware". It MUST contain the following:

```
firmware {
        compatible = "ibm,opal-firmware";
}
```

It MUST contain one of the following two properties: git-id, version. The git-id property is deprecated, and version SHOULD be used. These are informative and MUST NOT be used by the host OS to determine anything about the firmware environment.

The version property is a textual representation of the OPAL version. For example, it may be "skiboot-4.1" or other versioning described in more detail in doc/versioning.rst

2.8 OPAL log

OPAL implementation SHOULD have an in memory log where informational and error messages are stored. If present it MUST be human readable and text based. There is a separate facility (Platform Error Logs) for machine readable errors.

A conforming implementation MAY also output the log to a serial port or similar. An implementation MAY choose to only output certain log messages to a serial port.

For example, the reference implementation (skiboot) by default filters log messages so that only higher priority log messages go over the serial port while more messages go to the in memory buffer.

[TODO: add device-tree bits here]

CHAPTER

THREE

DEVELOPER GUIDE AND INTERNALS

3.1 SkiBoot Console Log

Skiboot maintains a circular textual log buffer in memory.

It can be accessed using any debugging method that can peek at memory contents. While the debug_descriptor does hold the location of the memory console, we're pretty keen on keeping its location static.

Events are logged in the following format: [timebase,log_level] message

You should use the new prlog() call for any log message and set the log level/priority appropriately.

printf() is mapped to PR_PRINTF and should be phased out and replaced with prlog() calls.

See timebase.h for full timebase explanation.

Log level from skiboot.h:

Define	Value
PR_EMERG	0
PR_ALERT	1
PR_CRIT	2
PR_ERR	3
PR_WARNING	4
PR_NOTICE	5
PR_PRINTF	PR_NOTICE
PR_INFO	6
PR_DEBUG	7
PR_TRACE	8
PR_INSANE	9

The console_log_levels byte in the debug_descriptor controls what messages are written to any console drivers (e.g. fsp, uart) and what level is just written to the in memory console (or not at all).

This enables (advanced) users to vary what level of output they want at runtime in the memory console and through console drivers (fsp/uart)

You can vary two things by poking in the debug descriptor:

1. what log level is printed at all e.g. only turn on PR_TRACE at specific points during runtime
2. what log level goes out the fsp/uart console, defaults to PR_PRINTF

We use two 4bit numbers (1 byte) for this in debug descriptor (saving some space, not needlessly wasting space that we may want in future).

The default is 0x75 (7=PR_DEBUG to in memory console, 5=PR_PRINTF to drivers

If you write 0x77 you will get debug info on uart/fsp console as well as in memory. If you write 0x95 you get PR_INSANE in memory but still only PR_NOTICE through drivers.

People who write something like 0x1f will get a very quiet boot indeed.

3.2 How to log errors on OPAL

Currently the errors reported by OPAL interfaces are in free form, where as errors reported by service processor is in standard Platform Error Log (PEL) format. For out-of band management via IPMI interfaces, it is necessary to push down the errors to service processor via mailbox (reported by OPAL) in PEL format.

PEL size can vary from 2K-16K bytes, fields of which needs to populated based on the kind of event and error that needs to be reported. All the information needed to be reported as part of the error, is passed by user using the error-logging interfaces outlined below. Following which, PEL structure is generated based on the input and then passed on to service processor.

We do create eSEL error log format for some service processors but it's just a wrapper around PEL format. Actual data still stays in PEL format.

3.2.1 Error logging interfaces in OPAL

Interfaces are provided for the user to log/report an error in OPAL. Using these interfaces relevant error information is collected and later converted to PEL format and then pushed to service processor.

Step 1: To report an error, invoke `opal_elog_create()` with required argument.

```
struct errorlog *opal_elog_create(struct opal_err_info
*e_info,uint32_t tag);
```

Parameters:

- `struct opal_err_info *e_info` Struct to hold information identifying error/event source.
- **`uint32_t tag`: Unique value to identify the data.** Ideal to have ASCII value for 4-byte string.

The opal_err_info struct holds several pieces of information to help identify the error/event. The struct can be obtained via the `DEFINE_LOG_ENTRY` macro as below - it only needs to be called once.

```
DEFINE_LOG_ENTRY(OPAL_RC_ATTN, OPAL_PLATFORM_ERR_EVT, OPAL_CHIP,
                 OPAL_PLATFORM_FIRMWARE, OPAL_PREDICTIVE_ERR_GENERAL,
                 OPAL_NA);
```

The various attributes set by this macro are described below.

`uint8_t opal_error_event_type`: Classification of error/events type reported on OPAL.

```
/* Platform Events/Errors: Report Machine Check Interrupt */
#define OPAL_PLATFORM_ERR_EVT           0x01
/* INPUT_OUTPUT: Report all I/O related events/errors */
#define OPAL_INPUT_OUTPUT_ERR_EVT       0x02
/* RESOURCE_DEALLOC: Hotplug events and errors */
#define OPAL_RESOURCE_DEALLOC_ERR_EVT   0x03
/* MISC: Miscellaneous error */
#define OPAL_MISC_ERR_EVT               0x04
```

`uint16_t component_id`: Component ID of OPAL component as listed in `include/errorlog.h`.

`uint8_t subsystem_id`: ID of the sub-system reporting error.

```
/* OPAL Subsystem IDs listed for reporting events/errors */
#define OPAL_PROCESSOR_SUBSYSTEM        0x10
#define OPAL_MEMORY_SUBSYSTEM           0x20
#define OPAL_IO_SUBSYSTEM               0x30
#define OPAL_IO_DEVICES                 0x40
#define OPAL_CEC_HARDWARE               0x50
#define OPAL_POWER_COOLING              0x60
#define OPAL_MISC                       0x70
#define OPAL_SURVEILLANCE_ERR           0x7A
#define OPAL_PLATFORM_FIRMWARE          0x80
#define OPAL_SOFTWARE                   0x90
#define OPAL_EXTERNAL_ENV               0xA0
```

`uint8_t event_severity`: Severity of the event/error to be reported.

```
#define OPAL_INFO                                   0x00
#define OPAL_RECOVERED_ERR_GENERAL                  0x10

/* 0x2X series is to denote set of Predictive Error */
/* 0x20 Generic predictive error */
#define OPAL_PREDICTIVE_ERR_GENERAL                         0x20
/* 0x21 Predictive error, degraded performance */
#define OPAL_PREDICTIVE_ERR_DEGRADED_PERF                   0x21
/* 0x22 Predictive error, fault may be corrected after reboot */
#define OPAL_PREDICTIVE_ERR_FAULT_RECTIFY_REBOOT            0x22
/*
 * 0x23 Predictive error, fault may be corrected after reboot,
 * degraded performance
 */
#define OPAL_PREDICTIVE_ERR_FAULT_RECTIFY_BOOT_DEGRADE_PERF 0x23
/* 0x24 Predictive error, loss of redundancy */
#define OPAL_PREDICTIVE_ERR_LOSS_OF_REDUNDANCY              0x24

/* 0x4X series for Unrecoverable Error */
/* 0x40 Generic Unrecoverable error */
#define OPAL_UNRECOVERABLE_ERR_GENERAL                      0x40
/* 0x41 Unrecoverable error bypassed with degraded performance */
#define OPAL_UNRECOVERABLE_ERR_DEGRADE_PERF                 0x41
/* 0x44 Unrecoverable error bypassed with loss of redundancy */
#define OPAL_UNRECOVERABLE_ERR_LOSS_REDUNDANCY              0x44
/* 0x45 Unrecoverable error bypassed with loss of redundancy
 * and performance
 */
#define OPAL_UNRECOVERABLE_ERR_LOSS_REDUNDANCY_PERF         0x45
/* 0x48 Unrecoverable error bypassed with loss of function */
#define OPAL_UNRECOVERABLE_ERR_LOSS_OF_FUNCTION             0x48

#define OPAL_ERROR_PANIC                                    0x50
```

`uint8_t event_subtype`: Event Sub-type

```
#define OPAL_NA                                         0x00
#define OPAL_MISCELLANEOUS_INFO_ONLY                    0x01
#define OPAL_PREV_REPORTED_ERR_RECTIFIED                0x10
```

```
#define OPAL_SYS_RESOURCES_DECONFIG_BY_USER             0x20
#define OPAL_SYS_RESOURCE_DECONFIG_PRIOR_ERR            0x21
#define OPAL_RESOURCE_DEALLOC_EVENT_NOTIFY              0x22
#define OPAL_CONCURRENT_MAINTENANCE_EVENT               0x40
#define OPAL_CAPACITY_UPGRADE_EVENT                     0x60
#define OPAL_RESOURCE_SPARING_EVENT                     0x70
#define OPAL_DYNAMIC_RECONFIG_EVENT                     0x80
#define OPAL_NORMAL_SYS_PLATFORM_SHUTDOWN               0xD0
#define OPAL_ABNORMAL_POWER_OFF                         0xE0
```

`uint8_t opal_srctype`: SRC type, value should be OPAL_SRC_TYPE_ERROR. SRC refers to System Reference Code. It is 4 byte hexa-decimal number that reflects the current system state. Eg: BB821010,

- 1st byte -> BB -> SRC Type
- 2nd byte -> 82 -> Subsystem
- 3rd, 4th byte -> Component ID and Reason Code

SRC needs to be generated on the fly depending on the state of the system. All the parameters needed to generate a SRC should be provided during reporting of an event/error.

`uint32_t reason_code`: Reason for failure as stated in `include/errorlog.h` for OPAL. Eg: Reason code for code-update failures can be

- `OPAL_RC_CU_INIT` -> Initialisation failure
- `OPAL_RC_CU_FLASH` -> Flash failure

Step 2: Data can be appended to the user data section using the either of the below two interfaces:

```
void log_append_data(struct errorlog *buf, unsigned char *data,
                     uint16_t size);
```

Parameters:

`struct opal_errorlog *buf`: `struct opal_errorlog` pointer returned by `opal_elog_create()` call.

`unsigned char *data`: Pointer to the dump data

`uint16_t size`: Size of the dump data.

```
void log_append_msg(struct errorlog *buf,const char *fmt,...);
```

Parameters:

`struct opal_errorlog *buf`: pointer returned by `opal_elog_create()` call.

`const char *fmt`: Formatted error log string.

Additional user data sections can be added to the error log to separate data (eg. readable text vs binary data) by calling `log_add_section()`. The interfaces in Step 2 operate on the 'last' user data section of the error log.

```
void log_add_section(struct errorlog *buf,uint32_t tag);
```

Parameters:

`struct opal_errorlog *buf`: pointer returned by `opal_elog_create()` call.

`uint32_t tag`: Unique value to identify the data. Ideal to have ASCII value for 4-byte string.

Step 3: There is a platform hook for the OPAL error log to be committed on any service processor(Currently used for FSP and BMC based machines).

Below is snippet of the code of how this hook is called.

```
void log_commit(struct errorlog *elog)
{
        ....
        ....
        if (platform.elog_commit) {
                rc = platform.elog_commit(elog);
                if (rc)
                        prerror("ELOG: Platform commit error %d"
                                "\n", rc);
                return;
        }
        ....
        ....
}
```

Step 3.1 FSP:

```
.elog_commit            = elog_fsp_commit
```

Once all the data for an error is logged in, the error needs to be committed in FSP.

In the process of committing an error to FSP, log info is first internally converted to PEL format and then pushed to the FSP. FSP then take cares of sending all logs including its own and OPAL's one to the POWERNV.

OPAL maintains timeout field for all error logs it is sending to FSP. If it is not logged within allotted time period (e.g if FSP is down), in that case OPAL sends those logs to POWERNV.

Step 3.2 BMC:

```
.elog_commit            = ipmi_elog_commit
```

In case of BMC machines, error logs are first converted to eSEL format. i.e:

```
eSEL = SEL header + PEL data
```

SEL header contains below fields.

```
struct sel_header {
        uint16_t id;
        uint8_t record_type;
        uint32_t timestamp;
        uint16_t genid;
        uint8_t evmrev;
        uint8_t sensor_type;
        uint8_t sensor_num;
        uint8_t dir_type;
        uint8_t signature;
        uint8_t reserved[2];
}
```

After filling up the SEL header fields, OPAL copies the error log PEL data after the header section in the error log buffer. Then using IPMI interface, eSEL gets logged in BMC.

If the user does not intend to dump various user data sections, but just log the error with some amount of description around that error, they can do so using just the simple error logging interface.

```
log_simple_error(uint32_t reason_code, char *fmt, ...);
```

For example:

```
log_simple_error(OPAL_RC_SURVE_STATUS,
                 "SURV: Error retrieving surveillance status: %d\n",
                                                        err_len);
```

Using the reason code, an error log is generated with the information derived from the look-up table, populated and committed to service processor. All of it is done with just one call.

3.3 Error logging retrieval from FSP:

FSP sends error log notification to OPAL via mailbox protocol.

OPAL maintains below lists:

- Free list : List of free nodes.
- Pending list : List of nodes which is yet to be read by the POWERNV.
- **Processed list** [List of nodes which has been read but still waiting for] acknowledgement.

Below is the structure of the node:

```
struct fsp_log_entry {
        uint32_t log_id;
        size_t log_size;
        struct list_node link;
};
```

OPAL maintains a state machine which has following states.

```
enum elog_head_state {
        ELOG_STATE_FETCHING,    /*In the process of reading log from FSP. */
        ELOG_STATE_FETCHED_INFO,/* Indicates reading log info is completed */
        ELOG_STATE_FETCHED_DATA,/* Indicates reading log is completed */
        ELOG_STATE_HOST_INFO,   /* Host read log info */
        ELOG_STATE_NONE,        /* Indicates to fetch next log */
        ELOG_STATE_REJECTED,    /* resend all pending logs to linux */
};
```

Initially, state of the state machine is `ELOG_STATE_NONE`. When OPAL gets the notification about the error log, it takes out the node from free list and put it into pending list and update the state machine to fetching state (`ELOG_STATE_FETCHING`). It also gives response back to FSP about the received error log notification.

It then queue mailbox message to get the error log data in OPAL error log buffer, once it is done state machine gets into fetched state (`ELOG_STATE_FETCHED_DATA`). After that, OPAL notifies POWERNV host to fetch new error log.

POWERNV uses the OPAL interface to get the error log info(elogid, elog_size, elog_type) first then it reads the error log data in its buffer that moves the pending error log to processed list. After reading, the state machine moves to `ELOG_STATE_NONE` state.

It acknowledges the error log id after reading error log data by sending the call to OPAL, which in turn sends the acknowledgement mbox message to FSP and moves error log id from processed list to again back to free node list and this process goes on every FSP error log.

3.3.1 Design constraints:

```
#define ELOG_READ_MAX_RECORD            128
```

Currently, the number of error logs from FSP, OPAL can hold is limited to 128. If OPAL run out of free node in the list for the new error log, it sends 'Discarded by OPAL' message to the FSP. At some point in the future, it is upto FSP when it notifies again to OPAL about the discarded error log.

```
#define ELOG_WRITE_MAX_RECORD           64
```

There is also limitation on the number of OPAL error logs OPAL can hold is 64. If it is run out of the buffers in the pool, it will log the message saying 'Failed to get the buffer'.

3.3.2 Note

- For more information regarding error logging and PEL format refer to PAPR doc and P7 PEL and SRC PLDD document.
- Refer to `include/errorlog.h` for all the error logging interface parameters and `include/pel.h` for PEL structures.

3.3.3 Sample error logging

```
DEFINE_LOG_ENTRY(OPAL_RC_ATTN, OPAL_PLATFORM_ERR_EVT, OPAL_ATTN,
                 OPAL_PLATFORM_FIRMWARE, OPAL_PREDICTIVE_ERR_GENERAL,
                 OPAL_NA);

void report_error(int index)
{
       struct errorlog *buf;
       char data1[] = "This is a sample user defined data section1";
       char data2[] = "Error logging sample. These are dummy errors. Section 2";
       char data3[] = "Sample error Sample error Sample error Sample error \
                       Sample error abcdefghijklmnopqrstuvwxyz";
       int tag;

       printf("ELOG: In machine check report error index: %d\n", index);

       /* To report an error, create an error log with relevant information
        * opal_elog_create(). Call returns a pre-allocated buffer of type
        * 'struct errorlog' buffer with relevant fields updated.
        */

       /* tag -> unique ascii tag to identify a particular data dump section */
       tag = 0x4b4b4b4b;
       buf = opal_elog_create(&e_info(OPAL_RC_ATTN), tag);
       if (buf == NULL) {
               printf("ELOG: Error getting buffer.\n");
               return;
       }

       /* Append data or text with log_append_data() or log_append_msg() */
       log_append_data(buf, data1, sizeof(data1));

       /* In case of user wanting to add multiple sections of various dump data
```

```
         * for better debug, data sections can be added using this interface
         * void log_add_section(struct errorlog *buf, uint32_t tag);
         */
        tag = 0x4c4c4c4c;
        log_add_section(buf, tag);
        log_append_data(buf, data2, sizeof(data2));
        log_append_data(buf, data3, sizeof(data3));

        /* Once all info is updated, ready to be sent to FSP */
        printf("ELOG:commit to FSP\n");
        log_commit(buf);
}
```

3.3.4 Sample output PEL dump got from FSP

```
$ errl -d -x 0x533C9B37
|   00000000     50480030  01004154  20150728  02000500     PH.0..AT ..(....   |
|   00000010     20150728  02000566  4B000107  00000000      ..(...fK.......   |
|   00000020     00000000  00000000  B0000002  533C9B37     ............S..7   |
|   00000030     55480018  01004154  80002000  00000000     UH....AT.. .....   |
|   00000040     00002000  01005300  50530050  01004154     .. ...S.PS.P..AT   |
|   00000050     02000008  00000048  00000080  00000000     .......H........   |
|   00000060     00000000  00000000  00000000  00000000     ................   |
|   00000070     00000000  00000000  42423832  31343130     ........BB821410   |
|   00000080     20202020  20202020  20202020  20202020                        |
|   00000090     20202020  20202020  4548004C  01004154             EH.L..AT   |
|   000000A0     38323836  2D343241  31303738  34415400     8286-42A10784AT.   |
|   000000B0     00000000  00000000  00000000  00000000     ................   |
|   000000C0     00000000  00000000  00000000  00000000     ................   |
|   000000D0     00000000  00000000  20150728  02000500     ........ ..(....   |
|   000000E0     00000000  4D54001C  01004154  38323836     ....MT....AT8286   |
|   000000F0     2D343241  31303738  34415400  00000000     -42A10784AT.....   |
|   00000100     5544003C  01004154  4B4B4B4B  00340000     UD....ATKKKK.4..   |
|   00000110     54686973  20697320  61207361  6D706C65     This is a sample   |
|   00000120     20757365  72206465  66696E65  64206461      user defined da   |
|   00000130     74612073  65637469  6F6E3100  554400A7     ta section1.UD..   |
|   00000140     01004154  4C4C4C4C  009F0000  4572726F     ..ATLLLL....Erro   |
|   00000150     72206C6F  6767696E  67207361  6D706C65     r logging sample   |
|   00000160     2E205468  65736520  61726520  64756D6D     . These are dumm   |
|   00000170     79206572  726F7273  2E205365  6374696F     y errors. Sectio   |
|   00000180     6E203200  53616D70  6C652065  72726F72     n 2.Sample error   |
|   00000190     2053616D  706C6520  6572726F  72205361      Sample error Sa   |
|   000001A0     6D706C65  20657272  6F722053  616D706C     mple error Sampl   |
|   000001B0     65206572  726F7220  09090953  616D706C     e error ...Sampl   |
|   000001C0     65206572  726F7220  61626364  65666768     e error abcdefgh   |
|   000001D0     696A6B6C  6D6E6F70  71727374  75767778     ijklmnopqrstuvwx   |
|   000001E0     797A00                                     yz.                |
|-------------------------------------------------------------------------------|
|                      Platform Event Log - 0x533C9B37                          |
|-------------------------------------------------------------------------------|
|                               Private Header                                  |
|-------------------------------------------------------------------------------|
| Section Version          : 1                                                  |
| Sub-section type         : 0                                                  |
| Created by               : 4154                                               |
| Created at               : 07/28/2015 02:00:05                                |
```

```
| Committed at             : 07/28/2015 02:00:05                                |
| Creator Subsystem        : OPAL                                               |
| CSSVER                   :                                                    |
| Platform Log Id          : 0xB0000002                                         |
| Entry Id                 : 0x533C9B37                                         |
| Total Log Size           : 483                                                |
|-------------------------------------------------------------------------------|
|                                User Header                                    |
|-------------------------------------------------------------------------------|
| Section Version          : 1                                                  |
| Sub-section type         : 0                                                  |
| Log Committed by         : 4154                                               |
| Subsystem                : Platform Firmware                                  |
| Event Scope              : Unknown - 0x00000000                               |
| Event Severity           : Predictive Error                                   |
| Event Type               : Not Applicable                                     |
| Return Code              : 0x00000000                                         |
| Action Flags             : Report Externally                                  |
| Action Status            : Sent to Hypervisor                                 |
|-------------------------------------------------------------------------------|
|                      Primary System Reference Code                            |
|-------------------------------------------------------------------------------|
| Section Version          : 1                                                  |
| Sub-section type         : 0                                                  |
| Created by               : 4154                                               |
| SRC Format               : 0x80                                               |
| SRC Version              : 0x02                                               |
| Virtual Progress SRC     : False                                              |
| I5/OS Service Event Bit  : False                                              |
| Hypervisor Dump Initiated: False                                              |
| Power Control Net Fault  : False                                              |
|                                                                               |
| Valid Word Count         : 0x08                                               |
| Reference Code           : BB821410                                           |
| Hex Words 2 - 5          : 00000080 00000000 00000000 00000000                |
| Hex Words 6 - 9          : 00000000 00000000 00000000 00000000                |
|                                                                               |
|-------------------------------------------------------------------------------|
|                           Extended User Header                                |
|-------------------------------------------------------------------------------|
| Section Version          : 1                                                  |
| Sub-section type         : 0                                                  |
| Created by               : 4154                                               |
| Reporting Machine Type   : 8286-42A                                           |
| Reporting Serial Number  : 10784AT                                            |
| FW Released Ver          :                                                    |
| FW SubSys Version        :                                                    |
| Common Ref Time          : 07/28/2015 02:00:05                                |
| Symptom Id Len           : 0                                                  |
| Symptom Id               :                                                    |
|-------------------------------------------------------------------------------|
|                  Machine Type/Model & Serial Number                           |
|-------------------------------------------------------------------------------|
| Section Version          : 1                                                  |
| Sub-section type         : 0                                                  |
| Created by               : 4154                                               |
| Machine Type Model       : 8286-42A                                           |
| Serial Number            : 10784AT                                            |
```

```
|-------------------------------------------------------------------------------|
|                             User Defined Data                                 |
|-------------------------------------------------------------------------------|
| Section Version          : 1                                                  |
| Sub-section type         : 0                                                  |
| Created by               : 4154                                               |
|                                                                               |
|   00000000     4B4B4B4B  00340000  54686973  20697320     KKKK.4..This is    |
|   00000010     61207361  6D706C65  20757365  72206465     a sample user de   |
|   00000020     66696E65  64206461  74612073  65637469     fined data secti   |
|   00000030     6F6E3100                                    on1.               |
|                                                                               |
|-------------------------------------------------------------------------------|
|                             User Defined Data                                 |
|-------------------------------------------------------------------------------|
| Section Version          : 1                                                  |
| Sub-section type         : 0                                                  |
| Created by               : 4154                                               |
|                                                                               |
|   00000000     4C4C4C4C  009F0000  4572726F  72206C6F     LLLL....Error lo   |
|   00000010     6767696E  67207361  6D706C65  2E205468     gging sample. Th   |
|   00000020     65736520  61726520  64756D6D  79206572     ese are dummy er   |
|   00000030     726F7273  2E205365  6374696F  6E203200     rors. Section 2.   |
|   00000040     53616D70  6C652065  72726F72  2053616D     Sample error Sam   |
|   00000050     706C6520  6572726F  72205361  6D706C65     ple error Sample   |
|   00000060     20657272  6F722053  616D706C  65206572      error Sample er   |
|   00000070     726F7220  09090953  616D706C  65206572     ror ...Sample er   |
|   00000080     726F7220  61626364  65666768  696A6B6C     ror abcdefghijkl   |
|   00000090     6D6E6F70  71727374  75767778  797A00       mnopqrstuvwxyz.    |
|                                                                               |
|-------------------------------------------------------------------------------|
```

3.4 OPAL <–> BMC interactions

This document provides information about some of the user-visible interactions that skiboot performs with the BMC.

3.4.1 IPMI sensors

OPAL will interact with a few IPMI sensors during the boot process. These are:

- Boot Count [type 0xc3: OEM reserved]
- FW Boot progress [type 0x0f: System Firmware Progress]

Boot Count: assertion type. When OPAL reaches a late stage of boot, it sets the boot count sensor to 0x02. This is intended to allow the BMC detect a failed or aborted boot, for switching to a known-good firmware image.

FW Boot Progress: assertion type. During boot, skiboot will update this sensor to one of the IPMI-defined progress codes. The codes use by skiboot are:

- **PCI Resource configuration (0x01)**
  - asserted as the PCI devices have been probed and resources allocated
- **Motherboard init (0x14)**
  - asserted as the platform-specific components have been initialised

- **OS boot (0x13)**
  - asserted after skiboot has loaded the PAYLOAD image, and is about to boot it.

3.4.2 Chassis control messages

OPAL uses chassis control messages to instruct the BMC to remove power from the host. These messages are sent during graceful reboot and shutdown processes initiated by the host.

For a BMC-initiated graceful power-down (or reboot), the BMC is expected to send an OEM-defined SEL message, using a SMS_ATN to trigger a BMC-to-host notification. This SEL has a type of 0xc0, and command of 0x04. The data0 field of the SEL indicates shutdown (0x0) or reboot (0x1).

3.4.3 Watchdog support

OPAL supports a BMC watchdog during the boot process. This will be disabled before entering the OS.

3.4.4 Real-time clock

On platforms where a real-time-clock is not available, skiboot may use the IPMI SEL Time as a real-time-clock device.

3.5 GCOV for skiboot

3.5.1 Unit tests

All unit tests are built+run with gcov enabled.

```
make coverage-report
```

will generate a unit test coverage report like: http://open-power.github.io/skiboot/coverage-report/

3.5.2 Skiboot

You can now build Skiboot itself with gcov support, boot it on a machine, do things, and then extract out gcda files to generate coverage reports from real hardware (or a simulator).

3.5.3 Building Skiboot with GCOV

```
SKIBOOT_GCOV=1 make
```

You may need to `make clean` first.

This will build a skiboot lid roughly *twice* the size.

Flash/Install the skiboot.lid and boot.

3.5.4 Extracting GCOV data

The way we extract the gcov data from a system is by dumping the contents of skiboot memory and then parsing the data structures in user space with the extract-gcov utility in the skiboot repo.

mambo:

```
mysim memory fwrite 0x30000000 0x240000 skiboot.dump
```

FSP:

```
getmemproc 30000000 3407872 -fb skiboot.dump
```

linux (e.g. petitboot environment):

```
dd if=/proc/kcore skip=1572864 count=6656  of=skiboot.dump
```

You basically need to dump out the first 3MB of skiboot memory.

Then you need to find out where the gcov data structures are:

```
perl -e "printf '0x%x', 0x30000000 + 0x`grep gcov_info_list skiboot.map|cut -f 1 -d '
↪'`"
```

That address needs to be supplied to the extract-gcov utility:

```
./extract-gcov skiboot.dump 0x3023ec40
```

Once you've run extract-gcov, it will have extracted the gcda files from the skiboot memory image.

You can then run lcov:

```
lcov -b . -q -c -d . -o skiboot-boot.info
--gcov-tool
/opt/cross/gcc-4.8.0-nolibc/powerpc64-linux/bin/powerpc64-linux-gcov
```

IMPORTANT you should point lcov to the gcov for the compiler you used to build skiboot, otherwise you're likely to get errors.

3.6 Memory in skiboot

There are regions of memory we statically allocate for firmware as well as a HEAP region for boot and runtime allocations.

A design principle of skiboot is to attempt not to allocate memory at runtime, or at least keep it to a minimum, and not do so in any critical code path for the system to remain running.

At no point during runtime should a skiboot memory allocation failure cause the system to stop functioning.

3.6.1 HEAP

Dynamic memory allocations go in a single heap. This is identified as Region ibm,firmware-heap and appears as a reserved section in the device tree.

Originally, it was 12582912 bytes in size (declared in mem_map.h). Now, it is 13631488 bytes after being bumped as part of the GCOV work.

We increased heap size as on larger systems, we were getting close to using all the heap once skiboot became 2MB with GCOV.

Heap usage is printed before running the payload.

For example, as of writing, on a dual socket Tuleta:

```
[45215870591,5] SkiBoot skiboot-5.0.1-94-gb759ce2 starting...
[3680939340,5] CUPD: T side MI Keyword = SV830_027
[3680942658,5] CUPD: T side ML Keyword = FW830.00
[15404383291,5] Region ibm,firmware-heap free: 5378072
```

and on a palmetto:

```
[24748502575,5] SkiBoot skiboot-5.0.1-94-gb759ce2 starting...
[9870429550,5] Region ibm,firmware-heap free: 10814856
```

Our memory allocator is simple, a use pattern of:

```
A = malloc();
B = malloc();
free(A);
```

is likely to generate fragmentation, so it should generally be avoided where possible.

3.7 OPAL/Skiboot Nvlink Interface Documentation

3.7.1 Overview

NV-Link is a high speed interconnect that is used in conjunction with a PCI-E connection to create an interface between chips that provides very high data bandwidth. The PCI-E connection is used as the control path to initiate and report status of large data transfers. The data transfers themselves are sent over the NV-Link.

On IBM Power systems the NV-Link hardware is similar to our standard PCI hardware so to maximise code reuse the NV-Link is exposed as an emulated PCI device through system firmware (OPAL/skiboot). Thus each NV-Link capable device will appear as two devices on a system, the real PCI-E device and at least one emulated PCI device used for the NV-Link.

Presently the NV-Link is only capable of data transfers initiated by the target, thus the emulated PCI device will only handle registers for link initialisation, DMA transfers and error reporting (EEH).

3.7.2 Emulated PCI Devices

Each link will be exported as an emulated PCI device with a minimum of two emulated PCI devices per GPU. Emulated PCI devices are grouped per GPU.

The emulated PCI device will be exported as a standard PCI device by the Linux kernel. It has a standard PCI configuration space to expose necessary device parameters. The only functionality available is related to the setup of DMA windows.

Configuration Space Parameters

Vendor ID	0x1014	(IBM)
Device ID	0x04ea	
Revision ID	0x00	
Class	0x068000	(Bridge Device Other, ProgIf = 0x0)
BAR0/1	TL/DL Registers	

TL/DL Registers

Each link has 128KB of TL/DL registers. These will always be mapped to 64-bit BAR#0 of the emulated PCI device configuration space.

```
BAR#0 + 128K +-----------+
             | NTL (64K) |
BAR#0 + 64K  +-----------+
             | DL (64K)  |
BAR#0        +-----------+
```

Vendor Specific Capabilities

```
+----------------+----------------+----------------+----------------+
|  Version (0x02) |   Cap Length   |  Next Cap Ptr  |  Cap ID (0x09) |
+----------------+----------------+----------------+----------------+
|                       Procedure Status Register                    |
+--------------------------------------------------------------------+
|                       Procedure Control Register                   |
+--------------------------------------------------+-----------------+
|             Reserved             |   PCI Dev Flag  |   Link Number  |
+--------------------------------------------------+-----------------+
```

Version

> This refers to the version of the NPU config space. Used by device drivers to determine which fields of the config space they can expect to be available.

Procedure Control Register

> Used to start hardware procedures.
>
> Writes will start the corresponding procedure and set bit 31 in the procedure status register. This register must not be written while bit 31 is set in the status register. Performing a write while another procudure is already in progress will abort that procedure.
>
> Reads will return the in progress procedure or the last completed procedure number depending on the procedure status field.
>
> Procedure Numbers:
>
> 0. Abort in-progress procedure
> 1. NOP
> 2. Unsupported procedure
> 3. Unsupported procedure

4. Naples PHY - RESET
5. Naples PHY - TX_ZCAL
6. Naples PHY - RX_DCCAL
7. Naples PHY - TX_RXCAL_ENABLE
8. Naples PHY - TX_RXCAL_DISABLE
9. Naples PHY - RX_TRAINING
10. Naples NPU - RESET
11. Naples PHY - PHY preterminate
12. Naples PHY - PHY terminated

Procedure 5 (TX_ZCAL) should only be run once. System firmware will ensure this so device drivers may call this procedure mutiple times.

Procedure Status Register

The procedure status register is used to determine when execution of the procedure number in the control register is complete and if it completed successfully.

This register must be polled frequently to allow system firmware to execute the procedures.

Fields: Bit 31 - Procedure in progress Bit 30 - Procedure complete Bit 3-0 - Procedure completion code

Procedure completion codes: 0 - Procedure completed successfully. 1 - Transient failure. Procedure should be rerun. 2 - Permanent failure. Procedure will never complete successfully. 3 - Procedure aborted. 4 - Unsupported procedure.

PCI Device Flag

Bit 0 is set only if an actual PCI device was bound to this emulated device.

Link Number

Physical link number this emulated PCI device is associated with. One of 0, 1, 4 or 5 (links 2 & 3 do not exist on Naples).

Reserved

These fields must be ignored and no value should be assumed.

Interrupts

Each link has a single DL/TL interrupt assigned to it. These will be exposed as an LSI via the emulated PCI device. There are 4 links consuming 4 LSI interrupts. The 4 remaining interrupts supported by the corresponding PHB will be routed to OS platform for the purpose of error reporting.

3.7.3 Device Tree Bindings

See doc/device-tree/nvlink.txt

3.8 Stable Skiboot tree/releases

If you're at all familiar with the Linux kernel stable trees, this should seem fairly familiar.

The purpose of a -stable tree is to give vendors a stable base to create firmware releases from and to incorporate into service packs. New stable releases contain critical fixes only.

As a general rule, on the most recent skiboot release gets a maintained -stable tree. If you wish to maintain an older tree, speak up! For example, with my IBMer hat on, we'll maintain branches that we ship in products.

3.8.1 What patches are accepted?

- Patches must be obviously correct and tested
  - A Tested-by signoff is *important*
- A patch must fix a real bug
- No trivial patches, such fixups belong in main branch
- Not fix a purely theoretical problem unless you can prove how it's exploitable
- The patch, or an equivalent one, must already be in master
  - Submitting to both at the same time is okay, but backporting is better

3.8.2 HOWTO submit to stable

Two ways: 1. Send patch to the skiboot@ list with "[PATCH stable]" in subject

- This targets the patch *ONLY* to the stable branch.
  - Such commits will *NOT* be merged into master.
- Use this when:
  1. cherry-picking a fix from master
  2. fixing something that is only broken in stable
  3. fix in stable needs to be completely different than in master

  If b or c: explain why.
- If cherry-picking, include the following at the top of your commit message:

```
commit <sha1> upstream.
```

- If the patch has been modified, explain why in description.

2. Add "Cc: stable" above your Signed-off-by line when sending to skiboot@
   - This targets the patch to master and stable.
   - You can target a patch to a specific stable tree with:

     ```
     Cc: stable # 5.1.x
     ```

     and that will target it to the 5.1.x branch.
   - You can ask for prerequisites to be cherry-picked:

```
Cc: stable # 5.1.x 55ae15b Ensure we run pollers in cpu_wait_job()
Cc: stable # 5.1.x
```

Which means:

(a) please `git cherry-pick 55ae15b`

(b) then apply this patch to 5.1.x".

3.8.3 Trees

- https://github.com/open-power/skiboot/tree/stable (or via ssh at `git@github.com:open-power/skiboot.git`)
  - (branches are skiboot-X.Y.x - e.g. skiboot-5.1.x)
- Some stable versions may last longer than others
  - So there may be skiboot-5.1.x and skiboot-5.2.x actively maintained and skiboot-5.1.x could possibly outlast skiboot-5.2.x

3.9 Versioning Scheme of skiboot

3.9.1 History

For roughly the first six months of public life, skiboot just presented a git SHA1 as a version "number". This was "user visible" in two places:

1. `/sys/firmware/opal/msglog` the familiar `SkiBoot 71664fd-dirty starting...` message
2. device tree: `/proc/device-tree/ibm,opal/firmware/git-id`

Builds were also referred to by date and by corresponding PowerKVM release. Clearly, this was unlikely to be good practice going forward.

As of skiboot-4.0, this scheme has changed and we now present a version string instead. This better addresses the needs of everybody who is building OpenPower systems.

3.9.2 Current practice

The version string is constructed from a few places and is designed to be *highly* informative about what you're running. For the most part, it should be automatically constructed by the skiboot build system. The only times you need to do something is if you are a) making an upstream skiboot release or b) building firmware to release for your platform(s).

OPAL/skiboot has several consumers, for example:

- IBM shipping POWER8 systems with an FSP (FW810.XX and future)
- OpenPower
- OpenPower partners manufacturing OpenPower systems
- developers, test and support needing to understand what code a system is running

and there are going to be several concurrent maintained releases in the wild, likely build by different teams of people at different companies.

tl;dr; is you're likely going to see version numbers like this (for the hypothetical platforms 'ketchup' and 'mustard'):

- `skiboot-4.0-ketchup-0`
- `skiboot-4.0-ketchup-1`
- `skiboot-4.1-mustard-4`
- `skiboot-4.1-ketchup-0`

If you see *extra* things on the end of the version, then you're running a custom build from a developer (e.g. `skiboot-4.0-1-g23f147e-stewart-dirty-f42fc40` means something to us - explained below).

If you see less, for example `skiboot-4.0`, then you're running a build directly out of the main git tree. Those producing OPAL builds for users must *not* ship like this, even if the tree is identical.

Here are the components of the version string from master:

```
skiboot-4.0-1-g23f147e-debug-occ-stewart-dirty-f42fc40
^        ^^^ ^  ^^^^^^^ ^_______^    ^       ^   ^^^^^^^
|         |  |     |         |         |       |      |
|         |  |     |         |          \     /        - 'git diff|sha1sum'
|         |  |     |         |           \  /
|         |  |     |         |             - built from a dirty tree of $USER
|         |  |     |         |
|         |  |     |          - $EXTRA_VERSION (optional)
|         |  |     |
|         |  |      - git SHA1 of commit built
|         |  |
|         |   - commits head of skiboot-4.0 tag
|         |
|          - skiboot version number ---\
|                                      >--  from  the 'skiboot-4.0' git tag
 - product name (always skiboot)   ---/
```

When doing a release for a particular platform, you are expected to create and tag a branch from master. For the (hypothetical) ketchup platform which is going to do a release based on skiboot-4.0, you would create a tag 'skiboot-4.0-ketchup-0' pointing to the same revision as the 'skiboot-4.0' tag and then make any additional modifications to skiboot that were not in the 4.0 release. So, you could ship a skiboot with the following version string:

```
skiboot-4.0-ketchup-1
^        ^^^ ^       ^
|         |  |       |
|         |  |        - revision for this platform
|         |  |
|         |  |
|         |   - Platform name/version
|         |
|          - skiboot version number
|
 - product name (always skiboot)
```

This version string tells your users to expect what is in skiboot-4.0 plus some revisions for your platform.

3.9.3 Practical Considerations

You MUST correctly tag your git tree for sensible version numbers to be generated. Look at the (generated) version.c file to confirm you're building the correct version number. You will need annotated tags (git tag -a).

If your build infrastructure does *not* build skiboot from a git tree, you should specify SKIBOOT_VERSION as an environment variable (following this versioning scheme), otherwise the build will fail.

3.10 PCI

WARNING: This documentation **urgently needs updating** and is *woefully* incomplete.

3.10.1 IODA PE Setup Sequences

(**WARNING**: this was rescued from old internal documentation. Needs verification)

To setup basic PE mappings, the host performs this basic sequence:

For ibm,opal-ioda2, prior to allocating PHB resources to PEs, the host must allocate memory for PE structures and then calls `opal_pci_set_phb_table_memory( phb_id,rtt_addr,ivt_addr,ivt_len,rrba_addr,peltv_addr)` to define them to the PHB. OPAL returns `OPAL_UNSUPPORTED` status for `ibm,opal-ioda` PHBs.

The host calls `opal_pci_set_pe( phb_id,pe_number,bus,dev,func,validate_mask,bus_mask,dev_mask,fu mask)` to map a PE to a PCI RID or range of RIDs in the same PE domain.

The host calls `opal_pci_set_peltv(phb_id,parent_pe,child_pe,state)` to set a parent PELT vector bit for the child PE argument to 1 (a child of the parent) or 0 (not in the parent PE domain).

3.10.2 IODA MMIO Setup Sequences

(**WARNING**: this was rescued from old internal documentation. Needs verification)

The host calls `opal_pci_phb_mmio_enable( phb_id,window_type,window_num,0x0)` to disable the MMIO window.

The host calls `opal_pci_set_phb_mmio_window( phb_id,mmio_window,starting_real_address,starting_p` to change the MMIO window location in PCI and/or processor real address space, or to change the size – and corresponding window size – of a particular MMIO window.

The host calls `opal_pci_map_pe_mmio_window( pe_number,mmio_window,segment_number)` to map PEs to window segments, for each segment mapped to each PE.

The host calls `opal_pci_phb_mmio_enable( phb_id,window_type,window_num,0x1)` to enable the MMIO window.

3.10.3 IODA MSI Setup Sequences

(**WARNING**: this was rescued from old internal documentation. Needs verification)

To setup MSIs:

1. For ibm,opal-ioda PHBs, the host chooses an MVE for a PE to use and calls `opal_pci_set_mve( phb_id,mve_number,pe_number,)` to setup the MVE for the PE number. HAL treats this call as a NOP and returns hal_success status for ibm,opal-ioda2 PHBs.
2. The host chooses an XIVE to use with a PE and calls a. `opal_pci_set_xive_pe( phb_id,xive_number,pe_number)` to authorize that PE to signal that XIVE as an interrupt. The host must call this function for each XIVE assigned to a particular PE, but may use this call for all XIVEs prior to calling `opel_pci_set_mve()` to bind the PE XIVEs to an MVE. For MSI conventional, the host must bind a unique MVE for each sequential set of 32 XIVEs. b. The host forms the interrupt_source_number from the combination of the device tree MSI property base BUID and XIVE number, as an input to `opal_set_xive(interrupt_source_number,server_number,priority)` and `opal_get_xive(interrupt_source_number,server_number,priority)`

to set or return the server and priority numbers within an XIVE. c. `opal_get_msi_64[32](phb_id,mve_number,xive_num,msi_range,msi_address,message_data)` to determine the MSI DMA address (32 or 64 bit) and message data value for that xive.

For MSI conventional, the host uses this for each sequential power of 2 set of 1 to 32 MSIs, to determine the MSI DMA address and starting message data value for that MSI range. For MSI-X, the host calls this uniquely for each MSI interrupt with an msi_range input value of 1.

3. For `ibm,opal-ioda` PHBs, once the MVE and XIVRs are setup for a PE, the host calls `opal_pci_set_mve_enable( phb_id,mve_number,state)` to enable that MVE to be a valid target of MSI DMAs. The host may also call this function to disable an MVE when changing PE domains or states.

3.10.4 IODA DMA Setup Sequences

(**WARNING**: this was rescued from old internal documentation. Needs verification)

To Manage DMA Windows :

1. The host calls `opal_pci_map_pe_dma_window( phb_id,dma_window_number,pe_number,tce_levels,tce` `pci_start_addr )` to setup a DMA window for a PE to translate through a TCE table structure in KVM memory.
2. The host calls `opal_pci_map_pe_dma_window_real( phb_id,dma_window_number,pe_number,mem_low_a` to setup a DMA window for a PE that is translated (but validated by the PHB as an untranlsated address space authorized to this PE).

3.11 PCI Slots

The PCI slots are instantiated to represent their associated properties and operations. The slot properties are exported to OS through the device tree node of the corresponding parent PCI device. The slot operations are used to accomodate requests from OS regarding the indicated PCI slot:

- PCI slot reset
- PCI slot property retrival

The PCI slots are expected to be created by individual platforms based on the given templates, which are classified to PHB slot or normal one currently. The PHB slot is instantiated based on PHB types like P7IOC and PHB3. However, the normal PCI slots are created based on general RC (Root Complex), PCIE switch ports, PCIE-to-PCIx bridge. Individual platform may create PCI slot, which doesn't have existing template.

The PCI slots are created at different stages according to their types. PHB slots are expected to be created once the PHB is register (struct platform::pci_setup_phb()) because the PHB slot reset operations are required at early stage of PCI enumeration. The normal slots are populated after their parent PCI devices are instantiated at struct platform::pci_get_slot_info().

The operation set supplied by the template might be overrided and reimplemented, or partially. It's usually done according to the VPD figured out by individual platforms.

3.11.1 PCI Slot Operations

The following operations are supported to one particular PCI slot. More details could be found from the definition of struct pci_slot_ops:

Operation	Definition
get_presence_state	Check if any adapter connected to slot
get_link_state	Retrieve PCIE link status: up, down, link width
get_power_state	Retrieve the power status: on, off
get_attention_state	Retrieve attention status: on, off, blinking
get_latch_state	Retrieve latch status
set_power_state	Configure the power status: on, off
set_attention_state	Configure attention status: on, off, blinking
prepare_link_change	Prepare PCIE link status change
poll_link	Poll PCIE link until it's up or down permanently
creset	Complete reset, only available to PHB slot
freset	Fundamental reset
pfreset	Post fundamental reset
hreset	Hot reset
poll	Interface for OPAL API to drive internal state machine
add_properties	Additional PCI slot properties seen by platform

3.12 PCI Slot Properties

The following PCI slot properties have been exported through PCI device tree node for a root port, a PCIE switch port, or a PCIE to PCIx bridge. If the individual platforms (e.g. Firenze and Apollo) have VPD for the PCI slot, they should extract the PCI slot properties from VPD and export them accordingly.

Property	Definition
ibm,reset-by-firmware	Boolean indicating whether the slot reset should be done in firmware
ibm,slot-pluggable	Boolean indicating whether the slot is pluggable
ibm,slot-power-ctl	Boolean indicating whether the slot has power control
ibm,slot-wired-lanes	The number of hardware lanes that are wired
ibm,slot-pwr-led-ctl	Presence of slot power led, and controlling entity
ibm,slot-attn-led-ctl	Presence of slot ATTN led, and controlling entity

3.12.1 PCI Hotplug

The implementation of PCI slot hotplug heavily relies on its power state. Initially, the slot is powered off if there are no adapters behind it. Otherwise, the slot should be powered on.

In hot add scenario, the adapter is physically inserted to PCI slot. Then the PCI slot is powered on by OPAL API opal_pci_set_power_state(). The power is supplied to the PCI slot, the adapter behind the PCI slot is probed and the device sub-tree (for hot added devices) is populated. A OPAL message is sent to OS on completion. The OS needs retrieve the device sub-tree through OPAL API opal_get_device_tree(), unflatten it and populate the device sub-tree. After that, the adapter behind the PCI slot should be probed and added to the system.

On the other hand, the OS removes the adapter behind the PCI slot before calling opal_pci_set_power_state(). Skiboot cuts off the power supply to the PCI slot, removes the adapter behind the PCI slot and the corresponding device sub-tree. A OPAL message (OPAL_MSG_ASYNC_COMP) is sent to OS. The OS removes the device sub-tree for the adapter behind the PCI slot.

The OPAL message used in PCI hotplug is comprised of 4 dwords in sequence: asychronous token from OS, PCI slot device node's phandle, OPAL_PCI_SLOT_POWER_{ON, OFF}, OPAL_SUCCESS or errcode.

The states OPAL_PCI_SLOT_OFFLINE and OPAL_PCI_SLOT_ONLINE are used for removing or adding devices behind the slot. The device nodes in the device tree are removed or added accordingly, without actually changing the

slot's power state. The API call will return OPAL_SUCCESS immediately and no further asynchronous message will be sent.

3.12.2 PCI Slot on Apollo and Firenze

On IBM's Apollo and Firenze platform, the PCI VPD is fetched from dedicated LID, which is organized in so-called 1004, 1005, or 1006 format. 1006 mapping format isn't supported currently. The PCI slot properties are figured out from the VPD. On the other hand, there might have external power management entity hooked to I2C buses for one PCI slot. The fundamental reset operation of the PCI slot should be implemented based on the external power management entity for that case.

On Firenze platform, PERST pin is accessible through bit#10 of PCI config register (offset: 0x80) for those PCI slots behind some PLX switch downstream ports. For those PCI slots, PERST pin is utilized to implement fundamental reset if external power management entity doesn't exist.

For Apollo and Firenze platform, following PCI slot properties are exported through PCI device tree node except those generic properties (as above):

Property	Definition
ibm,slot-location-code	System location code string for the slot connector
ibm,slot-label	Slot label, part of "ibm,slot-location-code"

3.13 XSCOM Bindings

3.13.1 XSCOM regions

The top-level xscom nodes specify the mapping range from the 64-bit address space into the PCB address space.

There's one mapping range per chip xscom, therefore one node per mapping range.

```
/
/xscom@<chip-base-address-0>/
/xscom@<chip-base-address-1>/
...
/xscom@<chip-base-address-n>/
```

- where <chip-base-address-n> is the xscom base address with the gcid-specific bits (for chip n) OR-ed in.

Each xscom node has the following properties:

- #address-cells = 1
- #size-cells = 1
- reg = <base-address[#parent-address-cells] size[#parent-size-cells]>
- ibm,chip-id = gcid
- compatible = "ibm,xscom", "ibm,power8-scom" / "ibm,power7-xscom"

3.13.2 Chiplet endpoints

One sub-node per endpoint. Endpoints are defined by their (port, endpoint-address) data on the PCB, and are named according to their endpoint types:

```
/xscom@<chip-base-address>/
/xscom@<chip-base-address>/chiptod@<endpoint-addr>
/xscom@<chip-base-address>/lpc@<endpoint-addr>
```

- where the <endpoint-addr> is a single address (as distinct from the current (gcid,base) format), consisting of the SCOM port and SCOM endpoint bits in their 31-bit address format.

Each endpoint node has the following properties:

- reg = <endpoint-address[#parent-address-cells] size[#parent-size-cells]>
- compatible - depends on endpoint type, eg "ibm,power8-chiptod"

The endpoint address specifies the address on the PCB. So, to calculate the MMIO address for a PCB register:

```
mmio_addr  = <xscom-base-addr> | (pcb_addr[1:27] << 4)
                               | (pcb_addr[28:31] << 3)
```

Where:

- xscom-base-addr is the address from the first two cells of the parent node's reg property
- pcb_addr is the first cell of the endpoint's reg property

CHAPTER

FOUR

OPAL ABI

4.1 Device Tree

General notes on the Device Tree produced by skiboot. This chapter **needs updating**.

4.1.1 General comments

- skiboot does not require nodes to have phandle properties, but if you have them then *all* nodes must have them including the root of the device-tree (currently a HB bug !). It is recommended to have them since they are needed to represent the cache levels.
- **NOTE**: The example tree below only has phandle properties for nodes that are referenced by other nodes. This is *not* correct and is purely done for keeping this document smaller, make sure to follow the rule above.
- Only the "phandle" property is required. Sapphire also generates a "linux,phandle" for backward compatibility but doesn't require it as an input
- Any property not specifically documented must be put in "as is"
- All ibm,chip-id properties contain a HW chip ID which correspond on P8 to the PIR value shifted right by 7 bits, ie. it's a 6-bit value made of a 3-bit node number and a 3-bit chip number.
- Unit addresses (@xxxx part of node names) should if possible use lower case hexadecimal to be consistent with what skiboot does and to help some stupid parsers out there...

4.1.2 Reserve Map

Here are the reserve map entries. They should exactly match the reserved-ranges property of the root node (see documentation of that property):

```
/memreserve/  0x00000007fe600000 0x0000000000100000;
/memreserve/  0x00000007fe200000 0x0000000000100000;
/memreserve/  0x0000000031e00000 0x00000000003e0000;
/memreserve/  0x0000000031000000 0x0000000000e00000;
/memreserve/  0x0000000030400000 0x0000000000c00000;
/memreserve/  0x0000000030000000 0x0000000000400000;
/memreserve/  0x0000000400000000 0x0000000000600450;
```

4.1.3 Root Node

Root node of device tree:

```
/ {
        /*
         * "compatible" properties are string lists (ASCII strings separated by
         * \0 characters) indicating the overall compatibility from the more
         * specific to the least specific.
         *
         * The root node compatible property *must* contain "ibm,powernv" for
         * Linux to have the powernv platform match the machine. It is recommended
         * to add a slightly more precise property (first in order) indicating more
         * precisely the board type. We don't currently do that in HDAT based
         * setups but will.
         *
         * The standard naming is "vendor,name" so in your case, something like
         *
         * compatible = "goog,rhesus","ibm,powernv";
         *
         * would work. Or even better:
         *
         * compatible = "goog,rhesus-v1","goog,rhesus","ibm,powernv";
         */
        compatible = "ibm,powernv";

        /* mandatory */
        #address-cells = <0x2>;
        #size-cells = <0x2>;

        /* User visible board name (will be shown in /proc/cpuinfo) */
        model = "Machine Name";

        /*
         * The reserved-names and reserve-names properties work hand in hand. The␣
↪first one
         * is a list of strings providing a "name" for each entry in the second one␣
↪using
         * the traditional "vendor,name" format.
         *
         * The reserved-ranges property contains a list of ranges, each in the form of␣
↪2 cells
         * of address and 2 cells of size (64-bit x2 so each entry is 4 cells)␣
↪indicating
         * regions of memory that are reserved and must not be overwritten by skiboot␣
↪or
         * subsequently by the Linux Kernel.
         *
         * Corresponding entries must also be created in the "reserved map" part of␣
↪the flat
         * device-tree (which is a binary list in the header of the fdt).
         *
         * Unless a component (skiboot or Linux) specifically knows about a region␣
↪(usually
         * based on its name) and decides to change or remove it, all these regions are
         * passed as-is to Linux and to subsequent kernels across kexec and are kept
         * preserved.
         *
         * NOTE: Do *NOT* copy the entries below, they are just an example and are␣
↪actually
         * created by skiboot itself. They represent the SLW image as "detected" by␣
↪reading
```

```
        * the PBA BARs and skiboot own memory allocations.
        *
        * I would recommend that you put in there the SLW and OCC (or HOMER as one␣
↪block
        * if that's how you use it) and any additional memory you want to preserve␣
↪such
        * as FW log buffers etc...
        */

       reserved-names = "ibm,slw-image", "ibm,slw-image", "ibm,firmware-stacks",
↪"ibm,firmware-data", "ibm,firmware-heap", "ibm,firmware-code", "memory@400000000";
       reserved-ranges = <0x7 0xfe600000 0x0 0x100000 0x7 0xfe200000 0x0 0x100000 0x0␣
↪0x31e00000 0x0 0x3e0000 0x0 0x31000000 0x0 0xe00000 0x0 0x30400000 0x0 0xc00000 0x0␣
↪0x30000000 0x0 0x400000 0x4 0x0 0x0 0x600450>;

       /* Mandatory */
       cpus {
               #address-cells = <0x1>;
               #size-cells = <0x0>;

               /*
                * The following node must exist for each *core* in the system. The␣
↪unit
                * address (number after the @) is the hexadecimal HW CPU number (PIR␣
↪value)
                * of thread 0 of that core.
                */
               PowerPC,POWER8@20 {
                       /* mandatory/standard properties */
                       device_type = "cpu";
                       64-bit;
                       32-64-bridge;
                       graphics;
                       general-purpose;

                       /*
                        * The "status" property indicate whether the core is␣
↪functional. It's
                        * a string containing "okay" for a good core or "bad" for a␣
↪non-functional
                        * one. You can also just ommit the non-functional ones from␣
↪the DT
                        */
                       status = "okay";

                       /*
                        * This is the same value as the PIR of thread 0 of that core
                        * (ie same as the @xx part of the node name)
                        */
                       reg = <0x20>;

                       /* same as above */
                       ibm,pir = <0x20>;

                       /* chip ID of this core */
                       ibm,chip-id = <0x0>;

                       /*
```

```
                         * interrupt server numbers (aka HW processor numbers) of all␣
↪threads
                         * on that core. This should have 8 numbers and the first one␣
↪should
                         * have the same value as the above ibm,pir and reg properties
                         */
                        ibm,ppc-interrupt-server#s = <0x20 0x21 0x22 0x23 0x24 0x25␣
↪0x26 0x27>;

                        /*
                         * This is the "architected processor version" as defined in␣
↪PAPR. Just
                         * stick to 0x0f000004 for P8 and things will be fine
                         */
                        cpu-version = <0x0f000004>;

                        /*
                         * These are various definitions of the page sizes and segment␣
↪sizes
                         * supported by the MMU, those values are fine for P8 for now
                         */
                        ibm,processor-segment-sizes = <0x1c 0x28 0xffffffff 0xffffffff>
↪;
                        ibm,processor-page-sizes = <0xc 0x10 0x18 0x22>;
                        ibm,segment-page-sizes = <0xc 0x0 0x3 0xc 0x0 0x10 0x7 0x18␣
↪0x38 0x10 0x110 0x2 0x10 0x1 0x18 0x8 0x18 0x100 0x1 0x18 0x0 0x22 0x120 0x1 0x22␣
↪0x3>;

                        /*
                         * Similarly that might need to be reviewed later but will do␣
↪for now...
                         */
                        ibm,pa-features = [0x6 0x0 0xf6 0x3f 0xc7 0x0 0x80 0xc0];

                        /* SLB size, use as-is */
                        ibm,slb-size = <0x20>;

                        /* VSX support, use as-is */
                        ibm,vmx = <0x2>;

                        /* DFP support, use as-is */
                        ibm,dfp = <0x2>;

                        /* PURR/SPURR support, use as-is */
                        ibm,purr = <0x1>;
                        ibm,spurr = <0x1>;

                        /*
                         * Old-style core clock frequency. Only create this property␣
↪if the frequency fits
                         * in a 32-bit number. Do not create it if it doesn't
                         */
                        clock-frequency = <0xf5552d00>;

                        /*
                         * mandatory: 64-bit version of the core clock frequency,␣
↪always create this
                         * property.
```

```
                        */
                       ibm,extended-clock-frequency = <0x0 0xf5552d00>;

                       /* Timebase freq has a fixed value, always use that */
                       timebase-frequency = <0x1e848000>;

                       /* Same */
                       ibm,extended-timebase-frequency = <0x0 0x1e848000>;

                       /* Use as-is, values might need to be adjusted but that will␣
↪do for now */
                       reservation-granule-size = <0x80>;
                       d-tlb-size = <0x800>;
                       i-tlb-size = <0x0>;
                       tlb-size = <0x800>;
                       d-tlb-sets = <0x4>;
                       i-tlb-sets = <0x0>;
                       tlb-sets = <0x4>;
                       d-cache-block-size = <0x80>;
                       i-cache-block-size = <0x80>;
                       d-cache-size = <0x10000>;
                       i-cache-size = <0x8000>;
                       i-cache-sets = <0x4>;
                       d-cache-sets = <0x8>;
                       performance-monitor = <0x0 0x1>;

                       /*
                        * optional: phandle of the node representing the L2 cache for␣
↪this core,
                        * note: it can also be named "next-level-cache", Linux will␣
↪support both
                        * and Sapphire doesn't currently use those properties, just␣
↪passes them
                        * along to Linux
                        */
                       l2-cache = < 0x4 >;
               };

               /*
                * Cache nodes. Those are siblings of the processor nodes under /cpus␣
↪and
                * represent the various level of caches.
                *
                * The unit address (and reg property) is mostly free-for-all as long␣
↪as
                * there is no collisions. On HDAT machines we use the following␣
↪encoding
                * which I encourage you to also follow to limit surprises:
                *
                * L2   :  (0x20 << 24) | PIR (PIR is PIR value of thread 0 of core)
                * L3   :  (0x30 << 24) | PIR
                * L3.5 :  (0x35 << 24) | PIR
                *
                * In addition, each cache points to the next level cache via its
                * own "l2-cache" (or "next-level-cache") property, so the core node
                * points to the L2, the L2 points to the L3 etc...
                */
```

```
                l2-cache@20000020 {
                        phandle = <0x4>;
                        device_type = "cache";
                        reg = <0x20000020>;
                        status = "okay";
                        cache-unified;
                        d-cache-sets = <0x8>;
                        i-cache-sets = <0x8>;
                        d-cache-size = <0x80000>;
                        i-cache-size = <0x80000>;
                        l2-cache = <0x5>;
                };

                l3-cache@30000020 {
                        phandle = <0x5>;
                        device_type = "cache";
                        reg = <0x30000020>;
                        status = "bad";
                        cache-unified;
                        d-cache-sets = <0x8>;
                        i-cache-sets = <0x8>;
                        d-cache-size = <0x800000>;
                        i-cache-size = <0x800000>;
                };

        };

        /*
         * Interrupt presentation controller (ICP) nodes
         *
         * There is some flexibility as to how many of these are presents since
         * a given node can represent multiple ICPs. When generating from HDAT we
         * chose to create one per core
         */
        interrupt-controller@3ffff80020000 {
                /* Mandatory */
                compatible = "IBM,ppc-xicp", "IBM,power8-icp";
                interrupt-controller;
                #address-cells = <0x0>;
                device_type = "PowerPC-External-Interrupt-Presentation";

                /*
                 * Range of HW CPU IDs represented by that node. In this example
                 * the core starting at PIR 0x20 and 8 threads, which corresponds
                 * to the CPU node of the example above. The property in theory
                 * supports multiple ranges but Linux doesn't.
                 */
                ibm,interrupt-server-ranges = <0x20 0x8>;

                /*
                 * For each server in the above range, the physical address of the
                 * ICP register block and its size. Since the root node #address-cells
                 * and #size-cells properties are both "2", each entry is thus
                 * 2 cells address and 2 cells size (64-bit each).
                 */
                reg = <0x3ffff 0x80020000 0x0 0x1000 0x3ffff 0x80021000 0x0 0x1000␣
↪0x3ffff 0x80022000 0x0 0x1000 0x3ffff 0x80023000 0x0 0x1000 0x3ffff 0x80024000 0x0␣
↪0x1000 0x3ffff 0x80025000 0x0 0x1000 0x3ffff 0x80026000 0x0 0x1000 0x3ffff␣
↪0x80027000 0x0 0x1000>;
```

```
};

/*
 * The "memory" nodes represent physical memory in the system. They
 * do not represent DIMMs, memory controllers or Centaurs, thus will
 * be expressed separately.
 *
 * In order to be able to handle affinity properly, we require that
 * a memory node is created for each range of memory that has a different
 * "affinity", which in practice means for each chip since we don't
 * support memory interleaved across multiple chips on P8.
 *
 * Additionally, it is *not* required that one chip = one memory node,
 * it is perfectly acceptable to break down the memory of one chip into
 * multiple memory nodes (typically skiboot does that if the two MCs
 * are not interlaved).
 */
memory@0 {
        device_type = "memory";

        /*
         * We support multiple entries in the ibm,chip-id property for
         * memory nodes in case the memory is interleaved across multiple
         * chips but that shouldn't happen on P8
         */
        ibm,chip-id = <0x0>;

        /* The "reg" property is 4 cells, as usual for a child of
         * the root node, 2 cells of address and 2 cells of size
         */
        reg = <0x0 0x0 0x4 0x0>;
};

/*
 * The XSCOM node. This is the closest thing to a "chip" node we have.
 * there must be one per chip in the system (thus a DCM has two) and
 * while it represents the "parent" of various devices on the PIB/PCB
 * that we want to expose, it is also used to store all sort of
 * miscellaneous per-chip information on HDAT based systems (such
 * as VPDs).
 */
xscom@3fc0000000000 {
        /* standard & mandatory */
        #address-cells = <0x1>;
        #size-cells = <0x1>;
        scom-controller;
        compatible = "ibm,xscom", "ibm,power8-xscom";

        /* The chip ID as usual ... */
        ibm,chip-id = <0x0>;

        /* The base address of xscom for that chip */
        reg = <0x3fc00 0x0 0x8 0x0>;

        /*
         * This comes from HDAT and I *think* is the raw content of the
         * module VPD eeprom (and thus doesn't have a standard ASCII keyword
         * VPD format). We don't currently use it though ...
```

```
                 */
                ibm,module-vpd = < ... big pile of binary data ... >;

                /* PSI host bridge XSCOM register set */
                psihb@2010900 {
                        reg = <0x2010900 0x20>;
                        compatible = "ibm,power8-psihb-x", "ibm,psihb-x";
                };

                /* Chip TOD XSCOM register set */
                chiptod@40000 {
                        reg = <0x40000 0x34>;
                        compatible = "ibm,power-chiptod", "ibm,power8-chiptod";

                        /*
                         * Create that property with no value if this chip has
                         * the Primary TOD in the topology. If it has the secondary
                         * one (backup master ?) use "secondary".
                         */
                        primary;
                };

                /* NX XSCOM register set */
                nx@2010000 {
                        reg = <0x2010000 0x4000>;
                        compatible = "ibm,power-nx", "ibm,power8-nx";
                };

                /*
                 * PCI "PE Master" XSCOM register set for each active PHB
                 *
                 * For now, do *not* create these if the PHB isn't connected,
                 * clocked, or the PHY/HSS not configured.
                 */
                pbcq@2012000 {
                        reg = <0x2012000 0x20 0x9012000 0x5 0x9013c00 0x15>;
                        compatible = "ibm,power8-pbcq";

                        /* Indicate the PHB index on the chip, ie, 0,1 or 2 */
                        ibm,phb-index = <0x0>;

                        /* Create that property to use the IBM-style "A/B" dual input
                         * slot presence detect mechanism.
                         */
                        ibm,use-ab-detect;

                        /*
                         * TBD: Lane equalization values. Not currently used by
                         * skiboot but will have to be sorted out
                         */
                        ibm,lane_eq = <0x0>;
                };

                pbcq@2012400 {
                        reg = <0x2012400 0x20 0x9012400 0x5 0x9013c40 0x15>;
                        compatible = "ibm,power8-pbcq";
                        ibm,phb-index = <0x1>;
                        ibm,use-ab-detect;
```

```
                ibm,lane_eq = <0x0>;
        };

        /*
         * Here's the LPC bus. Ideally each chip has one but in
         * practice it's ok to only populate the ones actually
         * used for something. This is not an exact representation
         * of HW, in that case we would have eccb -> opb -> lpc,
         * but instead we just have an lpc node and the address is
         * the base of the ECCB register set for it
         *
         * Devices on the LPC are represented as children nodes,
         * see example below for a standard UART.
         */
        lpc@b0020 {
                /*
                 * Empty property indicating this is the primary
                 * LPC bus. It will be used for the default UART
                 * if any and this is the bus that will be used
                 * by Linux as the virtual 64k of IO ports
                 */
                primary;

                /*
                 * 2 cells of address, the first one indicates the
                 * address type, see below
                 */
                #address-cells = <0x2>;
                #size-cells = <0x1>;
                reg = <0xb0020 0x4>;
                compatible = "ibm,power8-lpc";

                /*
                 * Example device: a UART on IO ports.
                 *
                 * LPC address have 2 cells. The first cell is the
                 * address type as follow:
                 *
                 *   0 : LPC memory space
                 *   1 : LPC IO space
                 *   2:  LPC FW space
                 *
                 * (This corresponds to the OPAL_LPC_* arguments
                 * passed to the opal_lpc_read/write functions)
                 *
                 * The unit address follows the old ISA convention
                 * for open firmware which prefixes IO ports with "i".
                 *
                 * (This is not critical and can be 1,3f8 if that's
                 * problematic to generate)
                 */
                serial@i3f8 {
                        reg = <0x1 0x3f8 8>;
                        compatible = "ns16550", "pnpPNP,501";

                        /* Baud rate generator base frequency */
                        clock-frequency = < 1843200 >;
```

```
                                /* Default speed to use */
                                current-speed = < 115200 >;

                                /* Historical, helps Linux */
                                device_type = "serial";

                                /*
                                 * Indicate which chip ID the interrupt
                                 * is routed to (we assume it will always
                                 * be the "host error interrupt" (aka
                                 * "TPM interrupt" of that chip).
                                 */
                                ibm,irq-chip-id = <0x0>;
                        }
                };
        };
};
```

4.2 Device Tree

Device Tree for OPAL. Please refer to Device Tree Spec.

4.2.1 ibm,opal

ibm,opal/diagnostics device tree entries

The diagnostics node under ibm,opal describes a userspace-to-firmware interface, supporting the runtime processor recovery diagnostics functions.

The properties of a prd node are:

```
compatible = "ibm,opal-prd"
```

System Firmware

The 'firmware' node under 'ibm,opal' lists system and OPAL firmware version.

```
firmware {
      symbol-map = <0x0 0x300ac650 0x0 0x1b3f5>;
      compatible = "ibm,opal-firmware";
      ml-version = [4d 4c 20 46 57 37 37 30 2e 32 30 20 46 57 37 37 30 2e 32 30 20 46␣
↪57 37 37 30 2e 32 30];
      mi-version = <0x4d49205a 0x4c373730 0x5f303735 0x205a4c37 0x37305f30 0x3735205a␣
↪0x4c373730 0x5f303735>;
      version = "skiboot-5.0-rc2";
      phandle = <0x8e>;
      linux,phandle = <0x8e>;
};
```

compatible property describes OPAL compatibility.

symbol-map property describes OPAL symbol start address and size.

version property describes OPAL version. Replaces 'git-id', so may not be present. On POWER9 and above, it is always present.

mi-version property describes Microcode Image. Only on IBM FSP systems. Will (likely) not be present on POWER9 systems.

ml-version property describes Microcode Level. Only on IBM FSP systems. Will (likely) not be present on POWER9 systems.

MI/ML format

```
<ML/MI> <T side version> <P side version> <boot side version>
```

ibm,opal/flash device tree entries

The flash@<n> nodes under ibm,opal describe flash devices that can be accessed through the OPAL_FLASH_{READ,ERASE,WRITE} interface.

These interfaces take an 'id' parameter, which corresponds to the ibm,opal-id property of the node.

The properties under a flash node are:

- `compatible = "ibm,opal-flash"`

ibm,opal-id = <id> provides the index used for the OPAL_FLASH_XXX calls to reference this flash device

reg = <0 size> the offset and size of the flash device

ibm,flash-block-size the read/write/erase block size for the flash interface. Calls to read/write/erase must be aligned to the block size.

#address-cells = <1>, #size-cells = <1> flash devices are currently 32-bit addressable

If valid partitions are found on the flash device, then `partition@<offset>` sub-nodes are added to the flash node. These match the Linux binding for flash partitions; the reg parameter contains the offset and size of the partition.

Service Indicators (LEDS)

The 'leds' node under 'ibm,opal' lists service indicators available in the system and their capabilities.

```
leds {
      compatible = "ibm,opal-v3-led";
      phandle = <0x1000006b>;
      linux,phandle = <0x1000006b>;
      led-mode = "lightpath";

      U78C9.001.RST0027-P1-C1 {
              led-types = "identify", "fault";
              phandle = <0x1000006f>;
              linux,phandle = <0x1000006f>;
      };
      ...
      ...
};
```

compatible property describes LEDs compatibility.

led-mode property describes service indicator mode (lightpath/guidinglight).

Each node under 'leds' node describes location code of FRU/Enclosure.

The properties under each node:

led-types Supported indicators (attention/identify/fault).

These LEDs can be accessed through OPAL_LEDS_{GET/SET}_INDICATOR interfaces. Refer to doc/opal-api/opal-led-get-set-114-115.txt for interface details.

Operator Panel (oppanel)

```
oppanel {
      compatible = "ibm,opal-oppanel";
      #lines = <0x2>;
      #length = <0x10>;
};
```

The Operator Panel is a device for displaying small amounts of textual data to an administrator. On IBM POWER8 systems with an FSP, this is a small 16x2 LCD panel that can be viewed either from the Web UI of the FSP (known as ASM) or by physically going to the machine and looking at the panel.

The operator panel does not have to be present.

If it is, there are OPAL calls to read and write to it.

The device tree entry is so that the host OS knows the size of the panel and can pass buffers of the appropriate size to the OPAL calls.

ibm,opal/power-mgt device tree entries

All available CPU idle states are listed in ibm,cpu-idle-state-names

For example:

```
ibm,cpu-idle-state-names = "nap", "fastsleep_", "winkle";
```

The idle states are characterized by latency and residency numbers which determine the breakeven point for entry into them. The latency is a measure of the exit overhead from the idle state and residency is the minimum amount of time that a CPU must be predicted to be idle so as to reap the powersavings from entering into that idle state.

These numbers are made use of by the cpuidle governors in the kernel to arrive at the appropriate idle state that a CPU must enter into when there is no work to be done. The values in ibm,cpu-idle-state-latencies-ns are the the measured latency numbers for the idle states. The residency numbers have been arrived at experimentally after ensuring that the performance of latency sensitive workloads do not regress while allowing deeper idle states to be entered into during low load situations. The kernel is expected to use these values for optimal power efficiency.

```
ibm,cpu-idle-state-residency-ns = <0x1 0x2 0x3>
ibm,cpu-idle-state-latencies-ns = <0x1 0x2 0x3>
```

ibm,cpu-idle-state-pmicr ibm,cpu-idle-state-pmicr-mask

In POWER8, idle states sleep and winkle have 2 modes- fast and deep. In fast mode, idle state puts the core into threshold voltage whereas deep mode completely turns off the core. Choosing fast vs deep mode for an idle state can be done either via PM_GP1 scom or by writing to PMICR special register. If using the PMICR path to choose fast/deep mode then ibm,cpu-idle-state-pmicr and ibm,cpu-idle-state-pmicr-mask properties expose relevant PMICR bits and values for corresponding idle states.

ibm,cpu-idle-state-psscr ibm,cpu-idle-state-psscr-mask

In POWER ISA v3, there is a common instruction 'stop' to enter any idle state and SPR PSSCR is used to specify which idle state needs to be entered upon executing stop instruction. Properties ibm,cpu-idle-state-psscr and ibm,cpu-idle-state-psscr-mask expose the relevant PSSCR bits and values for corresponding idle states.

ibm,cpu-idle-state-flags

These flags are used to describe the characteristics of the idle states like the kind of core state loss caused. These flags are used by the kernel to save/restore appropriate context while using the idle states.

ibm,pstate-ids

This property lists the available pstate identifiers, as signed 32-bit big-endian values. While the identifiers are somewhat arbitrary, these define the order of the pstates in other ibm,pstate-* properties.

ibm,pstate-frequencies-mhz

This property lists the frequency, in MHz, of each of the pstates listed in the ibm,pstate-ids file. Each frequency is a 32-bit big-endian word.

ibm,pstate-max ibm,pstate-min ibm,pstate-nominal

These properties give the maximum, minimum and nominal pstate values, as an id specified in the ibm,pstate-ids file.

ibm,pstate-vcss ibm,pstate-vdds

These properties list a voltage-identifier of each of the pstates listed in ibm,pstate-ids for the Vcs and Vdd values used for that pstate. Each VID is a single byte.

ibm,pstate-ultra-turbo ibm,pstate-turbo

These properties are added when ultra-turbo(WOF) is enabled. These properties give the max turbo and max ultra-turbo pstate.

ibm,pstate-core-max

This property is added when ultra_turbo(WOF) is enabled. This property gives the list of max pstate for each 'n' number of active cores in the chip.

ibm,opal/sensors/ device tree nodes

All sensors of a POWER8 system are made available to the OS in the ibm,opal/sensors/ directory. Each sensor is identified with a node which name follows this pattern :

```
<resource class name>@<resource identifier>/
```

For example :

```
core-temp@20/
```

Each node has a minimum set of properties describing the sensor :

- a “compatible” property which should be “ibm,opal-sensor”
- a “sensor-type” property, which can be “temp”, “fan”, “power”. More will be added when new resources are supported. This type is used “as is” by the Linux driver to map sensors in the sysfs interface of the hwmon framework of Linux.
- a “sensor-data” property giving a unique handler for the OPAL_SENSOR_READ call to be used by Linux to get the value of a sensor attribute. A sensor handler has the following encoding :

```
|  Attr.  |  Res.   |   Resource     |
| Number  | Class   |      Id        |
|---------|---------|----------------|
```

- a “sensor-status” property giving the state of the sensor. The status bits have the slightly meanings depending on the resource type but testing against 0x6 should raise an alarm.
- an optional “label” property

Each node can have some extra properties depending on the resource they represent. See the tree below for more information.

```
ibm,opal/sensors/ {

      /*
       * Core temperatures (DTS) nodes.
       *
       * We use the PIR of the core as a resource identifier.
       */
      core-temp@20 {
              compatible = "ibm,opal-sensor";
              name = "core-temp";
              sensor-type = "temp";

              /* Status bits :
               *
               * 0x0003       FATAL
               * 0x0002       CRITICAL
               * 0x0001       WARNING
               */
              sensor-data = <0x00800020>;

              /*
               * These are extra properties to help Linux output.
               */
              ibm,pir = <0x20>;
              label = "Core";
      };

      /*
       * Centaur temperatures (DTS) nodes. Open Power only.
       *
```

```
        * We use the PIR of the core as a resource identifier.
        */
       mem-temp@1 {
               compatible = "ibm,opal-sensor";
               name = "mem-temp";
               sensor-type = "temp";

               /* Status bits :
                *
                * 0x0003       FATAL
                * 0x0002       CRITICAL
                * 0x0001       WARNING
                */
               sensor-data = <0x00810001>;

               /*
                * These are extra properties to help Linux output.
                */
               ibm,chip-id = <0x80000001>;
               label = "Centaur";
       };

};
```

```
ibm,opal {
              #address-cells = <0x0>;
              #size-cells = <0x0>;
              compatible = "ibm,opal-v2", "ibm,opal-v3";

; v2 is maintained for possible compatibility with very, very old kernels
; it will go away at some point in the future. Detect and rely on ibm,opal-v3
; ibm,opal-v2 is *NOT* present on POWER9 and above.

              ibm,associativity-reference-points = <0x4 0x3>;
              ibm,heartbeat-ms = <0x7d0>;

; how often any OPAL call needs to be made to avoid a watchdog timer on BMC
; from kicking in

              ibm,opal-memcons = <0x0 0x3007a000>;

; location of in memory OPAL console buffer.

              ibm,opal-trace-mask = <0x0 0x3008c3f0>;
              ibm,opal-traces = <0x0 0x3007b010 0x0 0x10077 0x0 0x3b001010 0x0␣
↪0x1000a7 0x0 0x3b103010 0x0 0x1000a7 0x0 0x3b205010 0x0 0x1000a7 0x0 0x3b307010 0x0␣
↪0x1000a7 0x0 0x3b409010 0x0 0x1000a7 0x10 0x1801010 0x0 0x1000a7 0x10 0x1903010 0x0␣
↪0x1000a7 0x10 0x1a05010 0x0 0x1000a7 0x10 0x1b07010 0x0 0x1000a7 0x10 0x1c09010 0x0␣
↪0x1000a7 0x10 0x1d0b010 0x0 0x1000a7 0x10 0x1e0d010 0x0 0x1000a7 0x10 0x1f0f010 0x0␣
↪0x1000a7 0x10 0x2011010 0x0 0x1000a7 0x10 0x2113010 0x0 0x1000a7 0x10 0x2215010 0x0␣
↪0x1000a7 0x10 0x2317010 0x0 0x1000a7 0x10 0x2419010 0x0 0x1000a7 0x10 0x251b010 0x0␣
↪0x1000a7 0x10 0x261d010 0x0 0x1000a7>;

; see docs on tracing

              linux,phandle = <0x10000003>;
              opal-base-address = <0x0 0x30000000>;
              opal-entry-address = <0x0 0x300050c0>;
```

```
            opal-interrupts = <0x10 0x11 0x12 0x13 0x14 0x20010 0x20011 0x20012␣
↪0x20013 0x20014 0xffe 0xfff 0x17fe 0x17ff 0x2ffe 0x2fff 0x37fe 0x37ff 0x20ffe␣
↪0x20fff 0x22ffe 0x22fff 0x237fe 0x237ff>;
            opal-msg-async-num = <0x8>;
            opal-msg-size = <0x48>;
            opal-runtime-size = <0x0 0x9a00000>;
            phandle = <0x10000003>;
```

4.2.2 Nvlink Device Tree Bindings

See doc/nvlink.txt for general Nvlink information.

NPU bindings:

```
xscom@3fc0000000000 {
      npu@8013c00 {
                        reg = <0x8013c00 0x2c>;
                        compatible = "ibm,power8-npu";
                        ibm,npu-index = <0x0>;
                        ibm,npu-links = <0x4>;

; Number of links wired up to this npu.

                        phandle = <0x100002bc>;
                        linux,phandle = <0x100002bc>;

                        link@0 {
                                ibm,npu-pbcq = <0x1000000b>;

; phandle to the pbcq which connects to the GPU.

                                ibm,npu-phy = <0x80000000 0x8010c3f>;

; SCOM address of the IBM PHY controlling this link.

                                compatible = "ibm,npu-link";
                                ibm,npu-lane-mask = <0xff>;

; Mask specifying which IBM PHY lanes are used for this link.

                                phandle = <0x100002bd>;
                                ibm,npu-link-index = <0x0>;

; Hardware link index. Naples systems contain links at index 0,1,4 & 5.
; Used to calculate various address offsets.

                                linux,phandle = <0x100002bd>;
                        };

                        link@1 {
                                ibm,npu-pbcq = <0x1000000b>;
                                ibm,npu-phy = <0x80000000 0x8010c3f>;
                                compatible = "ibm,npu-link";
                                ibm,npu-lane-mask = <0xff00>;
                                phandle = <0x100002be>;
                                ibm,npu-link-index = <0x1>;
                                linux,phandle = <0x100002be>;
```

```
                        };

                        link@4 {
                                ibm,npu-pbcq = <0x1000000a>;
                                ibm,npu-phy = <0x80000000 0x8010c7f>;
                                compatible = "ibm,npu-link";
                                ibm,npu-lane-mask = <0xff00>;
                                phandle = <0x100002bf>;
                                ibm,npu-link-index = <0x4>;
                                linux,phandle = <0x100002bf>;
                        };

                        link@5 {
                                ibm,npu-pbcq = <0x1000000a>;
                                ibm,npu-phy = <0x80000000 0x8010c7f>;
                                compatible = "ibm,npu-link";
                                ibm,npu-lane-mask = <0xff>;
                                phandle = <0x100002c0>;
                                ibm,npu-link-index = <0x5>;
                                linux,phandle = <0x100002c0>;
                        };
        };
};
```

Emulated PCI device bindings

```
pciex@3fff000400000 {
         ibm,npcq = <0x100002bc>;

; phandle to the NPU node. Used to find associated PCI GPU devices.

         compatible = "ibm,power8-npu-pciex", "ibm,ioda2-npu-phb";

         pci@0 {
                 reg = <0x0 0x0 0x0 0x0 0x0>;
                 revision-id = <0x0>;
                 interrupts = <0x1>;
                 device-id = <0x4ea>;
                 ibm,pci-config-space-type = <0x1>;
                 vendor-id = <0x1014>;
                 ibm,gpu = <0x100002f7>;

; phandle pointing the associated GPU PCI device node

                 phandle = <0x100002fc>;
         };

         pci@1 {
                 reg = <0x800 0x0 0x0 0x0 0x0>;
                 revision-id = <0x0>;
                 interrupts = <0x1>;
                 device-id = <0x4ea>;
                 ibm,pci-config-space-type = <0x1>;
                 vendor-id = <0x1014>;
                 ibm,gpu = <0x100002f5>;
                 phandle = <0x100002fe>;
```

```
                class-code = <0x60400>;
                linux,phandle = <0x100002fe>;
        };

        pci@0,1 {
                reg = <0x100 0x0 0x0 0x0 0x0>;
                revision-id = <0x0>;
                interrupts = <0x2>;
                device-id = <0x4ea>;
                ibm,pci-config-space-type = <0x1>;
                vendor-id = <0x1014>;
                ibm,gpu = <0x100002f7>;
                phandle = <0x100002fd>;
                class-code = <0x60400>;
                linux,phandle = <0x100002fd>;
        };

        pci@1,1 {
               reg = <0x900 0x0 0x0 0x0 0x0>;
                revision-id = <0x0>;
                interrupts = <0x2>;
                device-id = <0x4ea>;
                ibm,pci-config-space-type = <0x1>;
                vendor-id = <0x1014>;
                ibm,gpu = <0x100002f5>;
                phandle = <0x100002ff>;
                class-code = <0x60400>;
                linux,phandle = <0x100002ff>;
        };
};
```

4.2.3 Nest (NX) Accelerator Coprocessor

The NX coprocessor is present in P7+ or later processors. Each NX node represents a unique NX coprocessor. The nodes are located under an xscom node, as:

```
/xscom@<xscom_addr>/nx@<nx_addr>
```

With unique xscom and nx addresses. Their compatible node contains "ibm,power-nx".

NX 842 Coprocessor

This is the memory compression coprocessor, which uses the IBM proprietary 842 compression algorithm and format. Each nx node contains an 842 engine.

```
ibm,842-coprocessor-type     : CT value common to all 842 coprocessors
ibm,842-coprocessor-instance : CI value unique to all 842 coprocessors
```

Access to the coprocessor requires using the ICSWX instruction, which uses a specific format including a Coprocessor Type (CT) and Coprocessor Instance (CI) value to address each request to the right coprocessor. The driver should use the CT and CI values for a particular node to communicate with it. For all 842 coprocessors in the system, the CT value will (should) be the same, while each will have a different CI value. The driver can use CI 0 to allow the hardware to automatically select which coprocessor instance to use.

NX RNG Coprocessor

This is the Random Number Generator (RNG) coprocessor, which is a part of each NX coprocessor. Each node represents a unique RNG coprocessor. Its nodes are not under the main nx node, they are located at:

```
/hwrng@<addr>          : RNG at address <addr>
ibm,chip-id            : chip id where the RNG is
reg                    : address of the register to read from
```

Each read from the RNG register will provide a new random number.

4.2.4 reserved-memory device tree nodes

OPAL exposes reserved memory through a top-level reserved-memory node, containing subnodes that represent each reserved memory region.

This follows the Linux specification for the /reserved-memory node, described in the kernel source tree, in:

> Documentation/devicetree/bindings/reserved-memory/reserved-memory.txt

The top-level /reserved-memory node contains:

```
#size-cells = <2>
#address-cells = <2>
```

Addresses and sizes are all 64-bits.

ranges the empty ranges node indicates no translation of physical addresses in the subnodes.

The sub-nodes under the /reserved-memory node contain:

reg = <address size> the address and size of the reserved memory region. The address and size values are two cells each, as signified by the top-level `#{address,size}-cells`

ibm,prd-label = "string" a string token for use by the prd system. Specific ranges may be used by prd - those will be referenced by this label.

4.2.5 VPD (Vital Product Data)

VPD provides the information about the FRUs (Field Replaceable Unit) present in the system and each vpd node in the device tree represents a FRU. These node and their properties are specific to the FSP-based systems, passed to the skiboot in the form of FSP-defined HDAT structures. skiboot parses these structures and add respective nodes in the device tree.

```
/vpd                    : VPD root node
<fru-name>@<rsrc-id>    : Node name
ibm,vpd                         : VPD data binary blob
ccin                    : Customer Card Identification Number
fru-type                : FRU type label (2 bytes ASCII character)
fru-number              : FRU stocking part number
ibm,loc-code            : Location code
part-number             : Part number
serial-number           : Serial number
ibm,chip-id             : Processor Id
size                    : DIMM size (applicable for DIMM VPD only)
ibm,memory-bus-frequency: DIMM frequency (applicable for DIMM VPD only)
```

The VPD tree in the device tree depicts the hierarchial structure of the FRUs having parent-child relationship.

```
root-node-vpd@a000
   |-- enclosure@1e00
   |    |-- air-mover@3a00
   |    |-- air-mover@3a01
   |    |-- backplane@800
   |    |    |-- anchor-card@500
   |    |    |-- backplane-extender@900
   |    |    |    |-- serial-connector@2a00
   |    |    |    |-- usb-connector@2900
   |    |    |    `-- usb-connector@2901
   |    |    |-- ethernet-connector@2800
   |    |    |-- ethernet-connector@2801
   |    |    |-- ms-dimm@d002
   |    |    |-- ms-dimm@d003
   |    |    |-- processor@1000
   |    |    |-- processor@1001
   |    |    |-- usb-connector@2902
   |    |    |-- usb-connector@2903
   |    |    |-- usb-connector@2904
   |    |    `-- usb-connector@2905
   |    |-- dasd-backplane@2400
   |    |-- dasd-backplane@2401
   |    |-- power-supply@3103
   |    `-- service-processor@200
   |-- enclosure-fault-led@a300
   |-- enclosure-led@a200
   |-- root-node-vpd@a001
   `-- system-vpd@1c00
```

Example vpd node:

```
anchor-card@500 {
       ccin = "52FE";
       fru-number = "00E2147";
       description = "System Anchor Card - IBM Power 824";
       ibm,loc-code = "U78C9.001.WZS007X-P1-C13";
       serial-number = "YL10113BJ001";
       ibm,vpd = <0x84cc0052 0x54045649 0x4e494452 0x10414e43 0x484f5220 0x20202020
↪0x20202020 0x20434501 0x31565a02 0x3031464e 0x7303045 0x32313437 0x504e0730
↪0x30453231 0x3438534e 0xc594c31 0x30313133 0x424a3030 0x31434304 0x35324645
↪0x50520881 0x300000 0x48 0x45043030 0x31304354 0x440b400 0x485702 0x14233 0x6000000
↪0x142 0x34010042 0x370c0000 0x0 0x0 0x4239 0x3c435333 0x22071917 0xd1569c53
↪0x50973c87 0x71f9c40 0x1d4d3142 0x985e80f1 0x5cb3614d 0x32a902cb 0xd9d714ab
↪0x164d3322 0xdda4f986 0x5a618f4d 0x340b157c 0x2cac0a94 0x6504603 0x78 0x0>;
       fru-type = [41 56];
       part-number = "00E2148";
       phandle = <0x8d>;
       linux,phandle = <0x8d>;
};
```

4.3 OPAL API Documentation

The OPAL API is the interface between an Operating System and OPAL.

4.3.1 OPAL_CEC_POWER_DOWN

```
#define OPAL_CEC_POWER_DOWN                    5

int64 opal_cec_power_down(uint64 request)
```

Arguments

```
uint64 request values as follows:
 0 - Power down normally
 1 - Power down immediately
```

This OPAL call requests OPAL to power down the system. The exact difference between a normal and immediate shutdown is platform specific.

Current Linux kernels just use power down normally (0). It is valid for a platform to only support some types of power down operations.

Return Values

OPAL_SUCCESS the power down was updated successful

OPAL_BUSY unable to power down, try again later

OPAL_PARAMETER a parameter was incorrect

OPAL_INTERNAL_ERROR hal code sent incorrect data to hardware device

OPAL_UNSUPPORTED this platform does not support being powered off.

4.3.2 OPAL_CEC_REBOOT and OPAL_CEC_REBOOT2

```
#define OPAL_CEC_REBOOT                6
#define OPAL_CEC_REBOOT2     116
```

There are two opal calls to invoke system reboot.

OPAL_CEC_REBOOT Used for normal reboot by Linux host.

OPAL_CEC_REBOOT2 Newly introduced to handle abnormal system reboots. The Linux kernel will make this OPAL call when it has to terminate abruptly due to an anomalous condition. The kernel will push some system state context to OPAL, which will in turn push it down to the BMC for further analysis.

OPAL_CEC_REBOOT

Syntax:

```
int64_t opal_cec_reboot(void)
```

System reboots normally.

OPAL_CEC_REBOOT2

Syntax:

```
int64_t opal_cec_reboot2(uint32_t reboot_type, char *diag)
```

Input parameters

reboot_type Type of reboot. (see below)

diag Null-terminated string.

Depending on reboot type, this call will carry out additional steps before triggering reboot.

Supported reboot types:

OPAL_REBOOT_NORMAL = 0 Behavior is as similar to that of opal_cec_reboot()

OPAL_REBOOT_PLATFORM_ERROR = 1 Log an error to the BMC and then trigger a system checkstop, using the information provided by 'ibm,sw-checkstop-fir' property in the device-tree. Post the checkstop trigger, OCC/BMC will collect relevant data for error analysis and trigger a reboot.

In absence of 'ibm,sw-checkstop-fir' device property, this function will return with OPAL_UNSUPPORTED and no reboot will be triggered.

Unsupported Reboot type For unsupported reboot type, this function will return with OPAL_UNSUPPORTED and no reboot will be triggered.

4.3.3 OPAL_CHECK_TOKEN

This OPAL call allows the host OS to determine if a particular OPAL call is present on a system. This allows for simple compatibility between OPAL versions and different OPAL implementations/platforms.

One parameter is accepted: the OPAL token number.

`OPAL_CHECK_TOKEN` will return:

```
enum OpalCheckTokenStatus {
  OPAL_TOKEN_ABSENT = 0,
  OPAL_TOKEN_PRESENT = 1
};
```

indicating the presence/absence of the particular OPAL_CALL.

`OPAL_CHECK_TOKEN` is REQUIRED to be implemented by a conformant OPAL implementation.

For skiboot, only positively ancient internal-to-IBM versions were missing OPAL_CHECK_TOKEN. In this case, OPAL_PARAMETER would be returned. There is no reason for a host OS to support this behaviour.

4.3.4 Code Update on FSP based machine

There are three OPAL calls for code update on FSP based machine:

```
#define OPAL_FLASH_VALIDATE    76
#define OPAL_FLASH_MANAGE      77
#define OPAL_FLASH_UPDATE      78
```

OPAL_FLASH_VALIDATE

Validate new image is valid for this platform or not. We do below validation in OPAL:

- We do below sys parameters validation to confirm inband update is allowed. - Platform is managed by HMC or not?. - Code update policy (inband code update allowed?).
- We parse candidate image header (first 4k bytes) to perform below validations. - Image magic number. - Image version to confirm image is valid for this platform.

Input

buffer First 4k bytes of new image

size Input buffer size

Output

buffer Output result (current and new image version details)

size Output buffer size

result Token to identify what will happen if update is attempted See hw/fsp/fsp-codeupdate.h for token values.

Return value

Validation status

OPAL_FLASH_MANAGE

Commit/Reject image.

- We can commit new image (T -> P), if system is running with T side image.
- We can reject T side image, if system is running with P side image.

Note: If a platform is running from a T side image when an update is to be applied, then the platform may automatically commit the current T side image to the P side to allow the new image to be updated to the temporary image area.

Input

op Operation (1 : Commit /0 : Reject)

Return value Commit operation status (0 : Success)

OPAL_FLASH_UPDATE

Update new image. It only sets the flag, actual update happens during system reboot/shutdown.

Host splits FW image to scatter/gather list and sends it to OPAL. OPAL parse the image to get indivisual LID and passes it to FSP via MBOX command.

FW update flow :

- **if (running side == T)** Swap P & T side

- Start code update
- Delete T side LIDs
- Write LIDs
- Code update complete
- Deep IPL

Input

list Real address of image scatter/gather list of the FW image

Return value: Update operation status (0: update requested)

4.3.5 OPAL Console calls

There are four OPAL calls relating to the OPAL console:

```
#define OPAL_CONSOLE_WRITE                      1
#define OPAL_CONSOLE_READ                       2
#define OPAL_CONSOLE_WRITE_BUFFER_SPACE                 25
#define OPAL_CONSOLE_FLUSH                      117
```

The OPAL console calls can support multiple consoles. Each console MUST be represented in the device tree.

A conforming implementation SHOULD have at least one console. It is valid for it to simply be an in-memory buffer and only support writing.

[TODO: details on device tree specs for console]

OPAL_CONSOLE_WRITE

Parameters:

```
int64_t term_number
int64_t *length,
const uint8_t *buffer
```

Returns:

```
OPAL_SUCCESS
OPAL_PARAMETER - invalid term_number
OPAL_CLOSED - console device closed
OPAL_BUSY_EVENT - unable to write any of buffer
```

`term_number` is the terminal number as represented in the device tree. `length` is a pointer to the length of buffer.

A conforming implementation SHOULD try to NOT do partial writes, although partial writes and not writing anything are valid.

OPAL_CONSOLE_WRITE_BUFFER_SPACE

Parameters:

```
int64_t term_number
int64_t *length
```

Returns:

```
OPAL_SUCCESS
OPAL_PARAMETER - invalid term_number
```

Returns the available buffer length for OPAL_CONSOLE_WRITE in `length`. This call can be used to help work out if there is sufficient buffer space to write your full message to the console with OPAL_CONSOLE_WRITE.

OPAL_CONSOLE_READ

Parameters:

```
int64_t term_number
int64_t *length
uint8_t *buffer
```

Returns:

```
OPAL_SUCCESS
OPAL_PARAMETER - invalid term_number
OPAL_CLOSED
```

Use OPAL_POLL_EVENTS for how to determine

OPAL_CONSOLE_FLUSH

Parameters:

```
int64_t term_number
```

Returns:

```
OPAL_SUCCESS
OPAL_UNSUPPORTED - the console does not implement a flush call
OPAL_PARAMETER - invalid term_number
OPAL_PARTIAL - more to flush, call again
OPAL_BUSY - nothing was flushed this call
```

4.3.6 OPAL_ELOG: Error logging

OPAL provides an abstraction to platform specific methods of storing and retrieving error logs. Some service processors may be able to store information in the Platform Error Log (PEL) format. These may be generated at runtime by the service processor or OPAL in reaction to certain events. For example, an IPL failure could be recorded in an error log, as could the reason and details of an unexpected shut-down/reboot (e.g. hard thermal limits, check-stop).

There are five OPAL calls from host to OPAL on error log:

```
#define OPAL_ELOG_READ          71
#define OPAL_ELOG_WRITE         72
#define OPAL_ELOG_ACK           73
```

```
#define OPAL_ELOG_RESEND        74
#define OPAL_ELOG_SIZE          75
```

Note: OPAL_ELOG_WRITE (72) Unused for now, can be used in future.

Not all platforms support these calls, so it's important for a host Operating System to use the OPAL_CHECK_TOKEN call first. If OPAL_ELOG_READ, OPAL_ELOG_ACK, OPAL_ELOG_RESEND, or OPAL_ELOG_SIZE is present, then the rest of that group is also present. The presence of OPAL_ELOG_WRITE must be checked separately.

TODO: we need a good explanation of the notification mechanism and in what order and *when* to call each of the OPAL APIs.

OPAL_ELOG_READ

The OPAL_ELOG_READ call will copy the error log identified by id into the buffer of size size.

OPAL_ELOG_READ accepts 3 parameters:

```
uint64_t *elog_buffer
uint64_t elog_size
uint64_t elog_id
```

Returns:

OPAL_WRONG_STATE When there are no error logs to read, or OPAL_ELOG calls are done in the wrong order.

OPAL_PARAMETER The id does not match the log id that is available.

OPAL_SUCCESS Error log is copied to buffer.

Other generic OPAL error codes may also be returned and should be treated like OPAL_INTERNAL_ERROR.

OPAL_ELOG_ACK

Acknowledging (ACKing) an error log tells OPAL and the service processor that the host operating system has dealt with the error log successfully. This allows OPAL and the service processor to delete the error log from their memory/storage.

OPAL_ELOG_ACK accepts 1 parameter:

```
uint64_t ack_id
```

Returns:

OPAL_INTERNAL_ERROR OPAL failed to send acknowledgement to the error log creator.

OPAL_SUCCESS Success!

Other generic OPAL error codes may also be returned, and should be treated like OPAL_INTERNAL_ERROR.

OPAL_ELOG_RESEND

The OPAL_ELOG_RESEND call will cause OPAL to resend notification to the host operating system of all outstanding error logs. This is commonly used (although doesn't have to be) in a kexec scenario.

The call registered with this token accepts no parameter and returns type is void.

OPAL_ELOG_SIZE

The OPAL_ELOG_SIZE call retrieves information about an error log.

Here, type specifies error log format. Supported types are :

```
0 -> Platform Error Log
```

OPAL_ELOG_SIZE accepts 3 parameters:

```
uint64_t *elog_id
uint64_t *elog_size
uint64_t *elog_type
```

Returns:

OPAL_WRONG_STATE There is no error log to fetch information about.

OPAL_SUCCESS Success.

Other general OPAL errors may be returned.

4.3.7 OPAL Flash calls

There are three OPAL calls for interacting with flash devices:

```
#define OPAL_FLASH_READ         110
#define OPAL_FLASH_WRITE        111
#define OPAL_FLASH_ERASE        112
```

Multiple flash devices are supported by OPAL - each of these calls takes an id parameter, which much match an ID found in the corresponding ibm,opal/flash@n device tree node. See doc/device-tree/ibm,opal/flash.txt for details of the device tree bindings.

All operations on the flash device must be aligned to the block size of the flash. This applies to both offset and size arguments.

This interface is asynchronous; all calls require a 'token' argument. On success, the calls will return OPAL_ASYNC_COMPLETION, and an opal_async_completion message will be sent (with the appropriate token argument) when the operation completes.

All calls share the same return values:

OPAL_ASYNC_COMPLETION operation started, an async completion will be triggered with the token argument

OPAL_PARAMETER invalid flash id

OPAL_PARAMETER invalid size or offset (alignment, or access beyond end of device)

OPAL_BUSY flash in use

OPAL_HARDWARE error accessing flash device

OPAL_FLASH_READ

Parameters:

```
uint64_t id
uint64_t offset
uint64_t buffer
uint64_t size
uint64_t token
```

Reads from the specified flash id, at the specified offset, into the buffer. Will trigger an async completion with token when completed.

OPAL_FLASH_ERASE

Parameters:

```
uint64_t id
uint64_t offset
uint64_t size
uint64_t token
```

Erases the specified flash id, at the specified offset and size. Will trigger an async completion with token when completed.

OPAL_FLASH_WRITE

Parameters:

```
uint64_t id
uint64_t offset
uint64_t buffer
uint64_t size
uint64_t token
```

Writes buffer to the specified flash id, at the specified offset and size. The flash must be erased before being written. Will trigger an async completion with token when completed.

4.3.8 OPAL_GET_DEVICE_TREE

Get device sub-tree.

Parameters:

```
uint32_t phandle: root device node phandle of the device sub-tree
uint64_t buf: FDT blob buffer or NULL
uint64_t len: length of the FDT blob buffer
```

Calling:

Retrieve device sub-tree. The root node's phandle is identified by @phandle. The typical use is for the kernel to update its device tree following a change in hardware (e.g. PCI hotplug).

Return Codes:

FDT blob size returned FDT blob buffer size when `buf` is NULL

OPAL_SUCCESS FDT blob is created successfully

OPAL_PARAMETER invalid argument @phandle or @len

OPAL_INTERNAL_ERROR failure creating FDT blob when calculating its size

OPAL_NO_MEM not enough room in buffer for device sub-tree

OPAL_EMPTY failure creating FDT blob

4.3.9 OPAL_GET_MSG

OPAL_GET_MSG will get the next pending OPAL Message (see opal-messages.txt).

Parameters:

```
buffer to copy message into
sizeof buffer to copy message into
```

The maximum size of an opal message is specified in the device tree passed to the host OS:

```
ibm,opal {
          opal-msg-size = <0x48>;
}
```

It is ALWAYS at least 72 bytes. In the future, OPAL may have messages larger than 72 bytes. Naturally, a HOST OS will only be able to interpret these if it correctly uses opal-msg-size. Any OPAL message > 72 bytes, a host OS may safely ignore.

A host OS *SHOULD* always supply a buffer to OPAL_GET_MSG of either 72 bytes or opal-msg-size. It MUST NOT supply a buffer of < 72 bytes.

Return values

OPAL_RESOURCE no available message.

OPAL_PARAMETER buffer is NULL or size is < 72 bytes. If buffer size < 72 bytes, the message will NOT be discarded by OPAL.

OPAL_PARTIAL If pending opal message is greater than supplied buffer. In this case the message is *DISCARDED* by OPAL. This is to keep compatibility with host Operating Systems with a hard coded opal-msg-size of 72 bytes. **NOT CURRENTLY IMPLEMENTED**. Specified so that host OS can prepare for the possible future with either a sensible error message or by gracefully ignoring such OPAL messages.

OPAL_SUCCESS message successfully copied to buffer.

4.3.10 OPAL_GET_MSI_32 and OPAL_GET_MSI_64

```
#define OPAL_GET_MSI_32                                 39
#define OPAL_GET_MSI_64                                 40
```

WARNING: the following documentation is from old sources, and is possibly not representative of OPALv3 as implemented by skiboot. This should be used as a starting point for full documentation.

OPAL PHBs encode MVE and XIVE specifiers in MSI DMA and message data values. The host calls these functions to determine the PHB MSI DMA address and message data to program into a PE PCIE function for a particular MVE and XIVE. The msi_address parameter returns the MSI DMA address and the msi_data parameter returns the MSI DMA message data value the PE uses to signal that interrupt.

`phb_id` The `phb_id` parameter is the value from the PHB node `ibm,opal-phbid` property.

mve_number The mve_number is the index of an MVE used to authorize this PE to this MSI. For ibm,opal-ioda2 PHBs, the MVE number argument is ignored.

xive_number The xive_number is the index of an XIVE that corresponds to a particular DMA address and message data value this PE will signal as an MSI ro MSI-X.

msi_range The msi_range parameter specifies the number of MSIs associated with the in put MVE and XIVE, primarily for MSI-conventional Multiple Message Enable > 1 MSI. MSI requires consecutive MSIs per MSI address, and each MSI DMA address must be unique for any given consecutive power of 2 set of 32 message data values,. which in turn select particular PHB XIVEs. This value must be a power of 2 value in the range of 0 to 32. OPAL returns opal_parameter for values outside of this range.

For MSI conventional, the MSI address and message data returned apply to a power of 2 sequential set of XIVRs starting from the xive_number for the power of 2 msi_range input argument. The message data returned represents the power of 2 aligned starting message data value of the first interrupt number in that sequential range. Valid msi_range input values are from 1 to 32. Non-power of 2 values result in a return code of opal_PARAMETER .

An msi_range value of 0 or 1 signifies that OPAL should return the message data and message address for exactly one MSI specified by the input XIVE number. For MSI conventional, the host should specify either a value of 0 or 1, for an MSI Capability MME value of 1 MSI. For MSI-X XIVRs, the host should specify a value of '1' for the msi_range argument and call this function for each MSI-X uniquely.

4.3.11 OPAL_GET_XIVE

```
#define OPAL_GET_XIVE                                   20
```

WARNING: following documentation is from old sources, and is possibly not representative of OPALv3 as implemented by skiboot. This should be used as a starting point for full documentation.

The host calls this function to return the POWER XIVE server and priority values currently set in a PHB XIVE.

phb_id The phb_id parameter is the value from the PHB node ibm,opal-phbid property.

xive_number The xive_number is the index of an XIVE that corresponds to a particular interrupt.

server_number the server_number returns the server (processor) that is set in this XIVE

priority the priority returns the interrupt priority value that is set in this XIVE

This call returns the server and priority numbers from within the XIVE specified by the XIVE_number.

4.3.12 OPAL_GET_XIVE_SOURCE

```
This function validates the given ``xive_num`` and sets the
``interrupt_source_number``. Then returns the proper return code.
```

Parameters

phb_id The phb_id parameter is the value from the PHB node ibm,opal-phbid property.

xive_num The xive_num is the index of an XIVE that corresponds to a particular interrupt.

interrupt_source_number The interrupt_source_number is a value formed by the combination of the device tree MSI property base BUID and xive_num

Return Codes

OPAL_PARAMETER The indicated phb_id not found

OPAL_UNSUPPORTED Presence retrieval not supported on the phb_id

OPAL_SUCCESS Indicates Success!

4.3.13 OPAL_HANDLE_INTERRUPT

The host OS must pass all interrupts in ibm,opal/opal-interrupts in the device tree to OPAL.

An example dt snippet is:

```
ibm,opal {
          ...
          opal-interrupts = <0x10 0x11 0x12 0x13 0x14 0x20010 0x20011 0x20012 0x20013␣
↪0x20014 0xffe 0xfff 0x17fe 0x17ff 0x2ffe 0x2fff 0x37fe 0x37ff 0x20ffe 0x20fff␣
↪0x217fe 0x217ff 0x22ffe 0x22fff 0x237fe 0x237ff>;
}
```

When the host OS gets any of these interrupts, it must call OPAL_HANDLE_INTERRUPT.

The OPAL_HANDLE_INTERRUPT call takes two parameters, one input and one output.

uint32_t isn the interrupt

uint64_t *outstanding_event_mask returns outstanding events for host OS to handle

The host OS should then handle any outstanding events.

See opal-poll-events.txt for documentation on events.

4.3.14 OPAL_INT_EOI

```
static int64_t opal_xive_eoi(uint32_t xirr)
```

Not yet implemented.

Modelled on the H_EOI PAPR call.

This can return a positive value, which means more interrupts are queued for that CPU/priority and must be fetched as the XIVE is not guaranteed to assert the CPU external interrupt line again until the pending queue for the current priority has been emptied.

For P9 and above systems where host doesn't know about interrupt controller. An OS can instead make OPAL calls for XICS emulation.

For an OS to use this OPAL call, an ibm,opal-intc compatible device must exist in the device tree. If OPAL does not create such a device, the host OS MUST NOT use this call.

4.3.15 OPAL_INT_GET_XIRR

```
int64_t opal_xive_get_xirr(uint32_t *out_xirr, bool just_poll)
```

Not yet implemented.

Modelled on the PAPR call.

For P9 and above systems where host doesn't know about interrupt controller. An OS can instead make OPAL calls for XICS emulation.

For an OS to use this OPAL call, an `ibm,opal-intc` compatible device must exist in the device tree. If OPAL does not create such a device, the host OS MUST NOT use this call.

4.3.16 OPAL_INT_SET_CPPR

```
static int64_t opal_xive_set_cppr(uint8_t cppr)
```

Not yet implemented.

Modelled on the `H_CPPR` PAPR call.

For P9 and above systems where host doesn't know about interrupt controller. An OS can instead make OPAL calls for XICS emulation.

For an OS to use this OPAL call, an `ibm,opal-intc` compatible device must exist in the device tree. If OPAL does not create such a device, the host OS MUST NOT use this call.

4.3.17 OPAL_INT_SET_MFRR

```
static int64_t opal_xive_set_mfrr(uint32_t cpu, uint8_t mfrr)
```

Not yet implemented.

Modelled on the `H_IPI` PAPR call.

For P9 and above systems where host doesn't know about interrupt controller. An OS can instead make OPAL calls for XICS emulation.

For an OS to use this OPAL call, an `ibm,opal-intc` compatible device must exist in the device tree. If OPAL does not create such a device, the host OS MUST NOT use this call.

4.3.18 OPAL_INVALID_CALL

An OPAL call of `-1` will always return `OPAL_PARAMETER`. It is always ivalid.

It exists purely for testing.

4.3.19 OPAL_IPMI_SEND

```
#define OPAL_IPMI_SEND                          107
```

`OPAL_IPMI_SEND` call will send an IPMI message to the service processor.

Parameters

```
uint64_t interface
struct opal_ipmi_msg *opal_ipmi_msg
uint64_t msg_len
```

interface interface parameter is the value from the ipmi interface node ibm,ipmi-interface-id

opal_ipmi_msg opal_ipmi_msg is the pointer to below structure opal_ipmi_msg

```
struct opal_ipmi_msg {
     uint8_t version;
     uint8_t netfn;
     uint8_t cmd;
     uint8_t data[];
};
```

msg_len ipmi message request size

Return Values

OPAL_SUCCESS msg queued successfully

OPAL_PARAMETER invalid ipmi message request length msg_len

OPAL_HARDWARE backend support is not present as block transfer/service processor ipmi routines are not initialized which are used for communication

OPAL_UNSUPPORTED in-correct opal ipmi message format version opal_ipmi_msg->version

OPAL_RESOURCE insufficient resources to create ipmi_msg structure

4.3.20 OPAL_IPMI_RECV

```
#define OPAL_IPMI_RECV                          108
```

OPAL_IPMI_RECV call reads an ipmi message of type ipmi_msg from ipmi message queue msgq into host OS structure opal_ipmi_msg.

Parameters

```
uint64_t interface
struct opal_ipmi_msg *opal_ipmi_msg
uint64_t *msg_len
```

interface interface parameter is the value from the ipmi interface node ibm,ipmi-interface-id

opal_ipmi_msg opal_ipmi_msg is the pointer to below structure opal_ipmi_msg

```
struct opal_ipmi_msg {
     uint8_t version;
     uint8_t netfn;
     uint8_t cmd;
     uint8_t data[];
};
```

msg_len msg_len is the pointer to ipmi message response size

Return Values

OPAL_SUCCESS ipmi message dequeued from `msgq` queue and memory taken by it got released successfully

OPAL_EMPTY `msgq` list is empty

OPAL_PARAMETER invalid ipmi `interface` value

OPAL_UNSUPPORTED in-correct opal ipmi message format version `opal_ipmi_msg->version`

4.3.21 Service Indicators (LEDS)

The service indicator is one element of an overall hardware service strategy where end user simplicity is a high priority. The goal is system firmware or operating system code to isolate hardware failures to the failing FRU and automatically activate the fault indicator associated with the failing FRU. The end user then needs only to look for the FRU with the active fault indicator to know which part to replace.

Different types of indicators handled by LED code:

- **System attention indicator (Check log indicator)** Indicates there is a problem with the system that needs attention.
- **Identify** Helps the user locate/identify a particular FRU or resource in the system.
- **Fault** Indicates there is a problem with the FRU or resource at the location with which the indicator is associated.

LED Design

When it comes to implementation we can classify LEDs into two categories:

1. Hypervisor (OPAL) controlled LEDs (All identify & fault indicators) During boot, we read/cache these LED details in OPAL (location code, state, etc). We use cached data to serve read request from FSP/Host. And we use SPCN passthrough MBOX command to update these LED state.
2. Service processor (FSP) controlled LEDs (System Attention Indicator) During boot, we read/cache this LED info using MBOX command. Later anytime FSP updates this LED, it sends update system parameter notification MBOX command. We use that data to update cached data. LED update request is sent via set/reset attn MBOX command.

LED update request: Both FSP and Host will send LED update requests. We have to serialize SPCN passthrough command. Hence we maintain local queue.

Note:

- For more information regarding service indicator refer to PAPR spec (Service Indicators chapter).

There are two OPAL calls relating to LED operations.

OPAL_LEDS_GET_INDICATOR

Returns LED state for the given location code.

OPAL_LEDS_SET_INDICATOR

Sets LED state for the given location code.

See hw/fsp/fsp-leds.c for more deatails.

4.3.22 OPAL_MESSAGE

The host OS can use OPAL_GET_MSG to retrive messages queued by OPAL. The messages are defined by enum opal_msg_type. The host is notified of there being messages to be consumed by the OPAL_EVENT_MSG_PENDING bit being set.

An opal_msg is:

```
struct opal_msg {
      __be32 msg_type;
      __be32 reserved;
      __be64 params[8];
};
```

The data structure is ALWAYS at least this size (4+4+8*8 = 72 bytes). Some messages define fewer than eight parameters. For messages that do not define all eight parameters, the value in the undefined parameters is undefined, although can safely be memcpy()d or otherwise moved.

In the device tree, there's an opal-msg-size property of the OPAL node that says the size of a struct opal-msg. In the future, OPAL may support larger messages. See `OPAL_GET_MESSAGE` documentation for details.

```
ibm,opal {
          opal-msg-size = <0x48>;
}
```

OPAL_MSG_ASYNC_COMP

```
params[0] = token
params[1] = rc
```

Additional parameters are function-specific.

OPAL_MSG_MEM_ERR

OPAL_MSG_EPOW

Used by OPAL to issue environmental and power warnings to host OS for conditions requiring an earlier poweroff. A few examples of these are high ambient temperature or system running on UPS power with low UPS battery. Host OS can query OPAL via `GET_EPOW_STATUS` API to obtain information about EPOW conditions present. Refer include/opal-api.h for description of all supported EPOW events. OPAL_SYSPOWER_CHNG, OPAL_SYSPOWER_FAIL and OPAL_SYSPOWER_INC events don't require system poweroff.

Host OS should look for 'ibm,opal-v3-epow' string as compatible property for 'epow' node under OPAL device-tree to determine epow support.

OPAL_MSG_SHUTDOWN

Used by OPAL to inform the host OS it must imitate a graceful shutdown. Uses the first parameter to indicate weather the system is going down for shutdown or a reboot.

```
params[0] = 0x01 reboot, 0x00 shutdown
```

OPAL_MSG_HMI_EVT

Used by OPAL to sends the OPAL HMI Event to the host OS that reports a summary of HMI error and whether it was successfully recovered or not.

HMI is a Hypervisor Maintenance Interrupt usually reports error related to processor recovery/checkstop, NX checkstop and Timer facility. Hypervisor then takes this opportunity to analyze and recover from some of these errors. Hypervisor takes assistance from OPAL layer to handle and recover from HMI. After handling HMI, OPAL layer sends the summary of error report and status of recovery action using HMI event structure shown below.

The HMI event structure uses version numbering to allow future enhancement to accommodate additional members. The version start from V1 onward. Version 0 is invalid version and unsupported.

The current version of HMI event structure V2 and is backward compatible to V1 version.

Notes:

- When adding new structure to the union in future, the version number must be bumped.
- All future versions must be backward compatible to all its older versions.
- Size of this structure should not exceed that of struct opal_msg.

```
struct OpalHMIEvent {
     uint8_t         version;        /* 0x00 */
     uint8_t         severity;       /* 0x01 */
     uint8_t         type;           /* 0x02 */
     uint8_t         disposition;    /* 0x03 */
     uint8_t         reserved_1[4];  /* 0x04 */

     __be64          hmer;
     /* TFMR register. Valid only for TFAC and TFMR_PARITY error type. */
     __be64          tfmr;

     /* version 2 and later */
     union {
             /*
              * checkstop info (Core/NX).
              * Valid for OpalHMI_ERROR_MALFUNC_ALERT.
              */
             struct {
                     uint8_t xstop_type;     /* enum OpalHMI_XstopType */
                     uint8_t reserved_1[3];
                     __be32 xstop_reason;
                     union {
                             __be32 pir;       /* for CHECKSTOP_TYPE_CORE */
                             __be32 chip_id; /* for CHECKSTOP_TYPE_NX */
                     } u;
             } xstop_error;
     } u;
};
```

OPAL_MSG_DPO

Delayed poweroff where OPAL informs host OS that a poweroff has been requested and a forced shutdown will happen in future. Host OS can use OPAL_GET_DPO_STATUS API to query OPAL the number of seconds remaining before a forced poweroff will occur.

OPAL_MSG_PRD

This message is a OPAL-to-HBRT notification, and contains a struct opal_prd_msg:

```
enum opal_prd_msg_type {
        OPAL_PRD_MSG_TYPE_INIT = 0,     /* HBRT --> OPAL */
        OPAL_PRD_MSG_TYPE_FINI,         /* HBRT --> OPAL */
        OPAL_PRD_MSG_TYPE_ATTN,         /* HBRT <-- OPAL */
        OPAL_PRD_MSG_TYPE_ATTN_ACK,     /* HBRT --> OPAL */
        OPAL_PRD_MSG_TYPE_OCC_ERROR,    /* HBRT <-- OPAL */
        OPAL_PRD_MSG_TYPE_OCC_RESET,    /* HBRT <-- OPAL */
};

struct opal_prd_msg {
        uint8_t         type;
        uint8_t         pad[3];
        __be32          token;
        union {
                struct {
                        __be64  version;
                        __be64  ipoll;
                } init;
                struct {
                        __be64  proc;
                        __be64  ipoll_status;
                        __be64  ipoll_mask;
                } attn;
                struct {
                        __be64  proc;
                        __be64  ipoll_ack;
                } attn_ack;
                struct {
                        __be64  chip;
                } occ_error;
                struct {
                        __be64  chip;
                } occ_reset;
        };
};
```

Responses from the kernel use the same message format, but are passed through the opal_prd_msg call.

OPAL_MSG_OCC

This is used by OPAL to inform host about OCC events like OCC reset, OCC load and throttle status change by OCC which can indicate the host the reason for frequency throttling/unthrottling.

```
#define OCC_RESET                       0
#define OCC_LOAD                        1
#define OCC_THROTTLE                    2
#define OCC_MAX_THROTTLE_STATUS                 5
/*
 * struct opal_occ_msg:
 * type: OCC_RESET, OCC_LOAD, OCC_THROTTLE
 * chip: chip id
 * throttle status: Indicates the reason why OCC may have limited
 * the max Pstate of the chip.
```

```
 * 0x00 = No throttle
 * 0x01 = Power Cap
 * 0x02 = Processor Over Temperature
 * 0x03 = Power Supply Failure (currently not used)
 * 0x04 = Over current (currently not used)
 * 0x05 = OCC Reset (not reliable as some failures will not allow for
 * OCC to update throttle status)
 */
struct opal_occ_msg {
      __be64 type;
      __be64 chip;
      __be64 throttle_status;
};
```

Host should read opal_occ_msg.chip and opal_occ_msg.throttle_status only when `opal_occ_msg.type = OCC_THROTTLE`. If host receives `OCC_THROTTLE` after an `OCC_RESET` then this throttle message will have a special meaning which indicates that all the OCCs have become active after a reset. In such cases `opal_occ_msg.chip` and `opal_occ_msg.throttle_status` will be set to 0 and host should not use these values.

If `opal_occ_msg.type > 2` then host should ignore the message for now, new events can be defined for `opal_occ_msg.type` in the future versions of OPAL.

4.3.23 OPAL_READ_NVRAM

```
#define OPAL_READ_NVRAM                         7
```

`OPAL_READ_NVRAM` call requests OPAL to read the data from system NVRAM memory into a memory buffer. The data at `offset` from nvram_image will be copied to memory `buffer` of size `size`.

Parameters

```
uint64_t buffer
uint64_t size
uint64_t offset
```

`buffer` the data from nvram will be copied to `buffer`

`size` the data of size `size` will be copied

`offset` the data will be copied from address equal to base `nvram_image` plus `offset`

Return Values

`OPAL_SUCCESS` data from nvram to memory `buffer` copied successfully

`OPAL_PARAMETER` a parameter `offset` or `size` was incorrect

`OPAL_HARDWARE` either nvram is not initialized or permanent error related to nvram hardware.

4.3.24 OPAL_WRITE_NVRAM

```
#define OPAL_WRITE_NVRAM                        8
```

OPAL_WRITE_NVRAM call requests OPAL to write the data to actual system NVRAM memory from memory `buffer` at `offset`, of size `size`

Parameters

```
uint64_t buffer
uint64_t size
uint64_t offset
```

buffer data from `buffer` will be copied to nvram

size the data of size `size` will be copied

offset the data will be copied to address which is equal to base `nvram_image` plus `offset`

Return Values

OPAL_SUCCESS data from memory `buffer` to actual nvram_image copied successfully

OPAL_PARAMETER a parameter `offset` or `size` was incorrect

OPAL_HARDWARE either nvram is not initialized or permanent error related to nvram hardware.

4.3.25 OPAL_PCI_GET_PHB_DIAG_DATA2

Get PCI diagnostic data from a given PHB

Parameters

uint64_t phb_id the ID of the PHB you want to retrieve data from

void *diag_buffer an allocated buffer to store diag data in

uint64_t diag_buffer_len size in bytes of the diag buffer

Calling

Retrieve the PHB's diagnostic data. The diagnostic data is stored in the buffer pointed by @diag_buffer. Different PHB versions will store different diagnostics, defined in include/opal-api.h as `struct OpalIo<PHBVer>ErrorData`.

`OPAL_PCI_GET_PHB_DIAG_DATA` is deprecated and `OPAL_PCI_GET_PHB_DIAG_DATA2` should be used instead.

Return Codes

OPAL_SUCCESS Diagnostic data has been retrieved and stored successfully

OPAL_PARAMETER The given buffer is too small to store the diagnostic data

OPAL_HARDWARE The PHB is in a broken state and its data cannot be retreived

OPAL_UNSUPPORTED Diagnostic data is not implemented for this PHB type

4.3.26 OPAL_PCI_GET_POWER_STATE

Get PCI slot power state

Parameter

uint64_t id PCI slot ID

uint64_t data memory buffer pointer for power state

Calling

Retrieve PCI slot's power state. The retrieved power state is stored in buffer pointed by @data.

Return Codes

OPAL_SUCCESS PCI slot's power state is retrieved successfully

OPAL_PARAMETER The indicated PCI slot isn't found

OPAL_UNSUPPORTED Power state retrieval not supported on the PCI slot

4.3.27 OPAL_PCI_GET_PRESENCE_STATE

Get PCI slot presence state

Parameters

uint64_t id PCI slot ID

uint64_t data memory buffer pointer for presence state

Calling

Retrieve PCI slot's presence state. The detected presence means there are adapters inserted to the PCI slot. Otherwise, the PCI slot is regarded as an empty one. The typical use is to ensure there are adapters existing before probing the PCI slot in PCI hot add path. The retrieved presence state is stored in buffer pointed by @data.

Return Codes

OPAL_SUCCESS PCI slot's presence state is retrieved successfully

OPAL_PARAMETER The indicated PCI slot isn't found

OPAL_UNSUPPORTED Presence retrieval not supported on the PCI slot

4.3.28 OPAL_PCI_GET_XIVE_REISSUE and OPAL_PCI_SET_XIVE_REISSUE

```
static int64_t opal_pci_get_xive_reissue(uint64_t phb_id __unused,
                                         uint32_t xive_number __unused,
                                         uint8_t *p_bit __unused,
                                         uint8_t *q_bit __unused)

static int64_t opal_pci_set_xive_reissue(uint64_t phb_id __unused,
                                         uint32_t xive_number __unused,
                                         uint8_t p_bit __unused,
                                         uint8_t q_bit __unused)
```

Both of these calls are remnants from previous OPAL versions, calling either of them shall return `OPAL_UNSUPPORTED`.

4.3.29 OPAL_PCI_MAP_PE_DMA_WINDOW

```
#define OPAL_PCI_MAP_PE_DMA_WINDOW           44

static int64_t opal_pci_map_pe_dma_window(uint64_t phb_id,
                                          uint64_t pe_number,
                                          uint16_t window_id,
                                          uint16_t tce_levels,
                                          uint64_t tce_table_addr,
                                          uint64_t tce_table_size,
                                          uint64_t tce_page_size)
```

WARNING: following documentation is from old sources, and is possibly not representative of OPALv3 as implemented by skiboot. This should be used as a starting point for full documentation.

The host calls this function to create a DMA window and map it to a PE. This call returns the address in PCI memory that corresponds to the specified DMA window, which in part may depend on the particular PHB DMA window used. An address that is all zeros in the upper 32 bits reflects a DMA window enabled for 32-bit DMA addresses.

The overall size of the DMA window in PCI memory is determined by the number of tce_levels times the tce_table_size times the tce_page_size.

`phb_id` is the value from the PHB node `ibm,opal-phbid` property.

`dma_window_number` specifies the DMA window

For ibm,opal-ioda PHBs the dma_window_number is an index from 0 to the PHB total number of windows minus 1. For ibm,opal-ioda2 PHBs the DMA window_number is an index from 0 to n-1, where n is the number of windows per window set, within the window set associated with the specified PE number.

`pe_number` is the index of the PE that is authorized to DMA to this window address space in PCI memory,

`tce_levels` is the number of TCE table levels in the translation hiearchy, from 1 to ibm,opal-dmawins property <translation levels>.

`tce_table_addr` is the 64-bit system real address of the first level (root, for mult-level) TCE table in the translation hiearchy.

`tce_table_size` is the size, in bytes, of each TCE table in the translation hierarchy. A value of '0' indicates to disable this DMA window.

For ibm,opal-ioda, this must be a value in the range from 128MB / tce_page_size to 256TB / tce_page_size, and must be in the format and matching a value in the tce_table ranges property that is minimally 256KB for 4K pages.

A particular PE may be mapped to multiple DMA windows, each spanning a DMA window size corresponding to the win_size32 or win_size_64 specified in the ibm,opal-dmawins<> property. However, the TCE table base address must be unique for each window unless it is intended that the same page address in each DMA window is mapped through the same TCE table entry. Generally, when mapping the same PE to multiple DMA windows, so as to create a larger overall DMA window, it is recommended to use consecutive DMA windows and each DMA window should use a TCE table address that is offset by the win_size value of predecessor DMA window.

tce_page_size is the size of PCI memory pages mapped to system real pages through all TCE tables in the translation hierarchy. This must be the same format as and match a value from the ibm,opal-dmawins property <dma-page-sizes>. This page size applies to all TCE tables in the translation hierarchy.

pci_start_addr returns the starting address in PCI memory that corresponds to this DMA window based on the input translation parameter values.

pci_mem_type selects whether this DMA window should be created in 32-bit or 64-bit PCI memory. The input values correspond to the same PCI memory space locators as MMIO spaces in the ranges<> property – 0x2 indicated 32-bit PCI memory and 0x3 indicates 64-bit memory.

Window 0 for both ibm,opal-ioda and ibm,opal-ioda2 PHBs must be within 32-bit PCI memory and this call return opal_parameter for calls that specify window 0 in 64-bit PCI memory.

The DMA win_size property for 32 bit DMA windows limits the number of ibm,opal-ioda PHB windows that can map32-bit address space. For example, with a win_size_32 = 256MB, only 16 DMA windows (and therefore no more than 16 distinct PEs) can map the 4GB of 32-bit PCI memory for DMA. OPAL does not police this limitation.

Return value:

```
if (!phb)
        return OPAL_PARAMETER;
if (!phb->ops->map_pe_dma_window)
        return OPAL_UNSUPPORTED;
```

4.3.30 OPAL_PCI_MAP_PE_DMA_WINDOW_REAL

```
#define OPAL_PCI_MAP_PE_DMA_WINDOW_REAL              45
```

WARNING: following documentation is from old sources, and is possibly not representative of OPALv3 as implemented by skiboot. This should be used as a starting point for full documentation.

The host calls this function to initialize the specified DMA window for untranslated DMA addresses. This allows a PE to DMA directly to system memory without TCE translation. The DMA window PCI memory address is equal to the system memory real address. The PHB passes PCI address bits 04:63 directly to system real address bits 04:63 when PCI address bits 04:39 are within the region specified by mem_addr t0 mem_addr + window_size.

The addresses must be 16MB aligned and a multiple of 16MB in size.

phb_id is the value from the PHB node ibm,opal-phbid property.

dma_window_number specifies the DMA window

For ibm,opal-ioda PHBs the dma_window_number is an index from 0 to the PHB total number of windows minus 1. For ibm,opal-ioda2 PHBs the DMA window_number is an index from 0 to n-1, where n is the number of windows per window set, within the window set associated with the specified PE number.

pe_number is the index of the PE that is authorized to DMA to this window address space in PCI memory,

mem_addr is the starting 64-bit system real address mapped directly to the starting address in PCI memory. Addresses below 4GB are zero in bits above bit 32. This value must be aligned on a 16MB boundary; OPAL returns OPAL_PARAMETER for any value that is not a multiple of 16MB.

window_size is the size, in bytes, of the address range defined by this window. This value must be a multiple of 16MB; OPAL returns OPAL_PARAMETER for any value that is not a multiple of 16MB. A value of '0' indicates to disable this DMA window.

4.3.31 OPAL_PCI_MAP_PE_MMIO_WINDOW

```
#define OPAL_PCI_MAP_PE_MMIO_WINDOW          29

static int64_t opal_pci_map_pe_mmio_window(uint64_t phb_id,
                                           uint64_t pe_number,
                                           uint16_t window_type,
                                           uint16_t window_num,
                                           uint16_t segment_num)
```

WARNING: following documentation is from old sources, and is possibly not representative of OPALv3 as implemented by skiboot. This should be used as a starting point for full documentation.

The host calls this function to map a segment of MMIO address space to a PE.

phb_id is the value from the PHB node ibm,opal-phbid property.

window_type specifies 32-bit or 64-bit PCI memory

'0' selects PCI IO Space. ibm,opal-ioda2 PHBs do not support IO space, and OPAL returns opal_unsupported if called for IO windows.

'1' selects 32-bit PCI memory space

'2' selects 64 bit PCI memory space

window_num is the MMIO window number within the specified PCI memory space

segment_num is an index from 0 to the number of segments minus 1 defined or this window, and selects a particular segment within the specified window.

Return value:

```
if (!phb)
        return OPAL_PARAMETER;
if (!phb->ops->map_pe_mmio_window)
        return OPAL_UNSUPPORTED;
```

4.3.32 OPAL_PCI_PHB_MMIO_ENABLE

```
#define OPAL_PCI_PHB_MMIO_ENABLE             27

static int64_t opal_pci_phb_mmio_enable(uint64_t phb_id, uint16_t window_type,
                                        uint16_t window_num, uint16_t enable)
```

WARNING: following documentation is from old sources, and is possibly not representative of OPALv3 as implemented by skiboot. This should be used as a starting point for full documentation.

The host calls this function to enable or disable PHB decode of the PCI IO and Memory address spaces below that PHB. Window_num selects an mmio window within that address space. Enable set to '1' enables the PHB to decode and forward system real addresses to PCI memory, while enable set to '0' disables PHB decode and forwarding for the address range defined in a particular MMIO window.

Not all PHB hardware may support disabling some or all MMIO windows. OPAL returns OPAL_UNSUPPORTED if called to disable an MMIO window for which hardware does not support disable. KVM may call this function for all MMIO windows and ignore the opal_unsuppsorted return code so long as KVM has disabled MMIO to all downstream PCI devices and assured that KVM and OS guest partitions cannot issue CI loads/stores to these address spaces from the processor (e.g.,via HPT).

OPAL returns OPAL_SUCCESS for calls to OPAL to enable them for PHBs that do not support disable.

phb_id is the value from the PHB node ibm,opal-phbid property.

window_type specifies 32-bit or 64-bit PCI memory

'0' selects PCI IO Space

'1' selects 32-bit PCI memory space

'2' selects 64 bit PCI memory space

window_num is the MMIO window number within the specified PCI memory space

enable specifies to enable or disable this MMIO window.

4.3.33 OPAL_PCI_SET_MVE

```
#define OPAL_PCI_SET_MVE                        33

static int64_t opal_pci_set_mve(uint64_t phb_id, uint32_t mve_number,
                                uint64_t pe_number)
```

WARNING: following documentation is from old sources, and is possibly not representative of OPALv3 as implemented by skiboot. This should be used as a starting point for full documentation.

The host calls this function to bind a PE to an MSI Validation Table Entry (MVE) in the PHB. The MVE compares the MSI requester (RID) to a PE RID, including within the XIVE, to validate that the requester is authorized to signal an interrupt to the associated DMA address for a message value that selects a particular XIVE.

phb_id is the value from the PHB node ibm,opal-phbid property.

mve_number is the index, from 0 to ibm,opal,ibm-num-msi-ports minus1

pe_number is the index of a PE, from 0 to ibm,opal-num-pes minus 1.

This call maps an MVE to a PE and PE RID domain. OPAL uses the PELT to determine the PE domain. OPAL treats this call as a NOP for IODA2 PHBs and returns a status of OPAL_SUCCESS.

Return value:

```
if (!phb)
        return OPAL_PARAMETER;
if (!phb->ops->set_mve)
        return OPAL_UNSUPPORTED;
```

4.3.34 OPAL_PCI_SET_MVE_ENABLE

```
#define OPAL_PCI_SET_MVE_ENABLE                        34

static int64_t opal_pci_set_mve_enable(uint64_t phb_id, uint32_t mve_number,
                                       uint32_t state)
```

```
enum OpalMveEnableAction {
     OPAL_DISABLE_MVE = 0,
     OPAL_ENABLE_MVE = 1
};
```

WARNING: following documentation is from old sources, and is possibly not representative of OPALv3 as implemented by skiboot. This should be used as a starting point for full documentation.

The host calls this function to enable or disable an MVE to respond to an MSI DMA address and message data value.

phb_id is the value from the PHB node ibm,opal-phbid property.

mve_number is the index, from 0 to ibm,opal,ibm-num-msi-ports minus1

state A '1' value of the state parameter indicates to enable the MVE and a '0' value indicates to disable the MVE.

This call sets the MVE to an enabled (1) or disabled (0) state.

Return value:

```
if (!phb)
        return OPAL_PARAMETER;
if (!phb->ops->set_mve_enable)
        return OPAL_UNSUPPORTED;
```

4.3.35 OPAL_PCI_SET_PE

```
#define OPAL_PCI_SET_PE                                 31
```

NOTE: The following two paragraphs come from some old documentation and have not been checked for accuracy. Same goes for bus_compare, dev_compare and func_compare documentation. Do *NOT* assume this documentation is correct without checking the source.

A host OS calls this function to map a PCIE function (RID), or range of function bus/dev/funcs (RIDs), to a PHB PE. The bus, device, func, and compare parameters define a range of bus, device, or function numbers to define a range of RIDs within this domain. A value of "7" for the bus_compare, and non-zero for the dev_compare and func_compare, define exactly one function RID to be a PE (within a PE number domain).

This must be called prior to ALL other OPAL calls that take a PE number argument, for OPAL to correlate the RID (bus/dev/func) domain of the PE. If a PE domain is changed, the host must call this to reset the PE bus/dev/func domain and then call all other OPAL calls that map PHB IODA resources to update those domains within PHB facilities.

```
static int64_t opal_pci_set_pe(uint64_t phb_id, uint64_t pe_number,
                               uint64_t bus_dev_func, uint8_t bus_compare,
                               uint8_t dev_compare, uint8_t func_compare,
                               uint8_t pe_action)
```

phb_id is the value from the PHB node ibm,opal-phbid property.

pe_number is the index of a PE, from 0 to ibm,opal-num-pes minus 1.

bus_compare is a value from 0 to 7 indicating which bus number bits define the range of buses in a PE domain:

0 = do not validate against RID bus number (PE = all bus numbers)

2 = compare high order 3 bits of RID bus number to high order 3 bits of PE bus number

3 = compare high order 4 bits of RID bus number to high order 4 bits of PE bus number

6 = compare high order 7 bits of RID bus number to high order 7 bits of PE bus number

7 = compare all bits of RID bus number to all bits of PE bus number

dev_compare indicates to compare the RID device number to the PE device number or not. '0' signifies that the RID device number is not compared – essentially all device numbers within the bus and function number range of this PE are also within this PE. Non-zero signifies to compare the RID device number to the PE device number, such that only that device number is in the PE domain, for all buses and function numbers in the PE domain.

func_compare indicates to compare the RID function number to the PE function number or not. '0' signifies that the RID function number is not compared – essentially all function numbers within the bus and device number range of this PE are also within this PE. Non-zero signifies to compare the RID function number to the PE function number, such that only that function number is in the PE domain, for all buses and device numbers in the PE domain.

pe_action is one of:

```
enum OpalPeAction {
    OPAL_UNMAP_PE = 0,
    OPAL_MAP_PE = 1
};
```

Return value:

OPAL_PARAMETER If one of the following: - invalid phb - invalid pe_action - invalid bus_dev_func - invalid bus_compare

OPAL_UNSUPPORTED PHB does not support set_pe operation

OPAL_SUCCESS if opreation was successful

4.3.36 OPAL_PCI_SET_PELTV

```
#define OPAL_PCI_SET_PELTV                      32
```

WARNING: This documentation comes from an old source and is possibly not up to date with OPALv3. Rely on this documentation only as a starting point, use the source (and update the docs).

```
static int64_t opal_pci_set_peltv(uint64_t phb_id, uint32_t parent_pe,
                                  uint32_t child_pe, uint8_t state)
```

This call sets the PELTV of a parent PE to add or remove a PE number as a PE within that parent PE domain. The host must call this function for each child of a parent PE.

phb_id is the value from the PHB node ibm,opal-phbid property

parent_pe is the PE number of a PE that is higher in the PCI hierarchy to other PEs, such that an error involving this parent PE should cause a collateral PE freeze for PEs below this PE in the PCI hierarchy. For example a switch upstream bridge is a PE that is parent to PEs reached through that upstream bridge such that an error involving the upstream bridge (e.g, ERR_FATAL) should cause the PHB to freeze all other PEs below that upstream bridge (e.g., a downstream bridge, or devices below a downstream bridge).

child_pe is the PE number of a PE that is lower in the PCI hierarchy than another PE, such that an error involving that other PE should cause a collateral PE freeze for this child PE. For example a device below a downstream bridge of a PCIE switch is a child PE that downstream bridge PE and the upstream bridge PE of that switch – an ERR_Fatal from either bridge should result in a collateral freeze of that device PE.

```
enum OpalPeltvAction {
     OPAL_REMOVE_PE_FROM_DOMAIN = 0,
```

```
    OPAL_ADD_PE_TO_DOMAIN = 1
};
```

OPAL Implementation Note: WARNING TODO: *CHECK IF THIS IS CORRECT FOR skiboot:* For ibm,opal-ioda2, OPAL sets the PELTV bit in all RTT entries for the parent PE when the state argument is '1'. OPAL clears the PELTV bit in all RTT entries for the parent PE when the state argument is '0' and setting the child PE bit in the parent PELTV results in an all-zeros value for that PELTV.

Return value:

```
if (!phb)
        return OPAL_PARAMETER;
if (!phb->ops->set_peltv)
        return OPAL_UNSUPPORTED;
```

4.3.37 OPAL_PCI_SET_PHB_MEM_WINDOW

```
#define OPAL_PCI_SET_PHB_MEM_WINDOW             28

static int64_t opal_pci_set_phb_mem_window(uint64_t phb_id,
                                           uint16_t window_type,
                                           uint16_t window_num,
                                           uint64_t addr,
                                           uint64_t pci_addr,
                                           uint64_t size)
```

WARNING: following documentation is from old sources, and is possibly not representative of OPALv3 as implemented by skiboot. This should be used as a starting point for full documentation.

The host calls this function to set the PHB PCI memory window parameters for PHBs. OPAL sets IO space for P7IOC and KVM cannot relocate this. KVM should changes these windows only while all devices below the PHB are disabled for PCI memory ops, and with the target window in disabled state (where supported by PHB hardware).

phb_id is the value from the PHB node ibm,opal-phbid property.

window_type specifies 32-bit or 64-bit PCI memory

'0' selects IO space, and is not supported for relocation. OPAL returns OPAL_UNSUPPORTED for this value.

'1' selects 32-bit PCI memory space

'2' selects 64 bit PCI memory space

window_num is the MMIO window number within the specified PCI memory space

starting_real_address specifies the location within sytsem (processor)real address space this MMIO window starts. This must be a location within the IO Hub or PHB node ibm,opal-mmio-real property.

starting_pci_address specifies the location within PCI 32 or 64-bit address space that this MMIO window starts. For 64-bit PCI memory, this must be within the low order 60 bit (1 Exabyte) region of PCI memory. Addresses above 1EB are reserved to IODA definitions.

segment_size defines the segment size of this window, in the same format as and a matching value from the ibm,opal-memwin32/64 <segment_size> property. The window total size, in bytes, is the segment_size times the ibm,opal-memwin32/64 <num_segments> property and must not extend beyond the ibm,opal-mmio-real property range within system real address space. The total MMIO window size is the segment_size times the num_segments supported for the specifice window. The host must assure that the cumulative address space for all enabled windows does not exceed the total PHB 32-bit or 64-bit real address window space, or extend

outside these address ranges, and that no windows overlap each other in real or PCI address space. OPAL does not validate those conditions.

A segment size of '0' indicates to disable this MMIO window. If the PHB hardware does not support disabling a window, OPAL returns OPAL_UNSUPPORTED status.

The size of the system real and PCI memory spaces are equal and defined by segment_size times the number of segments within this MMIO window.

The host must set PHB memory windows to be within the system real address ranges indicated in the PHB parent HDT hub node ibm,opal-mmio-real property.

Return value:

```
if (!phb)
        return OPAL_PARAMETER;
if (!phb->ops->set_phb_mem_window)
        return OPAL_UNSUPPORTED;
```

4.3.38 OPAL_PCI_SET_POWER_STATE

Set PCI slot power state

Parameters

uint64_t async_token Token of asynchronous message to be sent on completion of OPAL_PCI_SLOT_POWER_{OFF, ON}. It is ignored when @data is OPAL_PCI_SLOT_{OFFLINE, ONLINE}.

uint64_t id PCI slot ID

uint64_t data memory buffer pointer for the power state which can be one of OPAL_PCI_SLOT_POWER_{OFF, ON, OFFLINE, ONLINE}.

Calling

Set PCI slot's power state. The power state is stored in buffer pointed by @data. The typical use is to hot add or remove adapters behind the indicated PCI slot (by @id) in PCI hotplug path.

User will receive an asychronous message after calling the API. The message contains the API completion status: event (Power off or on), device node's phandle identifying the PCI slot, errcode (e.g. OPAL_SUCCESS). The API returns OPAL_ASYNC_COMPLETION for the case.

The states OPAL_PCI_SLOT_OFFLINE and OPAL_PCI_SLOT_ONLINE are used for removing or adding devices behind the slot. The device nodes in the device tree are removed or added accordingly, without actually changing the slot's power state. The API call will return OPAL_SUCCESS immediately and no further asynchronous message will be sent.

Return Codes

OPAL_SUCCESS PCI hotplug on the slot is completed successfully

OPAL_ASYNC_COMPLETION PCI hotplug needs further message to confirm

OPAL_PARAMETER The indicated PCI slot isn't found

OPAL_UNSUPPORTED Setting power state not supported on the PCI slot

4.3.39 OPAL_PCI_SET_XIVE_PE

```
static int64_t opal_pci_set_xive_pe(uint64_t phb_id, uint64_t pe_number,
                                    uint32_t xive_num)
```

WARNING: following documentation is from old sources, and is possibly not representative of OPALv3 as implemented by skiboot. This should be used as a starting point for full documentation.

The host calls this function to bind a PE to an XIVE. Only that PE may then signal an MSI that selects this XIVE.

phb_id is the value from the PHB node ibm,opal-phbid property.

pe_number is the index of a PE, from 0 to ibm,opal-num-pes minus 1.

xive_number is the index, from 0 to ibm,opal,ibm-num-msis minus (num_lsis+1)

This call maps the XIVR indexed by xive_num to the PE specified by pe_number. For ibm,opal-ioda HW, the pe_number must match the pe_number set in the MVE.

Return value:

```
if (!phb)
        return OPAL_PARAMETER;
if (!phb->ops->set_xive_pe)
        return OPAL_UNSUPPORTED;
```

4.3.40 OPAL_PCI_TCE_KILL

```
int64_t opal_pci_tce_kill(uint64_t phb_id,
                          uint32_t kill_type,
                          uint64_t pe_number,
                          uint32_t tce_size,
                          uint64_t dma_addr,
                          uint32_t npages)
```

An abstraction around TCE kill. This allows host OS kernels to use an OPAL call if they don't know the model specific invalidation method.

Where kill_type is one of:

```
enum {
   OPAL_PCI_TCE_KILL_PAGES,
   OPAL_PCI_TCE_KILL_PE,
   OPAL_PCI_TCE_KILL_ALL,
};
```

Not all PHB types currently support this abstraction. It is supported in PHB4, which means from POWER9 onwards it will be present.

Returns

OPAL_PARAMETER if phb_id is invalid (or similar)

OPAL_UNSUPPORTED if PHB model doesn't support this call. This is likely true for systems before POWER9/PHB4. Do *NOT* rely on this call existing for systems prior to POWER9 (i.e. PHB4).

Example code (from linux/arch/powerpc/platforms/powernv/pci-ioda.c)

```
static inline void pnv_pci_ioda2_tce_invalidate_pe(struct pnv_ioda_pe *pe)
{
      struct pnv_phb *phb = pe->phb;

      if (phb->model == PNV_PHB_MODEL_PHB3 && phb->regs)
         pnv_pci_phb3_tce_invalidate_pe(pe);
      else
         opal_pci_tce_kill(phb->opal_id, OPAL_PCI_TCE_KILL_PE,
                           pe->pe_number, 0, 0, 0);
}
```

and

```
struct pnv_phb *phb = pe->phb;
unsigned int shift = tbl->it_page_shift;
if (phb->model == PNV_PHB_MODEL_PHB3 && phb->regs)
      pnv_pci_phb3_tce_invalidate(pe, rm, shift,
                                  index, npages);
else
      opal_pci_tce_kill(phb->opal_id,
                        OPAL_PCI_TCE_KILL_PAGES,
                        pe->pe_number, 1u << shift,
                        index << shift, npages);
```

4.3.41 OPAL_POLL_EVENTS

Poll for outstanding events.

Fills in a bitmask of pending events.

Current events are:

OPAL_EVENT_OPAL_INTERNAL = 0x1

Currently unused.

OPAL_EVENT_NVRAM = 0x2

Unused

OPAL_EVENT_RTC = 0x4

TODO: clean this up, this is just copied from hw/fsp/fsp-rtc.c:

```
* Because the RTC calls can be pretty slow, these functions will shoot
* an asynchronous request to the FSP (if none is already pending)
*
* The requests will return OPAL_BUSY_EVENT as long as the event has
* not been completed.
*
* WARNING: An attempt at doing an RTC write while one is already pending
* will simply ignore the new arguments and continue returning
* OPAL_BUSY_EVENT. This is to be compatible with existing Linux code.
*
```

```
* Completion of the request will result in an event OPAL_EVENT_RTC
* being signaled, which will remain raised until a corresponding call
* to opal_rtc_read() or opal_rtc_write() finally returns OPAL_SUCCESS,
* at which point the operation is complete and the event cleared.
*
* If we end up taking longer than rtc_read_timeout_ms millieconds waiting
* for the response from a read request, we simply return a cached value (plus
* an offset calculated from the timebase. When the read request finally
* returns, we update our cache value accordingly.
*
* There is two separate set of state for reads and writes. If both are
* attempted at the same time, the event bit will remain set as long as either
* of the two has a pending event to signal.
```

OPAL_EVENT_CONSOLE_OUTPUT = 0x8

TODO

OPAL_EVENT_CONSOLE_INPUT = 0x10

TODO

OPAL_EVENT_ERROR_LOG_AVAIL = 0x20

TODO

OPAL_EVENT_ERROR_LOG = 0x40

TODO

OPAL_EVENT_EPOW = 0x80

TODO

OPAL_EVENT_LED_STATUS = 0x100

TODO

OPAL_EVENT_PCI_ERROR = 0x200

TODO

OPAL_EVENT_DUMP_AVAIL = 0x400

Signifies that there is a pending system dump available. See OPAL_DUMP suite of calls for details.

```
OPAL_EVENT_MSG_PENDING = 0x800,
```

4.3.42 OPAL_PRD_MSG

The OPAL_PRD_MSG call is used to pass a struct opal_prd_msg from the HBRT code into opal, and is paired with the OPAL_PRD_MSG message type.

Parameters

`struct opal_msg *msg` Passes an opal_msg, of type OPAL_PRD_MSG, from the OS to OPAL.

4.3.43 OPAL_READ_TPO and OPAL_WRITE_TPO

TPO is a Timed Power On facility.

It is an OPTIONAL part of the OPAL spec.

If a platform supports Timed Power On (TPO), the RTC node in the device tree (itself under the "ibm,opal" node will have the has-tpo property:

```
rtc {
   compatible = "ibm,opal-rtc";
   has-tpo;
};
```

If the "has-tpo" proprety is *NOT* present then OPAL does *NOT* support TPO.

4.3.44 OPAL_REGISTER_DUMP_REGION

This call is used to register regions of memory for a service processor to capture when the host crashes.

e.g. if an assert is hit in OPAL, a service processor will copy

This is an OPTIONAL feature that may be unsupported, the host OS should use an OPAL_CHECK_TOKEN call to find out if OPAL_REGISTER_DUMP_REGION is supported.

OPAL_REGISTER_DUMP_REGION accepts 3 parameters:

- region ID
- address
- length

There is a range of region IDs that can be used by the host OS. A host OS should start from OPAL_DUMP_REGION_HOST_END and work down if it wants to add a not well defined region to dump. Currently the only well defined region is for the host OS log buffer (e.g. dmesg on linux).

```
/*
 * Dump region ID range usable by the OS
 */
 #define OPAL_DUMP_REGION_HOST_START          0x80
 #define OPAL_DUMP_REGION_LOG_BUF             0x80
 #define OPAL_DUMP_REGION_HOST_END            0xFF
```

OPAL_REGISTER_DUMP_REGION will return OPAL_UNSUPPORTED if the call is present but the system doesn't support registering regions to be dumped.

In the event of being passed an invalid region ID, OPAL_REGISTER_DUMP_REGION will return OPAL_PARAMETER.

Systems likely have a limit as to how many regions they can support being dumped. If this limit is reached, OPAL_REGISTER_DUMP_REGION will return OPAL_INTERNAL_ERROR.

BUGS

Some skiboot versions incorrectly returned OPAL_SUCCESS in the case of OPAL_REGISTER_DUMP_REGION being supported on a platform (so the call was present) but the call being unsupported for some reason (e.g. on an IBM POWER7 machine).

See also: OPAL_UNREGISTER_DUMP_REGION

4.3.45 OPAL_REINIT_CPUS

```
static int64_t opal_reinit_cpus(uint64_t flags);
```

This OPAL call reinitializes some bit of CPU state across *ALL* CPUs. Consequently, all CPUs must be in OPAL for this call to succeed (either at boot time or after OPAL_RETURN_CPU is called)

Arguments

Currently, possible flags are:

```
enum {
      OPAL_REINIT_CPUS_HILE_BE         = (1 << 0),
      OPAL_REINIT_CPUS_HILE_LE         = (1 << 1),
};
```

Extra flags may be added in the future, so other bits *must* be 0.

On POWER7 CPUs, only OPAL_REINIT_CPUS_HILE_BE is supported. All other flags will return OPAL_UNSUPPORTED.

On POWER8 CPUs, only OPAL_REINIT_CPUS_HILE_BE and OPAL_REINIT_CPUS_HILE_LE are support and other bits *MUST NOT* be set.

On POWER9 CPUs, other flags may be supported in the future.

Returns

OPAL_SUCCESS Success!

OPAL_UNSUPPORTED Processor does not suport reinit flags.

4.3.46 OPAL_RETURN_CPU

```
int64_t opal_return_cpu(void);
```

When OPAL first starts the host, all secondary CPUs are spinning in OPAL. To start them, one must call OPAL_START_CPU (you may want to OPAL_REINIT_CPU to set the HILE bit first).

In cases where you need OPAL to do something for you across all CPUs, such as OPAL_REINIT_CPU, (on some platforms) a firmware update or get the machine back into a similar state as to when the host OS was started (e.g. for kexec) you may also need to return control of the CPU to OPAL.

Returns

This call does **not return**. You need to OPAL_START_CPU.

4.3.47 OPAL_RTC_READ

Read the Real Time Clock.

Parameters

uint32_t* year_month_day the year, month and day formatted as follows:

- bits 0-15 is bcd formatted year (0100-9999)
- bits 16-23 is bcd formatted month (01-12)
- bits 24-31 is bcd formatted day (01-31)

uint64_t* hour_minute_second_millisecond the hour, minute, second and millisecond formatted as follows:

- bits 0-16 is reserved
- bits 17-24 is bcd formatted hour (00-23)
- bits 25-31 is bcd formatted minute (00-59)
- bits 32-39 is bcd formatted second (00-60)
- bits 40-63 is bcd formatted milliseconds (000000-999999)

Calling

Since RTC calls can be pretty slow, OPAL_RTC_READ is likely to first return OPAL_BUSY_EVENT, requiring the caller to wait until the OPAL_EVENT_RTC event has been signaled. Once the event has been signaled, a subsequent OPAL_RTC_READ call will retrieve the time. Since the OPAL_EVENT_RTC event is used for both reading and writing the RTC, callers must be able to handle the event being signaled for a concurrent in flight OPAL_RTC_WRITE rather than this read request.

The following code is one way to correctly issue and then wait for a response:

```
int rc = OPAL_BUSY_EVENT;
while (rc == OPAL_BUSY_EVENT) {
      rc = opal_rtc_read(&y_m_d, &h_m_s_ms);
      if (rc == OPAL_BUSY_EVENT)
         opal_poll_events(NULL);
}
```

Although as of writing all OPAL_RTC_READ backends are asynchronous, there is no requirement for them to be - it is valid for OPAL_RTC_READ to immediately return the retreived value rather than OPAL_BUSY_EVENT.

TODO: describe/document format of arguments.

Return codes

OPAL_SUCCESS parameters now contain the current time, or one read from cache.

OPAL_HARDWARE error in retrieving the time. May be transient error, may be permanent.

OPAL_PARAMETER year_month_day or hour_minute_second_millisecond parameters are NULL

OPAL_INTERNAL_ERROR something went wrong, Possibly reported in error log.

OPAL_BUSY_EVENT request is in flight

4.3.48 OPAL_RTC_WRITE

OPAL_RTC_WRITE is much like OPAL_RTC_READ in that it can be asynchronous.

If multiple WRITES are issued before the first one completes, subsequent writes are ignored. There can only be one write in flight at any one time.

Format of the time is the same as for OPAL_RTC_READ.

4.3.49 OPAL_SENSOR_READ

The OPAL sensor call reads a sensor data using a unique handler to identity the targeted sensor.

This call can be asynchronous, when a message needs to be sent to a service processor for example. In this case, the call will return OPAL_ASYNC_COMPLETION and the token parameter will be used to wait for the completion of the request.

Parameters

```
uint32_t sensor_handler
int      token
uint32_t *sensor_data
```

Return values

OPAL_SUCCESS Success!

OPAL_PARAMETER invalid sensor handler

OPAL_UNSUPPORTED platform does not support reading sensors.

In case of communication with the FSP on IBM systems:

OPAL_ASYNC_COMPLETION a request was sent and an async completion will be triggered with the @token argument

OPAL_PARTIAL the request completed but the data returned is invalid

OPAL_BUSY_EVENT a previous request is still pending

OPAL_NO_MEM allocation failed

OPAL_INTERNAL_ERROR communication failure with the FSP

OPAL_HARDWARE FSP is not available

4.3.50 OPAL_SET_XIVE

```
#define OPAL_SET_XIVE                                   19
```

WARNING: following documentation is from old sources, and is possibly not representative of OPALv3 as implemented by skiboot. This should be used as a starting point for full documentation.

The host calls this function to set the POWER XIVE server and priority parameters into the PHB XIVE.

Parameters

phb_id is the value from the PHB node ibm,opal-phbid property.

xive_number is the index of an XIVE that corresponds to a particular interrupt

service_number is the server (processor) that is to receive the interrupt request

priority is the interrupt priority value applied to the interrupt (0=highest, 0xFF = lowest/disabled).

4.3.51 OPAL_SYNC_HOST_REBOOT

```
static int64_t opal_sync_host_reboot(void)
```

This OPAL call halts asynchronous operations in preparation for something like kexec. It will halt DMA as well notification of some events (such as a new error log being available for retreival).

It's meant to be called in a loop until OPAL_SUCCESS is returned.

Returns

OPAL_SUCCESS Success!

OPAL_BUSY_EVENT not yet complete, call opal_sync_host_reboot() again, possibly with a short delay.

OPAL_BUSY Call opal_poll_events() and then retry opal_sync_host_reboot

4.3.52 OPAL_TEST

OPAL_TEST is a REQUIRED call for OPAL and conforming implementations MUST have it.

It is designed to test basic OPAL call functionality.

Token:

```
#define OPAL_TEST                                       0
```

Arguments

```
uint64_t     arg
```

Returns

```
0xfeedf00d
```

Function

OPAL_TEST MAY print a string to the OPAL log with the value of argument.

For example, the reference implementation (skiboot) implements OPAL_TEST as:

```
static uint64_t opal_test_func(uint64_t arg)
{
      printf("OPAL: Test function called with arg 0x%llx\n", arg);

      return 0xfeedf00d;
}
```

4.3.53 OPAL_UNREGISTER_DUMP_REGION

While OPAL_REGISTER_DUMP_REGION registers a region, OPAL_UNREGISTER_DUMP_REGION will unregister a region by region ID.

OPAL_UNREGISTER_DUMP_REGION takes one argument: the region ID.

A host OS should check OPAL_UNREGISTER_DUMP_REGION is supported through a call to OPAL_CHECK_TOKEN.

If OPAL_UNREGISTER_DUMP_REGION is called on a system where the call is present but unsupported, it will return OPAL_UNSUPPORTED.

BUGS

Some skiboot versions incorrectly returned OPAL_SUCCESS in the case of OPAL_UNREGISTER_DUMP_REGION being supported on a platform (so the call was present) but the call being unsupported for some reason (e.g. on an IBM POWER7 machine).

4.3.54 OPAL_XSCOM_READ and OPAL_XSCOM_WRITE

These low level calls will read/write XSCOM values directly.

They should only be used by low level manufacturing/debug tools. "Normal" host OS kernel code should not know about XSCOM.

each takes three parameters:

```
int xscom_read(uint32_t partid, uint64_t pcb_addr, uint64_t *val)
int xscom_write(uint32_t partid, uint64_t pcb_addr, uint64_t val)
```

Returns

OPAL_SUCCESS Success!

OPAL_HARDWARE if operation failed

OPAL_WRONG_STATE if CPU is asleep

4.3.55 POWER9 Changes to OPAL API

This document is a summary of POWER9 changes to the OPAL API over what it was for POWER7 and POWER8. As the POWER series of processors (at least up to POWER9) require changes in the hypervisor to work on a new processor generation, this gives us an opportunity with POWER9 to clean up several parts of the OPAL API.

Eventually, when the kernel drops support for POWER8 and before, we can then remove the associated kernel code too.

OPAL_REINIT_CPUS

Can now be extended beyond HILE BE/LE bits. If invalid flags are set on POWER9, OPAL_UNSUPPORTED will be returned.

Device Tree

- `/ibm,opal/` compatible property now just lists `ibm,opal-v3` and no longer `ibm,opal-v2` (power9 and above only)

TODO

Things we still have to do for POWER9:

- PCI to use async API rather than returning delays
- deprecate/remove v1 APIs where there's a V2
- Fix this FWTS warning:

```
FAILED [MEDIUM] DeviceTreeBaseDTCWarnings: Test 3, dtc reports warnings from
device tree: Warning (reg_format): "reg" property in /ibm,opal/flash@0 has
invalid length (8 bytes) (#address-cells == 0, #size-cells == 0)
```

- Remove mi-version / ml-version from `/ibm,opal/firmware` and replace with something better and more portable

4.3.56 OPAL API Return Codes

All OPAL calls return an integer relaying the success/failure of the OPAL call.

Success is typically indicated by OPAL_SUCCESS. Failure is always indicated by a negative return code.

Conforming host Operating Systems MUST handle return codes other than those listed here. In future OPAL versions, additional return codes may be added.

In the reference implementation (skiboot) these are all in include/opal.h.

The core set of return codes are:

OPAL_SUCCESS

```
#define OPAL_SUCCESS            0
```

Success!

OPAL_PARAMETER

```
#define OPAL_PARAMETER          -1
```

A parameter was invalid. This will also be returned if you call an invalid OPAL call. To determine if a specific OPAL call is supported or not, OPAL_CHECK_TOKEN should be called rather than relying on OPAL_PARAMETER being returned for an invalid token.

OPAL_BUSY

```
#define OPAL_BUSY               -2
```

Try again later.

OPAL_PARTIAL

```
#define OPAL_PARTIAL            -3
```

The operation partially succeeded.

OPAL_CONSTRAINED

```
#define OPAL_CONSTRAINED        -4
```

FIXME

OPAL_CLOSED

```
#define OPAL_CLOSED             -5
```

FIXME document these

OPAL_HARDWARE

```
#define OPAL_HARDWARE                   -6
```

FIXME document these

OPAL_UNSUPPORTED

```
#define OPAL_UNSUPPORTED      -7
```

Unsupported operation. Non-fatal.

OPAL_PERMISSION

```
#define OPAL_PERMISSION               -8
```

Inadequate permission to perform the operation.

OPAL_NO_MEM

```
#define OPAL_NO_MEM           -9
```

Indicates a temporary or permanent lack of adequate memory to perform the operation. Ideally, this should never happen. Skiboot reserves a small amount of memory for its heap and some operations (such as I2C requests) are allocated from this heap.

If this is ever hit, you should likely file a bug.

OPAL_RESOURCE

```
#define OPAL_RESOURCE                 -10
```

FIXME

OPAL_INTERNAL_ERROR

```
#define OPAL_INTERNAL_ERROR   -11
```

FIXME

OPAL_BUSY_EVENT

```
#define OPAL_BUSY_EVENT               -12
```

OPAL_HARDWARE_FROZEN

```
#define OPAL_HARDWARE_FROZEN -13
```

OPAL_WRONG_STATE

```
#define OPAL_WRONG_STATE      -14
```

OPAL_ASYNC_COMPLETION

```
#define OPAL_ASYNC_COMPLETION        -15
```

For asynchronous calls, successfully queueing/starting executing the command is indicated by the OPAL_ASYNC_COMPLETION return code. pseudo-code for an async call:

```
token = opal_async_get_token();
rc = opal_async_example(foo, token);
if (rc != OPAL_ASYNC_COMPLETION)
    handle_error(rc);
rc = opal_async_wait(token);
// handle result here
```

OPAL_EMPTY

```
#define OPAL_EMPTY           -16
```

I2C Calls

Added for I2C, only applicable to I2C calls:

```
#define OPAL_I2C_TIMEOUT      -17
#define OPAL_I2C_INVALID_CMD  -18
#define OPAL_I2C_LBUS_PARITY  -19
#define OPAL_I2C_BKEND_OVERRUN        -20
#define OPAL_I2C_BKEND_ACCESS -21
#define OPAL_I2C_ARBT_LOST    -22
#define OPAL_I2C_NACK_RCVD    -23
#define OPAL_I2C_STOP_ERR     -24
```

CHAPTER

FIVE

SKIBOOT RELEASE NOTES

5.1 Release Notes

5.1.1 skiboot-5.1.0

skiboot-5.1.0 was released on August 17th, 2015.

skiboot-5.1.0 is the first stable release of 5.1.0 following two beta releases. This new stable release replaces skiboot-5.0 as the current stable skiboot release (5.0 was released April 14th 2015).

Skiboot 5.1.0 contains all fixes from skiboot-5.0 stable branch up to skiboot-5.0.5 and everything from 5.1.0-beta1 and 5.1.0-beta2.

Over skiboot-5.1.0-beta2, we have the following changes:

- opal_prd now supports multiple socket systems
- fix compiler warnings in gard and libflash

Below are the changes introduced in previous skiboot-5.1.0 releases over the previous stable release, skiboot-5.0:

New features

- Add Naples chip (CPU, PHB, LPC serial interrupts) support
- Added qemu platform
- improvements to FSI error handling
- improvements in chip TOD failover (some only on FSP systems)
- Set Relative Priority Register (RPR) to recommended value
  - this affects thread priority in SMT modes
- greatly reduce memory consumption by CPU stacks for non-present CPUs
  - Previously we would reserve enough memory for max PIR for each CPU type.
  - This fix frees up 77MB of RAM on a typical P8 system.
- increased OPAL API documentation
- Asynchronous preloading of resources from FSP/flash
  - improves boot time on some systems
- Basic Garrison platform support
- Add Mambo platform (P8 Functional Simulator, systemsim)

  - includes fake NVRAM, RTC
- Support building with GCOV, increasing memory for skiboot binary to 2MB
  - includes boot code coverage testing
- Increased skiboot HEAP size.
  - We are not aware of any system where you would run out, but on large systems it was getting closer than we liked.
- add boot_tests.sh for helping automate boot testing on FSP and BMC machines
- Versioning of pflash and gard utilities to help Linux (or other OS) distributions with packaging.
- OCC throttle status messages to host
- CAPP timebase sync ("ibm,capp-timebase-sync" in DT to indicate CAPP timebase was synced by OPAL)
- opal-api: Add OPAL call to handle abnormal reboots.

`OPAL_CEC_REBOOT2` currently supports two reboot types:

0. normal reboot, that will behave similar to that of opal_cec_reboot() call
1. platform error reboot.

Long term, this is designed to replace OPAL_CEC_REBOOT.

New features for FSP based machines

- in-band IPMI support
- ethernet adaptor location codes
- add DIMM frequency information to device tree
- improvements in FSP error log code paths
- fix some boot time memory leaks
  - harmless to end user

New features for AMI BMC based machines

- PCIe power workaround for K80
- Added support for Macronix 128Mbit flash chips
- Initial PRD support for Firestone platform
- improved reliability when BMC reboots

The following bugs have been fixed

- Increase PHB3 timeout for electrical links coming up to 2 seconds.
  - fixes issues with some Mellanox cards
- Hang in opal_reinit_cpus() that could prevent kdump from functioning
- PHB3: fix crash in phb3_init
- PHB3: fix crash with fenced PHB in phb3_init_hw()

- Fix bugs in hw/bt.c (interface for IPMI on BMC machines) that could possibly lead to a crash (dereferencing invalid address, deadlock)
- ipmi/sel: fix use-after-free
- Bug fixes in EEH handling
  - opal_pci_next_error() cleared OPAL_EVENT_PCI_ERROR unconditionally, possibly leading to missed errors.
- external/opal-prd: Only map each PRD range once
  - could eventually lead to failing to map PRD ranges
- On skiboot crash, don't try to print symbol when we didn't find one
  - makes backtrace prettier
- On skiboot crash, dump hssr0 and hsrr1 registers correctly.
- Better support old and biarch compilers
  - test "new" compiler flags before using them
  - Specify -mabi=elfv1 if supported (which means it's needed)
- fix boot-coverage-report makefile target
- ipmi: Fix the opal_ipmi_recv() call to handle the error path
- Could make kernel a sad panda when in continues with other IPMI commands
- IPMI: truncate SELs at 2kb
  - it's the limit of the astbmc. We think.
- IPMI/SEL/PEL:
  - As per PEL spec, we should log events with severity >= 0x22 and "service action flag" is "on". But in our case, all logs OPAL originagted logs are makred as report externally. We now only report logs with severity >= 0x22
- IPMI: fixes to eSEL logging
- hw/phb3: Change reserved PE to 255
  - Currently, we have reserved PE#0 to which all RIDs are mapped prior to PE assignment request from kernel. The last M64 BAR is configured to have shared mode. So we have to cut off the first M64 segment, which corresponds to reserved PE#0 in kernel. If the first BAR (for example PF's IOV BAR) requires huge alignment in kernel, we have to waste huge M64 space to accommodate the alignment. If we have reserved PE#256, the waste of M64 space will be avoided.

FSP-specific bugs fixed

- (also fixed in skiboot-5.0.2) Fix race in firenze_get_slot_info() leading to assert() with many PCI cards

  With many PCI cards, we'd hit a race where calls to firenze_add_pcidev_to_fsp_inventory would step on each other leading to memory corruption and finally an assert() in the allocator being hit during boot.
- PCIe power workaround for K80 cards
- /ibm,opal/led renamed to /ibm,opal/leds in Device Tree
  - compatible change as no FSP based systems shipped with skiboot-5.0

General improvements

- Preliminary Centaur i2c support
  - lays framework for supporting Centaur i2c
- don't run pollers on non-boot CPUs in time_wait
- improvements to opal-prd, pflash, libflash
  - including new blocklevel interface in libflash
- many minor fixes to issues found by static analysis
- improvements in FSP error log code paths
- code cleanup in memory allocator
- Don't expose individual nvram partitions in the device tree, just the whole flash device.
- build improvements for building on ppc64el host
- improvements in cpu_relax() for idle threads, needed for GCOV on large machines.
- Optimized memset() for POWER8, greatly reducing number of instructions executed for boot, which helps boot time in simulators.
- Major improvements in hello_world kernel
  - Bloat of huge 17 instruction test case reduced to 10.
- Disable bust_locks for general calls of abort()
  - Should enable better error messages during abort() when other users of LPC bus exist (e.g. flash)
- unified version numbers for bundled utilities
- external/boot_test/boot_test.sh
  - better usable for automated boot testing

Contributors

Since skiboot-5.0, we've had the following changesets:

Processed 372 csets from 27 developers 2 employers found A total of 15868 lines added, 3359 removed (delta 12509)

Developers with the most changesets

Developer	Changesets
Stewart Smith	117 (31.5%)
Jeremy Kerr	37 (9.9%)
Cyril Bur	33 (8.9%)
Vasant Hegde	32 (8.6%)
Benjamin Herrenschmidt	32 (8.6%)
Kamalesh Babulal	22 (5.9%)
Joel Stanley	12 (3.2%)
Mahesh Salgaonkar	12 (3.2%)
Alistair Popple	12 (3.2%)
Neelesh Gupta	9 (2.4%)
Gavin Shan	8 (2.2%)
Cédric Le Goater	8 (2.2%)
Ananth N Mavinakayanahalli	8 (2.2%)
Vipin K Parashar	6 (1.6%)
Michael Neuling	6 (1.6%)
Samuel Mendoza-Jonas	3 (0.8%)
Frederic Bonnard	3 (0.8%)
Andrew Donnellan	2 (0.5%)
Vaidyanathan Srinivasan	2 (0.5%)
Philippe Bergheaud	1 (0.3%)
Shilpasri G Bhat	1 (0.3%)
Daniel Axtens	1 (0.3%)
Hari Bathini	1 (0.3%)
Michael Ellerman	1 (0.3%)
Andrei Warkentin	1 (0.3%)
Dan Horák	1 (0.3%)
Anton Blanchard	1 (0.3%)

Developers with the most changed lines

Stewart Smith	4499 (27.3%)
Benjamin Herrenschmidt	3782 (22.9%)
Jeremy Kerr	1887 (11.4%)
Cyril Bur	1654 (10.0%)
Vasant Hegde	959 (5.8%)
Mahesh Salgaonkar	886 (5.4%)
Neelesh Gupta	473 (2.9%)
Samuel Mendoza-Jonas	387 (2.3%)
Vipin K Parashar	332 (2.0%)
Philippe Bergheaud	171 (1.0%)
Shilpasri G Bhat	165 (1.0%)
Alistair Popple	151 (0.9%)
Joel Stanley	105 (0.6%)
Cédric Le Goater	89 (0.5%)
Gavin Shan	83 (0.5%)
Frederic Bonnard	76 (0.5%)
Kamalesh Babulal	65 (0.4%)
Michael Neuling	46 (0.3%)
Daniel Axtens	31 (0.2%)
Andrew Donnellan	22 (0.1%)
Ananth N Mavinakayanahalli	20 (0.1%)
Anton Blanchard	3 (0.0%)
Vaidyanathan Srinivasan	2 (0.0%)
Hari Bathini	2 (0.0%)
Michael Ellerman	1 (0.0%)
Andrei Warkentin	1 (0.0%)
Dan Horák	1 (0.0%)

Developers with the most lines removed

Michael Neuling	24 (0.7%)
Hari Bathini	1 (0.0%)

Developers with the most signoffs (total 253)

Stewart Smith	249 (98.4%)
Mahesh Salgaonkar	4 (1.6%)

Developers with the most reviews (total 24)

Vasant Hegde	9 (37.5%)
Joel Stanley	3 (12.5%)
Gavin Shan	2 (8.3%)
Kamalesh Babulal	2 (8.3%)
Samuel Mendoza-Jonas	2 (8.3%)
Alistair Popple	2 (8.3%)
Stewart Smith	1 (4.2%)
Andrei Warkentin	1 (4.2%)
Preeti U Murthy	1 (4.2%)
Ananth N Mavinakayanahalli	1 (4.2%)

Developers with the most test credits (total 1)

Chad Larson	1 (100.0%)

Developers who gave the most tested-by credits (total 1)

Gavin Shan	1 (100.0%)

Developers with the most report credits (total 4)

Benjamin Herrenschmidt	2 (50.0%)
Chad Larson	1 (25.0%)
Andrei Warkentin	1 (25.0%)

Developers who gave the most report credits (total 4)

Stewart Smith	3 (75.0%)
Gavin Shan	1 (25.0%)

Top changeset contributors by employer

IBM	369 (99.2%)
(Unknown)	3 (0.8%)

Top lines changed by employer

IBM	16497 (100.0%)
(Unknown)	3 (0.0%)

Employers with the most signoffs (total 253)

IBM	253 (100.0%)

Employers with the most hackers (total 27)

IBM	24 (88.9%)
(Unknown)	3 (11.1%)

5.1.2 skiboot-5.1.0-beta1

skiboot-5.1.0-beta1 was released on July 21st, 2015.

skiboot-5.1.0-beta1 is the first beta release of skiboot 5.1, which will become a new stable release, replacing skiboot-5.0 (released April 14th 2015)

Skiboot 5.1-beta1 contains all fixes from skiboot-5.0 stable branch up to skiboot-5.0.5.

New features

Over skiboot-5.0, the following features have been added:

- Centaur i2c support
- Add Naples chip (CPU, PHB, LPC serial interrupts) support
- Added qemu platform

- improvements to FSI error handling
- improvements in chip TOD failover (some only on FSP systems)
- Set Relative Priority Register (RPR) to recommended value
  - this affects thread priority in SMT modes
- greatly reduce memory consumption by CPU stacks for non-present CPUs
  - Previously we would reserve enough memory for max PIR for each CPU type.
  - This fix frees up 77MB of RAM on a typical P8 system.
- increased OPAL API documentation
- Asynchronous preloading of resources from FSP/flash
  - improves boot time on some systems
- Basic Garrison platform support
- Add Mambo platform (P8 Functional Simulator, systemsim)
  - includes fake NVRAM, RTC
- Support building with GCOV, increasing memory for skiboot binary to 2MB
  - includes boot code coverage testing
- Increased skiboot HEAP size.
  - We are not aware of any system where you would run out, but on large systems it was getting closer than we liked.
- add boot_tests.sh for helping automate boot testing on FSP and BMC machines
- Versioning of pflash and gard utilities to help Linux (or other OS) distributions with packaging.
- OCC throttle status messages to host
- CAPP timebase sync ("ibm,capp-timebase-sync" in DT to indicate CAPP timebase was synced by OPAL)

New features for FSP based machines

- in-band IPMI support
- ethernet adaptor location codes
- add DIMM frequency information to device tree
- improvements in FSP error log code paths
- fix some boot time memory leaks
  - harmless to end user

New features for AMI BMC based machines

- PCIe power workaround for K80
- Added support for Macronix 128Mbit flash chips
- Initial PRD support for Firestone platform
- improved reliability when BMC reboots

Bug Fixes

The following bugs have been fixed:

- Increase PHB3 timeout for electrical links coming up to 2 seconds.
  - fixes issues with some Mellanox cards
- Hang in opal_reinit_cpus() that could prevent kdump from functioning
- PHB3: fix crash in phb3_init
- PHB3: fix crash with fenced PHB in phb3_init_hw()
- Fix bugs in hw/bt.c (interface for IPMI on BMC machines) that could possibly lead to a crash (dereferencing invalid address, deadlock)
- ipmi/sel: fix use-after-free
- Bug fixes in EEH handling
  - opal_pci_next_error() cleared OPAL_EVENT_PCI_ERROR unconditionally, possibly leading to missed errors.

FSP-specific bugs fixed:

- (also fixed in skiboot-5.0.2) Fix race in firenze_get_slot_info() leading to assert() with many PCI cards

  With many PCI cards, we'd hit a race where calls to firenze_add_pcidev_to_fsp_inventory would step on each other leading to memory corruption and finally an assert() in the allocator being hit during boot.
- PCIe power workaround for K80 cards
- /ibm,opal/led renamed to /ibm,opal/leds in Device Tree
  - compatible change as no FSP based systems shipped with skiboot-5.0

General improvements:

- don't run pollers on non-boot CPUs in time_wait
- improvements to opal-prd, pflash, libflash
  - including new blocklevel interface in libflash
- many minor fixes to issues found by static analysis
- improvements in FSP error log code paths
- code cleanup in memory allocator
- Don't expose individual nvram partitions in the device tree, just the whole flash device.
- build improvements for building on ppc64el host
- improvements in cpu_relax() for idle threads, needed for GCOV on large machines.
- Optimized memset() for POWER8, greatly reducing number of instructions executed for boot, which helps boot time in simulators.
- Major improvements in hello_world kernel
  - Bloat of huge 17 instruction test case reduced to 10.
- Disable bust_locks for general calls of abort()

- Should enable better error messages during abort() when other users of LPC bus exist (e.g. flash)

Contributors

Thanks to everyone who has made skiboot-5.1.0-beta1 happen!

Processed 321 csets from 25 developers 3 employers found A total of 13696 lines added, 2754 removed (delta 10942)

Developers with the most changesets

Developer	Changesets
Stewart Smith	101 (31.5%)
Benjamin Herrenschmidt	32 (10.0%)
Cyril Bur	31 (9.7%)
Vasant Hegde	28 (8.7%)
Jeremy Kerr	27 (8.4%)
Kamalesh Babulal	19 (5.9%)
Alistair Popple	12 (3.7%)
Mahesh Salgaonkar	12 (3.7%)
Neelesh Gupta	8 (2.5%)
Cédric Le Goater	8 (2.5%)
Joel Stanley	8 (2.5%)
Ananth N Mavinakayanahalli	8 (2.5%)
Gavin Shan	6 (1.9%)
Michael Neuling	6 (1.9%)
Frederic Bonnard	3 (0.9%)
Vipin K Parashar	2 (0.6%)
Vaidyanathan Srinivasan	2 (0.6%)
Philippe Bergheaud	1 (0.3%)
Shilpasri G Bhat	1 (0.3%)
Daniel Axtens	1 (0.3%)
Hari Bathini	1 (0.3%)
Michael Ellerman	1 (0.3%)
Andrei Warkentin	1 (0.3%)
Dan Horák	1 (0.3%)
Anton Blanchard	1 (0.3%)

Developers with the most changed lines

Developer	Changed Lines
Stewart Smith	3987 (27.9%)
Benjamin Herrenschmidt	3811 (26.6%)
Cyril Bur	1918 (13.4%)
Jeremy Kerr	1307 (9.1%)
Mahesh Salgaonkar	886 (6.2%)
Vasant Hegde	764 (5.3%)
Neelesh Gupta	473 (3.3%)
Vipin K Parashar	176 (1.2%)
Alistair Popple	175 (1.2%)
Philippe Bergheaud	171 (1.2%)
Shilpasri G Bhat	165 (1.2%)
Cédric Le Goater	89 (0.6%)
Frederic Bonnard	78 (0.5%)
Gavin Shan	73 (0.5%)
Joel Stanley	65 (0.5%)
Kamalesh Babulal	63 (0.4%)
Michael Neuling	47 (0.3%)
Daniel Axtens	31 (0.2%)
Ananth N Mavinakayanahalli	22 (0.2%)
Anton Blanchard	3 (0.0%)
Vaidyanathan Srinivasan	2 (0.0%)
Hari Bathini	2 (0.0%)
Michael Ellerman	1 (0.0%)
Andrei Warkentin	1 (0.0%)
Dan Horák	1 (0.0%)

Developers with the most lines removed:

Vipin K Parashar	105 (3.8%)
Michael Neuling	24 (0.9%)
Hari Bathini	1 (0.0%)

Developers with the most signoffs (total 214)

Stewart Smith	214 (100.0%)

Developers with the most reviews (total 21)

Vasant Hegde	7 (33.3%)
Joel Stanley	3 (14.3%)
Gavin Shan	2 (9.5%)
Kamalesh Babulal	2 (9.5%)
Alistair Popple	2 (9.5%)
Stewart Smith	1 (4.8%)
Andrei Warkentin	1 (4.8%)
Preeti U Murthy	1 (4.8%)
Samuel Mendoza-Jonas	1 (4.8%)
Ananth N Mavinakayanahalli	1 (4.8%)

Developers with the most test credits (total 1)

Chad Larson	1 (100.0%)

Developers who gave the most tested-by credits (total 1)

Gavin Shan	1 (100.0%)

Developers with the most report credits (total 4)

Benjamin Herrenschmidt	2 (50.0%)
Chad Larson	1 (25.0%)
Andrei Warkentin	1 (25.0%)

Developers who gave the most report credits (total 4)

Stewart Smith	3 (75.0%)
Gavin Shan	1 (25.0%)

Top changeset contributors by employer

IBM	319 (99.4%)
dan@danny.cz	1 (0.3%)
andrey.warkentin@gmail.com	1 (0.3%)

Top lines changed by employer

IBM	14309 (100.0%)
dan@danny.cz	1 (0.0%)
andrey.warkentin@gmail.com	1 (0.0%)

Employers with the most signoffs (total 214)

IBM	214 (100.0%)

Employers with the most hackers (total 25)

IBM	23 (92.0%)
dan@danny.cz	1 (4.0%)
andrey.warkentin@gmail.com	1 (4.0%)

5.1.3 skiboot-5.1.0-beta2

skiboot-5.1.0-beta2 was released on August 14th, 2015.

skiboot-5.1.0-beta2 is the second beta release of skiboot 5.1, which will become a new stable release, replacing skiboot-5.0 (released April 14th 2015)

Skiboot 5.1.0-beta2 contains all fixes from skiboot-5.0 stable branch up to skiboot-5.0.5 and everything from 5.1.0-beta1.

New Features

Over skiboot-5.1.0-beta1, the following features have been added:

- **opal-api: Add OPAL call to handle abnormal reboots.** OPAL_CEC_REBOOT2 Currently it will support two reboot types (0). normal reboot, that will behave similar to that of opal_cec_reboot() call, and (1). platform error reboot.

Long term, this is designed to replace OPAL_CEC_REBOOT.

Bug fixes

Over skiboot-5.1.0-beta1, the following bugs have been fixed:

- external/opal-prd: Only map each PRD range once
  - could eventually lead to failing to map PRD ranges
- On skiboot crash, don't try to print symbol when we didn't find one
  - makes backtrace prettier
- On skiboot crash, dump hssr0 and hsrr1 registers correctly.
- Better support old and biarch compilers
  - test "new" compiler flags before using them
  - Specify -mabi=elfv1 if supported (which means it's needed)
- fix boot-coverage-report makefile target
- ipmi: Fix the opal_ipmi_recv() call to handle the error path
  - Could make kernel a sad panda when in continues with other IPMI commands
- IPMI: truncate SELs at 2kb
  - it's the limit of the astbmc. We think.
- IPMI/SEL/PEL:
  - As per PEL spec, we should log events with severity >= 0x22 and "service action flag" is "on". But in our case, all logs OPAL originagted logs are makred as report externally. We now only report logs with severity >= 0x22
- IPMI: fixes to eSEL logging
- hw/phb3: Change reserved PE to 255
  - Currently, we have reserved PE#0 to which all RIDs are mapped prior to PE assignment request from kernel. The last M64 BAR is configured to have shared mode. So we have to cut off the first M64 segment, which corresponds to reserved PE#0 in kernel. If the first BAR (for example PF's IOV BAR) requires huge alignment in kernel, we have to waste huge M64 space to accommodate the alignment. If we have reserved PE#256, the waste of M64 space will be avoided.

Other changes

- unified version numbers for bundled utilities
- external/boot_test/boot_test.sh
  - better usable for automated boot testing

5.1.4 skiboot-5.1.1

skiboot-5.1.1 was released on August 18th, 2015.

skiboot-5.1.1 is the send stable release of 5.1, it follows skiboot-5.1.0.

Skiboot 5.1.1 contains all fixes from skiboot-5.1.0 and is a minor bugfix release.

Changes

Over skiboot-5.1.0, we have the following changes:

- Fix detection of compiler options on ancient GCC (e.g. gcc 4.4, shipped with RHEL6)
- ensure the GNUC version defines for GCOV are coming from target CC rather than host CC for extract-gcov
- phb3: Continue CAPP setup even if PHB is already in CAPP mode This fixes a critical bug in CAPI support.

  CAPI requires that all faults are escalated into a fence, not a freeze. This is done by setting bits in a number of MMIO registers. phb3_set_capi_mode() calls phb3_init_capp_errors() to do this. However, if the PHB is already in CAPP mode - for example in the recovery case - phb3_set_capi_mode() will bail out early, and those registers will not be set.

  This is quite easy to verify. PCI config space access errors, for example, normally cause a freeze. On a CAPI-mode PHB, they should cause a fence. Say we have a CAPI card on PHB 0, and we inject a PCI config space error:

  ```
  echo 0x8000000000000000 > /sys/kernel/debug/powerpc/PCI0000/err_injct_inboundA;
  lspci;
  ```

  The first time we inject this, the PHB will fence and recover, but won't reset the registers. Therefore, the second time we inject it, we will incorrectly freeze, not fence.

  Worse, the recovery for the resultant EEH freeze event interacts poorly with the CAPP, triggering an EEH recovery of the PHB. The combination of the two attempted recoveries will get the PHB into an inoperable state.

5.1.5 skiboot-5.1.10

skiboot-5.1.10 was released on Friday November 13th, 2015.

skiboot-5.1.10 is the 11th stable release of 5.1, it follows skiboot-5.1.9 (which was released October 30th, 2015).

Skiboot 5.1.10 contains all fixes from skiboot-5.1.9 and is a minor bug fix release.

Over skiboot-5.1.9, we have the following change:

IBM FSP machines

- FSP: Handle Delayed Power Off initiated CEC shutdown with FSP in Reset/Reload

  In a scenario where the DPO has been initiated, but the FSP then went into reset before the CEC power down came in, OPAL may not give up the link since it may never see the PSI interrupt. So, if we are in dpo_pending and an FSP reset is detected via the DISR, give up the PSI link voluntarily.

Generic

- sensor: add a compatible property OPAL needs an extra compatible property "ibm,opal-sensor" to make module autoload work smoothly in Linux for ibmpowernv driver.
- console: Completely flush output buffer before power down and reboot Completely flush the output buffer of the console driver before power down and reboot. Implements the flushing function for uart consoles, which includes the astbmc and rhesus platforms.

This fixes an issue where some console output is sometimes lost before power down or reboot in uart consoles. If this issue is also prevalent in other console types then it can be fixed later by adding a .flush to that driver's con_ops.

5.1.6 skiboot-5.1.11

skiboot-5.1.11 was released on Friday November 13th, 2015.

Since it was Friday 13th, we had to find a bug right after we tagged and released skiboot-5.1.10.

skiboot-5.1.11 is the 12th stable release of 5.1, it follows skiboot-5.1.10 (which was released November 13th, 2015).

Skiboot 5.1.11 contains one additional bug fix over skiboot-5.1.10.

It is:

- On IBM FSP machines, if IPMI/Serial console is not connected during shutdown or reboot, machine would enter termination state rather than shut down.

5.1.7 skiboot-5.1.12

skiboot-5.1.12 was released on Friday December 4th, 2015.

skiboot-5.1.12 is the 13th stable release of 5.1, it follows skiboot-5.1.11 (which was released November 13th, 2015).

Skiboot 5.1.12 contains bug fixes and a performance improvement.

opal-prd

- Display an explict and obvious message if running on a system that does not support opal-prd, such as an IBM FSP based POWER system, where the FSP takes on the role of opal-prd.

pflash

- Fix a missing (C) header - cherry-picked from master.

General

- Don't link with libgcc - On some toolchains, we don't have libgcc available.

POWER8 PHB (PCIe) specific

- **hw/phb3: Flush cache line after updating P/Q bits** When doing an MSI EOI, we update the P and Q bits in the IVE. That causes the corresponding cache line to be dirty in the L3 which will cause a subsequent update by the PHB (upon receiving the next MSI) to get a few retries until it gets flushed.

  We improve the situation (and thus performance) by doing a dcbf instruction to force a flush of the update we do in SW.

  This improves interrupt performance, reducing latency per interrupt. The improvement will vary by workload.

IBM FSP based machines

- FSP: Give up PSI link on shutdown This clears up some erroneous SRCs (error logs) in some situations.
- **Correctly report back Real Time Clock errors to host** Under certain rare error conditions, we could return an error code to the host OS that would cause current Linux kernels to get stuck in an infinite loop during boot. This was introduced in skiboot-5.0-rc1.

5.1.8 skiboot-5.1.13

skiboot-5.1.13 was released on Wed January 27th, 2016.

skiboot-5.1.13 is the 14th stable release of 5.1, it follows skiboot-5.1.12 (which was released December 4th, 2015). This release contains bug fixes.

General

- core/device.c: Sort nodes with name@unit names by unit
  - This gives predictable device tree ordering to the payload (usually petitboot)
  - This means that utilities such as "lspci" will always return the same ordering.
- Add OPAL_CONSOLE_FLUSH to the OPAL API uart consoles only flush output when polled. The Linux kernel calls these pollers frequently, except when in a panic state. As such, panic messages are not fully printed unless the system is configured to reboot after panic.

  This patch adds a new call to the OPAL API to flush the buffer. If the system has a uart console (i.e. BMC machines), it will incrementally flush the buffer, returning if there is more to be flushed or not. If the system has a different console, the function will have no effect. This will allow the Linux kernel to ensure that panic message have been fully printed out.

CAPI

- hmi: Identify the phb upon CAPI malfunction alert Previously, any error on a CAPI adapter would assume PHB0. This could cause issues on Firestone machines.

gard utility

- Fix displaying 'cleared' gard records When a garded component is replaced hostboot detects this and updates the gard partition.

  Previously, there was ambiguity on if the gard record ID or the whole gard record needed to be erased. This fix makes gard and hostboot agree.

firestone platform

- fix spacing in slot name The other SlotN names have no space.

5.1.9 skiboot-5.1.14

skiboot-5.1.14 was released on Wed March 9th, 2016.

skiboot-5.1.14 is the 15th stable release of 5.1, it follows skiboot-5.1.13 (which was released January 27th, 2016). This release contains a spelling fix in a log message and an added device tree property to enable older kernels (with bootloader support) to use a framebuffer that is redirected to the BMC VGA port.

As such, skiboot-5.1.14 has no advantage over skiboot-5.1.13 unless you are wanting the neat offb framebuffer trick.

Changes are:

- fsp: fix spelling of "advertise" in log message See: https://www.youtube.com/watch?v=8Gv0H-vPoDc
- Explicit 1:1 mapping in ranges properties have been added to PCI bridges. This allows a neat trick with offb and VGA ports that should probably not be told to young children.

5.1.10 skiboot-5.1.15

skiboot-5.1.15 was released on Wed March 16th, 2016.

skiboot-5.1.15 is the 16th stable release of 5.1, it follows skiboot-5.1.14 (which was released March 9th, 2016). This release contains one bug fix, a fix for a memory leak in an error path for AMI BMC based systems when logging non-severe errors. As such, it is a minor bug fix update.

5.1.11 skiboot-5.1.16

skiboot-5.1.16 was released on Friday April 29th, 2016.

skiboot-5.1.16 is the 17th stable release of 5.1, it follows skiboot-5.1.15 (which was released March 16th, 2016).

This release contains a few bug fixes and is a recommended upgrade.

Changes

PHB3 (all POWER8 platforms)

- hw/phb3: Ensure PQ bits are cleared in the IVC when masking IRQ When we mask an interrupt, we may race with another interrupt coming in from the hardware. If this occurs, the P and/or Q bit may end up being set but we never EOI/clear them. This could result in a lost interrupt or the next interrupt that comes in after re-enabling never being presented.

  This fixes a bug seen with some CAPI workloads which have lots of interrupt masking at the same time as high interrupt load. The fix is not specific to CAPI though.

- **hw/phb3: Fix potential race in EOI** When we EOI we need to clear the present (P) bit in the Interrupt Vector Cache (IVC). We must clear P ensuring that any additional interrupts that come in aren't lost while also maintaining coherency with the Interrupt Vector Table (IVT).

  To do this, the hardware provides a conditional update bit in the IVC. This bit ensures that generation counts between the IVT and the IVC updates are synchronised.

  Unfortunately we never set this the bit to conditionally update the P bit in the IVC based on the generation count. Also, we didn't set what we wanted the new generation count to be if the update was successful.

FSP platforms

- OPAL:Handle mbox response with bad status:0x24 during FSP termination OPAL committed a predictive log with SRC BB822411 in some situations.

Generic

- hmi: Fix a bug where partial hmi event was reported to host. This bug fix ensures the CPU PIR is reported correctly:

```
  [  305.628283] Fatal Hypervisor Maintenance interrupt [Not recovered]
  [  305.628341]  Error detail: Malfunction Alert
  [  305.628388]      HMER: 8040000000000000
- [  305.628423]       CPU PIR: 00000000
+ [  200.123021]    CPU PIR: 000008e8
  [  305.628458]  [Unit: VSU] Logic core check stop
```

5.1.12 skiboot-5.1.17

skiboot-5.1.17 was released on Thursday 21st July 2016.

skiboot-5.1.17 is the 18th stable release of 5.1, it follows skiboot-5.1.16 (which was released April 29th, 2016).

This release contains a few minor bug fixes.

Changes

All platforms:

- Fix a few typos in user visible (OPAL log) strings
- pci: Do a dummy config write to devices to establish bus number
- Make the XSCOM engine code more resilient to errors: - hw/xscom: Reset XSCOM engine after querying sleeping core FIR - hw/xscom: Reset XSCOM engine after finite number of retries when busy - xscom: Return OPAL_WRONG_STATE on XSCOM ops if CPU is asleep

5.1.13 skiboot-5.1.18

skiboot-5.1.18 was released on Friday 26th August 2016.

skiboot-5.1.18 is the 19th stable release of 5.1, it follows skiboot-5.1.17 (which was released July 21st, 2016).

This release contains a few minor bug fixes.

Changes are:

All platforms:

- opal/hmi: Fix a TOD HMI failure during a race condition. Rare race condition which meant we wouldn't recover from TOD error
- hw/phb3: Update capi initialization sequence The capi initialization sequence was revised in a circumvention document when a 'link down' error was converted from fatal to Endpoint Recoverable. Other, non-capi, register setup was corrected even before the initial open-source release of skiboot, but a few capi-related registers were

not updated then, so this patch fixes it. The point is that a link-down error detected by the UTL logic will lead to an AIB fence, so that the CAPP unit can detect the error.

FSP platforms:

- FSP/ELOG: Fix OPAL generated elog resend logic
- FSP/ELOG: Fix possible event notifier hangs
- FSP/ELOG: Disable event notification if list is not consistent
- FSP/ELOG: Fix OPAL generated elog event notification
- FSP/ELOG: Disable event notification during kexec

5.1.14 skiboot-5.1.2

skiboot-5.1.2 was released on September 9th, 2015.

skiboot-5.1.2 is the third stable release of 5.1, it follows skiboot-5.1.1 (which was released August 18th, 2015).

Skiboot 5.1.2 contains all fixes from skiboot-5.1.1 and is a minor bugfix release.

Changes

Over skiboot-5.1.1, we have the following changes:

- phb3: Handle fence in phb3_pci_msi_check_q to fix hang

  If the PHB is fenced during phb3_pci_msi_check_q, it can get stuck in an infinite loop waiting to lock the FFI. Further, as the phb lock is held during this function it will prevent any other CPUs from dealing with the fence, leading to the entire system hanging.

  If the PHB_FFI_LOCK returns all Fs, return immediately to allow the fence to be dealt with.
- phb3: Continue CAPP setup even if PHB is already in CAPP mode This fixes a critical bug in CAPI support.
- Platform hook for terminate call
  - on assert() or other firmware failure, we will make a SEL callout on ASTBMC platforms
  - (slight) refactor of code for IBM-FSP platforms
- refactor slot naming code
- Slot names for Habanero platform
- misc improvements in userspace utilities (incl pflash, gard)
- build improvements
  - fixes for two compiler warnings were squashed in 5.1.1 commit, re-introduce the fixes.
  - misc compiler/static analysis warning fixes
- gard utility:
  - If gard tool detects the GUARD PNOR partition is corrupted, it will pro-actively re-initialize it. Modern Hostboot is more sensitive to the content of the GUARD partition in order to boot.
  - Update record clearing to match Hostboots expectations We now write ECC bytes throughout the whole partition. Without this fix, hostboot may not bring up the machine.
  - In the event of a corrupted GUARD partition so that even the first entry cannot be read, the gard utility now provides the user with the option to wipe the entirety of the GUARD partition to attempt recovery.

- opal_prd utility:
  - Add run command to pass through commands to HostBoot RunTime (HBRT)
    - this is for OpenPower firmware developers only.
  - Add htmght-passthru command.
    - this is for OpenPower firmware developers only.
  - Add override interface to pass attribute-override information to HBRT.
  - Server sends response in error path, so that client doesn't block forever
- external/mambo tcl scripts
  - Running little-endian kernels in mambo requires HILE to be set properly, which requires a bump in the machine's pvr value to a DD2.x chip.

Stats

For skiboot-5.1.0 to 5.1.2: Processed 67 csets from 11 developers 1 employers found A total of 2258 lines added, 784 removed (delta 1474)

Developers with the most changesets

Stewart Smith	24 (35.8%)
Cyril Bur	18 (26.9%)
Vasant Hegde	8 (11.9%)
Neelesh Gupta	5 (7.5%)
Benjamin Herrenschmidt	5 (7.5%)
Daniel Axtens	2 (3.0%)
Samuel Mendoza-Jonas	1 (1.5%)
Vaidyanathan Srinivasan	1 (1.5%)
Vipin K Parashar	1 (1.5%)
Ian Munsie	1 (1.5%)
Michael Neuling	1 (1.5%)

Developers with the most changed lines

Cyril Bur	969 (42.5%)
Neelesh Gupta	433 (19.0%)
Benjamin Herrenschmidt	304 (13.3%)
Vasant Hegde	236 (10.3%)
Stewart Smith	163 (7.1%)
Vaidyanathan Srinivasan	135 (5.9%)
Vipin K Parashar	8 (0.4%)
Ian Munsie	8 (0.4%)
Daniel Axtens	2 (0.1%)
Michael Neuling	2 (0.1%)
Samuel Mendoza-Jonas	1 (0.0%)

Developers with the most lines removed

Daniel Axtens	2 (0.3%)
Michael Neuling	1 (0.1%)

Developers with the most signoffs (total 44)

Stewart Smith	43 (97.7%)
Neelesh Gupta	1 (2.3%)

Developers with the most reviews (total 8)

Patrick Williams	5 (62.5%)
Samuel Mendoza-Jonas	3 (37.5%)

Developers with the most test credits (total 0)

Developers who gave the most tested-by credits (total 0)

Developers with the most report credits (total 1)

Benjamin Herrenschmidt	1 (100.0%)

Developers who gave the most report credits (total 1)

Samuel Mendoza-Jonas	1 (100.0%)

Top changeset contributors by employer

IBM	67 (100.0%)

Top lines changed by employer

IBM	2281 (100.0%)

Employers with the most signoffs (total 44)

IBM	44 (100.0%)

Employers with the most hackers (total 11)

IBM	11 (100.0%)

5.1.15 skiboot-5.1.3

skiboot-5.1.3 was released on September 15th, 2015.

skiboot-5.1.3 is the 4th stable release of 5.1, it follows skiboot-5.1.2 (which was released September 9th, 2015).

Skiboot 5.1.3 contains all fixes from skiboot-5.1.2 and is a minor bugfix release.

Changes

Over skiboot-5.1.2, we have the following changes:

- slot names for firestone platform
- fix display of LPC errors
- SBE based timer support

  - on supported platforms limits reliance on Linux heartbeat

- fix use-after-free in fsp/ipmi
- fix hang on TOD/TB errors (time-of-day/timebase) on OpenPower systems
  - On getting a Hypervizor Maintenance Interrupt to get the timebase back into a running state, we would call prlog which would use the LPC UART console driver on OpenPower systems, which depends on a working timebase, leading to a hang. We now don't depend on a working timebase in this recovery codepath.
- enable prd for garrison platform
- PCI: Clear error bits after changing MPS Chaning MPS on PCI upstream bridge might cause error bits set on downstream endpoints when system boots into Linux as below case shows:

```
host# lspci -vvs 0001:06:00.0
0001:06:00.0 Ethernet controller: Broadcom Corporation \
             NetXtreme II BCM57810 10 Gigabit Ethernet (rev 10)
DevSta:      CorrErr+ UncorrErr- FatalErr- UnsuppReq+ AuxPwr- TransPend-
CESta:       RxErr- BadTLP- BadDLLP- Rollover- Timeout- NonFatalErr+
```

  This clears those error bits in AER and PCIe capability after MPS is changed. With the patch applied, no more error bits are seen.

Contributors

Processed 14 csets from 6 developers 1 employers found A total of 462 lines added, 163 removed (delta 299)

Developers with the most changesets

Benjamin Herrenschmidt	5 (35.7%)
Stewart Smith	4 (28.6%)
Mahesh Salgaonkar	2 (14.3%)
Gavin Shan	1 (7.1%)
Jeremy Kerr	1 (7.1%)
Neelesh Gupta	1 (7.1%)

Developers with the most changed lines

Benjamin Herrenschmidt	407 (80.8%)
Mahesh Salgaonkar	23 (4.6%)
Gavin Shan	19 (3.8%)
Stewart Smith	18 (3.6%)
Jeremy Kerr	5 (1.0%)
Neelesh Gupta	2 (0.4%)

Developers with the most lines removed

Stewart Smith	8 (4.9%)
Jeremy Kerr	3 (1.8%)
Neelesh Gupta	1 (0.6%)

Developers with the most signoffs (total 10)

Stewart Smith	10 (100.0%)

Developers with the most reviews (total 1)

Joel Stanley	1 (100.0%)

Developers with the most test credits (total 0)

Developers who gave the most tested-by credits (total 0)

Developers with the most report credits (total 1)

John Walthour	1 (100.0%)

Developers who gave the most report credits (total 1)

Gavin Shan	1 (100.0%)

Top changeset contributors by employer

IBM	14 (100.0%)

Top lines changed by employer

IBM	504 (100.0%)

Employers with the most signoffs (total 10)

IBM	10 (100.0%)

Employers with the most hackers (total 6)

IBM	6 (100.0%)

5.1.16 skiboot-5.1.4

skiboot-5.1.4 was released on September 26th, 2015.

skiboot-5.1.4 is the 5th stable release of 5.1, it follows skiboot-5.1.3 (which was released September 15th, 2015).

Skiboot 5.1.4 contains all fixes from skiboot-5.1.3 and is an important bug fix release and a strongly recommended update from any prior skiboot-5.1.x release.

Changes

Over skiboot-5.1.3, we have the following changes:

- Rate limit OPAL_MSG_OCC to only one outstanding message to host

  In the event of a lot of OCC events (or many CPU cores), we could send many OCC messages to the host, which if it wasn't calling opal_get_msg really often, would cause skiboot to malloc() additional messages until we ran out of skiboot heap and things didn't end up being much fun.

  When running certain hardware exercisers, they seem to steal all time from Linux being able to call opal_get_msg, causing these to queue up and get "opalmsg: No available node in the free list, allocating" warnings followed by tonnes of backtraces of failing memory allocations.

- Ensure reserved memory ranges are exposed correctly to host (fix corrupted SLW image)

  We seem to have not hit this on ASTBMC based OpenPower machines, but was certainly hit on FSP based machines

5.1.17 skiboot-5.1.5

skiboot-5.1.5 was released on October 1st, 2015.

skiboot-5.1.5 is the 6th stable release of 5.1, it follows skiboot-5.1.4 (which was released September 26th, 2015).

Skiboot 5.1.5 contains all fixes from skiboot-5.1.4 and is a minor bug fix release.

Changes

Over skiboot-5.1.4, we have the following changes:

Generic

- centaur: Add indirect XSCOM support Fixes a bug where opal-prd would not be able to recover from a bunch of errors as the indirect XSCOMs to centaurs would fail.
- xscom: Fix logging of indirect XSCOM errors Better logging of error messages.
- PHB3: Fix wrong PE number in error injection
- Improvement in boot_test.sh utility to support copying a pflash binary to BMCs.

AST BMC machines

- **ipmi-sel: Run power action immediately if host not up** Our normal sequence for a soft power action (IPMI 'power soft' or 'power cycle') involve receiving a SEL from the BMC, sending a message to Linux's opal platform support which instructs the host OS to shut down, and finally the host will request OPAL to cut power.

  When the host is not yet up we will send the message to /dev/null, and no action will be taken. This patches changes that behaviour to perform the action immediately if we know how.

OpenPower machines:

- opal-prd: Increase IPMI timeout to a slightly better value Proactively bump the timeout to 5seconds to match current value in petitboot Observed in the wild that this fixes bugs for petitboot.

5.1.18 skiboot-5.1.6

skiboot-5.1.6 was released on October 8th, 2015.

skiboot-5.1.6 is the 7th stable release of 5.1, it follows skiboot-5.1.5 (which was released October 1st, 2015).

Skiboot 5.1.6 contains all fixes from skiboot-5.1.5 and is a minor bug fix release.

Changes

Over skiboot-5.1.5, we have the following changes:

Generic:

- Ensure we run pollers in cpu_wait_job()

  In root causing a bug on AST BMC Alistair found that pollers weren't being run for around 3800ms.

  This could show as not resetting the boot count sensor on successful boot.

AST BMC Machines

- hw/bt.c: Check for timeout after checking for message response

  When deciding if a BT message has timed out we should first check for a message response. This will ensure that messages will not time out if there was a delay calling the pollers.

  This could show as not resetting the boot count sensor on successful boot.

5.1.19 skiboot-5.1.7

skiboot-5.1.7 was released on October 13th, 2015.

skiboot-5.1.7 is the 8th stable release of 5.1, it follows skiboot-5.1.6 (which was released October 8th, 2015).

Skiboot 5.1.7 contains all fixes from skiboot-5.1.6 and is a minor bug fix release with one important bug fix for FSP systems.

Over skiboot-5.1.6, we have the following changes:

Generic:

- **PHB3: Retry fundamental reset** This introduces another PHB3 state (PHB3_STATE_FRESET_START) allowing to redo fundamental reset if the link doesn't come up in time at the first attempt, to improve the robustness of PHB's fundamental reset. If the link comes up after the first reset, the 2nd reset won't be issued at all.

FSP based systems:

- hw/fsp/fsp-leds.c: use allocated buffer for FSP_CMD_GET_LED_LIST response

  This fixes a bug where we would overwrite roughly 4kb of memory belonging to Linux when the FSP would ask firmware for a list of LEDs in the system. This wouldn't happen often (once before Linux was running and possibly only once during runtime, and *early* runtime at that) but it was possible for this corruption to show up and be detected.

5.1.20 skiboot-5.1.8

skiboot-5.1.8 was released on October 19th, 2015.

skiboot-5.1.8 is the 9th stable release of 5.1, it follows skiboot-5.1.7 (which was released October 13th, 2015).

Skiboot 5.1.8 contains all fixes from skiboot-5.1.7 and is a minor bug fix release, with a single fix for recovery from a (rare) error.

Over skiboot-5.1.7, we have the following change:

- opal/hmi: Fix a soft lockup issue on Hypervisor Maintenance Interrupt for certain timebase errors.

  We also introduce a timeout to handle the worst situation where all other threads are badly stuck without setting a cleanup done bit. Under such situation timeout will help to avoid soft lockups and report failure to kernel.

5.1.21 skiboot-5.1.9

skiboot-5.1.9 was released on October 30th, 2015.

skiboot-5.1.9 is the 10th stable release of 5.1, it follows skiboot-5.1.8 (which was released October 19th, 2015).

Skiboot 5.1.9 contains all fixes from skiboot-5.1.8 and is a minor bug fix release, with a single fix to help diagnosis after a rare error condition.

Over skiboot-5.1.8, we have the following change:

- opal/hmi: Signal PRD about NX unit checkstop. We now signal Processor Recovery & Diagnostics (PRD) correctly following an NX unit checkstop
- minor fix to the boot_test.sh test script

5.1.22 skiboot-5.2.0

skiboot-5.2.0 was released on Wednesday March 16th, 2016.

skiboot-5.2.0 is the first stable release of skiboot 5.2, the new stable release of skiboot, which will take over from the 5.1.x series which was first released August 17th, 2015.

skiboot-5.2.0 contains all bug fixes as of skiboot-5.1.15.

This is the second release that will follow the (now documented) Skiboot stable rules - see doc/stable-skiboot-rules.txt.

Changes since rc2

Over skiboot-5.2.0-rc2, the following fixes are included:

- Include 'extract-gcov' in make clean.
- ipmi-sel: Fix esel event logger to handle early boot PANIC events
- IPMI: Enable synchronous eSEL logging option (for PANIC events)
- libflash/libffs: Reporting seeing all 0xFF bytes during init.
- ipmi-sel: Fix memory leak in error path

Changes since rc1

Over skiboot-5.2.0-rc1, we have the following changes:

- Add Barreleye platform

Generic

- hw/p8-i2c: Speed up SMBUS_WRITE
- Fix early backtraces

FSP Platforms

- fsp-sensor: rework device tree for sensors
- platforms/firenze: Fix I2C clock source frequency

Simics simulator

- Enable Simics UART console

Mambo simulator

- platforms/mambo: Add terminate callback
  - fix hang in multi-threaded mambo
  - add multithreaded mambo tests

IPMI

- hw/ipmi: fix event data 1 for System Firmware Progress sensor
- ipmi: Log exact NetFn value in OPAL logs

AST BMC based platforms

- hw/bt: allow BT driver to use different buffer size

opal-prd utility

- **opal-prd: Add debug output for firmware-driven OCC events** We indicate when we have a user-driven event, so add corresponding outputs for firmware-driven ones too.

getscom utility

- Add Naples chip support

New Features

Over skiboot-5.1, the following features have been added:

- Naples (P8', i.e. P8 with NVLINK) processor support, including NVLINK.
- Improvements in gard, libflash/pflash and opal-prd utilities
  - increased testing
  - increased usability
  - systemd scripts for opal-prd

  - pflash can now use the /dev/mtd device to access BMC flash rather than accessing it directly. It is *important* that you use –mtd if your BMC may otherwise know how to interact with its own flash.

- support for Micron N25Q256Ax and N25Qx256Ax NOR flash.
- support for Winbond W25Q256BV NOR flash
- support for an emulated ("fake") RTC clock, useful in simulators and during bringup
- Explicit 1:1 mapping in ranges properties have been added to PCI bridges. This allows a neat trick with offb and VGA ports that should probably not be told to young children.
- Added support to read the V2 format of the OCC-OPAL memory region, which supports Workload Optimized Frequency (WOF)

Changes in behavior

- Assigning OPAL IDs to PHBs is now fixed and based on the chip id and PHB index on that chip. On POWER7, we continue to use allocated numbers.
- We now query the BMC for BT capabilities rather than making assumptions

Removed support

- p5ioc2 is no longer supported. This affects a grand total of two POWER7 systems in the world.

NOTE: It is planned that skiboot-5.2 will be the last release supporting POWER7 machines.

Bugs fixed

- PHB3: Fix unexpected ER (all) on errinjct by PCI config
- hw/bt: timeout messages when BT interface isn't functional
- On Habanero, Slot3 should have been "Slot 3".
- We now completely flush the console buffer before power down and reboot
- For chips with ibm,occ-functional-state set to false, we don't wait for the OCC to start. This caused needless delay in booting on simulators which did not simulate OCCs.
- Change OCC reset order to always reset slave OCCs first.
- slw: Remove overwrites for EX_PM_CORE_ECO_VRET and EX_PM_CORE_PFET_VRET (these were already initialized in hostboot)
- p8-i2c: send stop bit on timeouts. Some devices can otherwise leave the bus in a held state.

Other improvements

- many fixes of compiler and static analysis warnings
- increased unit test coverage
- Unit test of "boot debian jessie installer"
- ability to plug in other simulators to run existing tests (e.g. simulator for non pegasus p8)
- Support using (patched) Qemu with PowerNV platform support for running unit tests.

- increased support for running with sparse
- We now build with -fstack-protector-strong if supported by the compiler
- We now build with -Werror for -Wformat
- pflash is now built as part of travis-ci and for Coverity Scan.
- There is now a RPM SPEC file that can be used as the basis for packaging skiboot and associated utilities.

Contributors

We have had a number of improvements in workflow over skiboot-5.1.0. Looking back, we have roughly the same number of changesets (372 for 5.1.0, 334 for 5.2.0-rc1 - even closer for 5.1.0-beta1) which indicates a relatively stable rate of development.

Complete statistics are included below (generated by gitdm), but I'd like to draw attention to a couple of stats:

Release	csets	Ack	Reviews	Tested	Reported
5.0	329	15	20	1	0
5.1	372	13	38	1	4
5.2-rc1	334	20	34	6	11

Overall, it looks like we're on the right trajectory for increasing the number of eyeballs looking at code before it heads in tree, especially around testing. Largely, this increase in Tested-by can be attributed to encouraging the existing test teams to start commenting on the patches themselves.

Anyway, here's the full stats from skiboot 5.1.0 to 5.2.0-rc1:

Processed 334 csets from 27 developers 2 employers found A total of 46172 lines added, 23274 removed (delta 22898)

Developers with the most changesets

Stewart Smith	146 (43.7%)
Cyril Bur	52 (15.6%)
Benjamin Herrenschmidt	15 (4.5%)
Joel Stanley	12 (3.6%)
Gavin Shan	12 (3.6%)
Alistair Popple	10 (3.0%)
Vasant Hegde	10 (3.0%)
Michael Neuling	10 (3.0%)
Russell Currey	9 (2.7%)
Cédric Le Goater	8 (2.4%)
Jeremy Kerr	8 (2.4%)
Samuel Mendoza-Jonas	6 (1.8%)
Neelesh Gupta	6 (1.8%)
Shilpasri G Bhat	4 (1.2%)
Oliver O'Halloran	4 (1.2%)
Mahesh Salgaonkar	4 (1.2%)
Vipin K Parashar	3 (0.9%)
Daniel Axtens	3 (0.9%)
Andrew Donnellan	2 (0.6%)
Philippe Bergheaud	2 (0.6%)
Ananth N Mavinakayanahalli	2 (0.6%)
Vaibhav Jain	1 (0.3%)
Sam Mendoza-Jonas	1 (0.3%)
Adriana Kobylak	1 (0.3%)
Shreyas B. Prabhu	1 (0.3%)
Vaidyanathan Srinivasan	1 (0.3%)
Ian Munsie	1 (0.3%)

Developers with the most changed lines

Stewart Smith	19533 (39.4%)
Oliver O'Halloran	17920 (36.1%)
Alistair Popple	3285 (6.6%)
Daniel Axtens	2154 (4.3%)
Cyril Bur	2028 (4.1%)
Benjamin Herrenschmidt	941 (1.9%)
Neelesh Gupta	434 (0.9%)
Gavin Shan	294 (0.6%)
Russell Currey	261 (0.5%)
Vasant Hegde	245 (0.5%)
Cédric Le Goater	209 (0.4%)
Vipin K Parashar	155 (0.3%)
Shilpasri G Bhat	153 (0.3%)
Joel Stanley	140 (0.3%)
Vaidyanathan Srinivasan	135 (0.3%)
Michael Neuling	111 (0.2%)
Samuel Mendoza-Jonas	81 (0.2%)
Jeremy Kerr	60 (0.1%)
Mahesh Salgaonkar	58 (0.1%)
Vaibhav Jain	50 (0.1%)
Ananth N Mavinakayanahalli	43 (0.1%)
Shreyas B. Prabhu	17 (0.0%)
Sam Mendoza-Jonas	12 (0.0%)
Andrew Donnellan	10 (0.0%)
Ian Munsie	8 (0.0%)
Philippe Bergheaud	6 (0.0%)
Adriana Kobylak	6 (0.0%)

Developers with the most lines removed

Daniel Axtens	2149 (9.2%)
Shreyas B. Prabhu	17 (0.1%)
Andrew Donnellan	9 (0.0%)
Vipin K Parashar	2 (0.0%)

Developers with the most signoffs (total 190)

Stewart Smith	188 (98.9%)
Gavin Shan	1 (0.5%)
Neelesh Gupta	1 (0.5%)

Developers with the most reviews (total 34)

Patrick Williams	5 (14.7%)
Joel Stanley	5 (14.7%)
Cédric Le Goater	5 (14.7%)
Vasant Hegde	4 (11.8%)
Alistair Popple	4 (11.8%)
Sam Mendoza-Jonas	3 (8.8%)
Samuel Mendoza-Jonas	3 (8.8%)
Andrew Donnellan	2 (5.9%)
Cyril Bur	2 (5.9%)
Vaibhav Jain	1 (2.9%)

Developers with the most test credits (total 6)

Vipin K Parashar	3 (50.0%)
Vaibhav Jain	2 (33.3%)
Gajendra B Bandhu1	1 (16.7%)

Developers who gave the most tested-by credits (total 6)

Gavin Shan	2 (33.3%)
Ananth N Mavinakayanahalli	2 (33.3%)
Alistair Popple	1 (16.7%)
Stewart Smith	1 (16.7%)

Developers with the most report credits (total 11)

Vaibhav Jain	2 (18.2%)
Paul Nguyen	2 (18.2%)
Alistair Popple	1 (9.1%)
Cédric Le Goater	1 (9.1%)
Aneesh Kumar K.V	1 (9.1%)
Dionysius d. Bell	1 (9.1%)
Pradeep Ramanna	1 (9.1%)
John Walthour	1 (9.1%)
Benjamin Herrenschmidt	1 (9.1%)

Developers who gave the most report credits (total 11)

Gavin Shan	6 (54.5%)
Stewart Smith	3 (27.3%)
Samuel Mendoza-Jonas	1 (9.1%)
Shilpasri G Bhat	1 (9.1%)

5.1.23 skiboot-5.2.0-rc1

skiboot-5.2.0-rc1 was released on Friday Feb 26th, 2016.

skiboot-5.2.0-rc1 is the first release candidate of skiboot 5.2, which will become the new stable release of skiboot following the 5.1 release, first released August 17th, 2015.

skiboot-5.2.0-rc1 contains all bug fixes as of skiboot-5.1.13.

This is the second release that will follow the (now documented) Skiboot stable rules - see doc/stable-skiboot-rules.txt.

The current plan is to release skiboot-5.2.0 mid-March 2016, with a focus on bug fixing for future 5.2.0-rc releases.

New Features

Over skiboot-5.1, the following features have been added:

- Naples (P8', i.e. P8 with NVLINK) processor support, including NVLINK.
- Improvements in gard, libflash/pflash and opal-prd utilities
  - increased testing
  - increased usability
  - systemd scripts for opal-prd
  - pflash can now use the /dev/mtd device to access BMC flash rather than accessing it directly. It is *important* that you use –mtd if your BMC may otherwise know how to interact with its own flash.
- support for Micron N25Q256Ax and N25Qx256Ax NOR flash.
- support for Winbond W25Q256BV NOR flash
- support for an emulated ("fake") RTC clock, useful in simulators and during bringup
- Explicit 1:1 mapping in ranges properties have been added to PCI bridges. This allows a neat trick with offb and VGA ports that should probably not be told to young children.
- Added support to read the V2 format of the OCC-OPAL memory region, which supports Workload Optimized Frequency (WOF)

Changes in behavior

- Assigning OPAL IDs to PHBs is now fixed and based on the chip id and PHB index on that chip. On POWER7, we continue to use allocated numbers.
- We now query the BMC for BT capabilities rather than making assumptions

Removed support

- p5ioc2 is no longer supported. This affects a grand total of two POWER7 systems in the world.

NOTE: It is planned that skiboot-5.2 will be the last release supporting POWER7 machines.

Bugs fixed

- PHB3: Fix unexpected ER (all) on errinjct by PCI config
- hw/bt: timeout messages when BT interface isn't functional
- On Habanero, Slot3 should have been "Slot 3".
- We now completely flush the console buffer before power down and reboot
- For chips with ibm,occ-functional-state set to false, we don't wait for the OCC to start. This caused needless delay in booting on simulators which did not simulate OCCs.
- Change OCC reset order to always reset slave OCCs first.
- slw: Remove overwrites for EX_PM_CORE_ECO_VRET and EX_PM_CORE_PFET_VRET (these were already initialized in hostboot)

- p8-i2c: send stop bit on timeouts. Some devices can otherwise leave the bus in a held state.

Other improvements

- many fixes of compiler and static analysis warnings
- increased unit test coverage
- Unit test of "boot debian jessie installer"
- ability to plug in other simulators to run existing tests (e.g. simulator for non pegasus p8)
- Support using (patched) Qemu with PowerNV platform support for running unit tests.
- increased support for running with sparse
- We now build with -fstack-protector-strong if supported by the compiler
- We now build with -Werror for -Wformat
- pflash is now built as part of travis-ci and for Coverity Scan.
- There is now a RPM SPEC file that can be used as the basis for packaging skiboot and associated utilities.

Contributors

We have had a number of improvements in workflow over skiboot-5.1.0. Looking back, we have roughly the same number of changesets (372 for 5.1.0, 334 for 5.2.0-rc1 - even closer for 5.1.0-beta1) which indicates a relatively stable rate of development.

Complete statistics are included below (generated by gitdm), but I'd like to draw attention to a couple of stats:

Release	csets	Ack	Reviews	Tested	Reported
5.0	329	15	20	1	0
5.1	372	13	38	1	4
5.2-rc1	334	20	34	6	11

Overall, it looks like we're on the right trajectory for increasing the number of eyeballs looking at code before it heads in tree, especially around testing. Largely, this increase in Tested-by can be attributed to encouraging the existing test teams to start commenting on the patches themselves.

Anyway, here's the full stats from skiboot 5.1.0 to 5.2.0-rc1:

Processed 334 csets from 27 developers 2 employers found A total of 46172 lines added, 23274 removed (delta 22898)

Developers with the most changesets

Stewart Smith	146 (43.7%)
Cyril Bur	52 (15.6%)
Benjamin Herrenschmidt	15 (4.5%)
Joel Stanley	12 (3.6%)
Gavin Shan	12 (3.6%)
Alistair Popple	10 (3.0%)
Vasant Hegde	10 (3.0%)
Michael Neuling	10 (3.0%)
Russell Currey	9 (2.7%)
Cédric Le Goater	8 (2.4%)
Jeremy Kerr	8 (2.4%)
Samuel Mendoza-Jonas	6 (1.8%)
Neelesh Gupta	6 (1.8%)
Shilpasri G Bhat	4 (1.2%)
Oliver O'Halloran	4 (1.2%)
Mahesh Salgaonkar	4 (1.2%)
Vipin K Parashar	3 (0.9%)
Daniel Axtens	3 (0.9%)
Andrew Donnellan	2 (0.6%)
Philippe Bergheaud	2 (0.6%)
Ananth N Mavinakayanahalli	2 (0.6%)
Vaibhav Jain	1 (0.3%)
Sam Mendoza-Jonas	1 (0.3%)
Adriana Kobylak	1 (0.3%)
Shreyas B. Prabhu	1 (0.3%)
Vaidyanathan Srinivasan	1 (0.3%)
Ian Munsie	1 (0.3%)

Developers with the most changed lines

Stewart Smith	19533 (39.4%)
Oliver O'Halloran	17920 (36.1%)
Alistair Popple	3285 (6.6%)
Daniel Axtens	2154 (4.3%)
Cyril Bur	2028 (4.1%)
Benjamin Herrenschmidt	941 (1.9%)
Neelesh Gupta	434 (0.9%)
Gavin Shan	294 (0.6%)
Russell Currey	261 (0.5%)
Vasant Hegde	245 (0.5%)
Cédric Le Goater	209 (0.4%)
Vipin K Parashar	155 (0.3%)
Shilpasri G Bhat	153 (0.3%)
Joel Stanley	140 (0.3%)
Vaidyanathan Srinivasan	135 (0.3%)
Michael Neuling	111 (0.2%)
Samuel Mendoza-Jonas	81 (0.2%)
Jeremy Kerr	60 (0.1%)
Mahesh Salgaonkar	58 (0.1%)
Vaibhav Jain	50 (0.1%)
Ananth N Mavinakayanahalli	43 (0.1%)
Shreyas B. Prabhu	17 (0.0%)
Sam Mendoza-Jonas	12 (0.0%)
Andrew Donnellan	10 (0.0%)
Ian Munsie	8 (0.0%)
Philippe Bergheaud	6 (0.0%)
Adriana Kobylak	6 (0.0%)

Developers with the most lines removed

Daniel Axtens	2149 (9.2%)
Shreyas B. Prabhu	17 (0.1%)
Andrew Donnellan	9 (0.0%)
Vipin K Parashar	2 (0.0%)

Developers with the most signoffs (total 190)

Stewart Smith	188 (98.9%)
Gavin Shan	1 (0.5%)
Neelesh Gupta	1 (0.5%)

Developers with the most reviews (total 34)

Patrick Williams	5 (14.7%)
Joel Stanley	5 (14.7%)
Cédric Le Goater	5 (14.7%)
Vasant Hegde	4 (11.8%)
Alistair Popple	4 (11.8%)
Sam Mendoza-Jonas	3 (8.8%)
Samuel Mendoza-Jonas	3 (8.8%)
Andrew Donnellan	2 (5.9%)
Cyril Bur	2 (5.9%)
Vaibhav Jain	1 (2.9%)

Developers with the most test credits (total 6)

Vipin K Parashar	3 (50.0%)
Vaibhav Jain	2 (33.3%)
Gajendra B Bandhu1	1 (16.7%)

Developers who gave the most tested-by credits (total 6)

Gavin Shan	2 (33.3%)
Ananth N Mavinakayanahalli	2 (33.3%)
Alistair Popple	1 (16.7%)
Stewart Smith	1 (16.7%)

Developers with the most report credits (total 11)

Vaibhav Jain	2 (18.2%)
Paul Nguyen	2 (18.2%)
Alistair Popple	1 (9.1%)
Cédric Le Goater	1 (9.1%)
Aneesh Kumar K.V	1 (9.1%)
Dionysius d. Bell	1 (9.1%)
Pradeep Ramanna	1 (9.1%)
John Walthour	1 (9.1%)
Benjamin Herrenschmidt	1 (9.1%)

Developers who gave the most report credits (total 11)

Gavin Shan	6 (54.5%)
Stewart Smith	3 (27.3%)
Samuel Mendoza-Jonas	1 (9.1%)
Shilpasri G Bhat	1 (9.1%)

5.1.24 skiboot-5.2.0-rc2

skiboot-5.2.0-rc2 was released on Wednesday March 9th, 2016.

skiboot-5.2.0-rc2 is the second release candidate of skiboot 5.2, which will become the new stable release of skiboot following the 5.1 release, first released August 17th, 2015.

skiboot-5.2.0-rc2 contains all bug fixes as of skiboot-5.1.14.

This is the second release that will follow the (now documented) Skiboot stable rules - see doc/stable-skiboot-rules.txt.

The current plan is to release skiboot-5.2.0 mid-March 2016, with a focus on bug fixing for future 5.2.0-rc releases (if any - I hope this will be the last)

Over skiboot-5.2.0-rc1, we have the following changes:

New platform!

- Add Barreleye platform

Generic

- hw/p8-i2c: Speed up SMBUS_WRITE
- Fix early backtraces

FSP Platforms

- fsp-sensor: rework device tree for sensors
- platforms/firenze: Fix I2C clock source frequency

Simics simulator

- Enable Simics UART console

Mambo simulator

- platforms/mambo: Add terminate callback
  - fix hang in multi-threaded mambo
  - add multithreaded mambo tests

IPMI

- hw/ipmi: fix event data 1 for System Firmware Progress sensor
- ipmi: Log exact NetFn value in OPAL logs

AST BMC based platforms

- hw/bt: allow BT driver to use different buffer size

opal-prd utility

- **opal-prd: Add debug output for firmware-driven OCC events** We indicate when we have a user-driven event, so add corresponding outputs for firmware-driven ones too.

getscom utility

- Add Naples chip support

5.1.25 skiboot-5.2.1

skiboot-5.2.1 was released on Wednesday April 27th, 2016.

skiboot-5.2.1 is the second stable release of skiboot 5.2, the new stable release of skiboot, which will take over from the 5.1.x series which was first released August 17th, 2015.

skiboot-5.2.1 contains all bug fixes as of skiboot-5.1.15.

This is the second release that will follow the (now documented) Skiboot stable rules - see doc/stable-skiboot-rules.txt.

Changes

Over skiboot-5.2.0, the following fixes are included:

pflash

- Allow building under yocto. Makefile fixes to enable building as part of an OpenBMC build.

Garrison platform

- Add PCIe and NPU slot location names
- hw/npu.c: Add ibm, npu-index property to npu device tree
- hmi: Add handling for NPU checkstops

PHB3 (all POWER8 platforms)

- hw/phb3: Ensure PQ bits are cleared in the IVC when masking IRQ When we mask an interrupt, we may race with another interrupt coming in from the hardware. If this occurs, the P and/or Q bit may end up being set but we never EOI/clear them. This could result in a lost interrupt or the next interrupt that comes in after re-enabling never being presented.

  This fixes a bug seen with some CAPI workloads which have lots of interrupt masking at the same time as high interrupt load. The fix is not specific to CAPI though.

- hw/phb3: Fix potential race in EOI When we EOI we need to clear the present (P) bit in the Interrupt Vector Cache (IVC). We must clear P ensuring that any additional interrupts that come in aren't lost while also maintaining coherency with the Interrupt Vector Table (IVT).

  To do this, the hardware provides a conditional update bit in the IVC. This bit ensures that generation counts between the IVT and the IVC updates are synchronised.

  Unfortunately we never set this the bit to conditionally update the P bit in the IVC based on the generation count. Also, we didn't set what we wanted the new generation count to be if the update was successful.

FSP platforms

- OPAL:Handle mbox response with bad status:0x24 during FSP termination OPAL committed a predictive log with SRC BB822411 in some situations.

Generic

- hmi: Fix a bug where partial hmi event was reported to host. This bug fix ensures the CPU PIR is reported correctly:

```
  [  305.628283] Fatal Hypervisor Maintenance interrupt [Not recovered]
  [  305.628341]  Error detail: Malfunction Alert
  [  305.628388]    HMER: 8040000000000000
- [  305.628423]    CPU PIR: 00000000
+ [  200.123021]    CPU PIR: 000008e8
  [  305.628458]    [Unit: VSU] Logic core check stop
```

- xscom: Return OPAL_WRONG_STATE on XSCOM ops if CPU is asleep

Contributors

Processed 15 csets from 7 developers A total of 436 lines added, 59 removed (delta 377)

Developers with the most changesets

Russell Currey	7 (46.7%)
Alistair Popple	2 (13.3%)
Michael Neuling	2 (13.3%)
Patrick Williams	1 (6.7%)
Stewart Smith	1 (6.7%)
Mamatha	1 (6.7%)
Mahesh Salgaonkar	1 (6.7%)

Developers with the most changed lines

Alistair Popple	215 (48.3%)
Russell Currey	140 (31.5%)
Michael Neuling	55 (12.4%)
Mamatha	15 (3.4%)
Patrick Williams	9 (2.0%)
Mahesh Salgaonkar	8 (1.8%)
Stewart Smith	3 (0.7%)

Developers with the most lines removed

Patrick Williams	5 (8.5%)

Developers with the most signoffs (total 30)

Stewart Smith	15 (50.0%)
Russell Currey	7 (23.3%)
Michael Neuling	2 (6.7%)
Alistair Popple	2 (6.7%)
Patrick Williams	1 (3.3%)
Oliver O'Halloran	1 (3.3%)
Mahesh Salgaonkar	1 (3.3%)
Mamatha	1 (3.3%)

Developers with the most reviews (total 11)

Alistair Popple	5 (45.5%)
Andrew Donnellan	3 (27.3%)
Mahesh Salgaonkar	2 (18.2%)
Joel Stanley	1 (9.1%)

Developers with the most Acked-by (total 1)

Alistair Popple	1 (100.0%)

Developers with the most test credits (total 3)

Andrew Donnellan	2 (66.7%)
Vaibhav Jain	1 (33.3%)

Developers who received the most tested-by credits (total 3)

Michael Neuling	3 (100.0%)

5.1.26 skiboot-5.2.2

skiboot-5.2.2 was released on Thursday May 5th, 2016.

skiboot-5.2.2 is the third stable release of skiboot 5.2, the new stable release of skiboot, which will take over from the 5.1.x series which was first released August 17th, 2015.

Skiboot 5.2.2 replaces skiboot-5.2.1 as the current stable version, which was released on April 27th, 2016. Over skiboot-5.2.1, skiboot 5.2.2 contains one bug fix targeted at P8NVL systems, notably the Garrison platform.

skiboot-5.2.2 contains all bug fixes as of skiboot-5.1.16.

This is the second release that will follow the (now documented) Skiboot stable rules - see doc/stable-skiboot-rules.txt.

Over skiboot-5.2.1, the following fixes are included:

P8NVL/Garrison

- **PHB3: Fix corruption of pref window register** On P8+ Garrison platform, the root port's pref window register might be not writable and we have to emulate the window because of hardware defect. In order to detect that, we read the register content, write inversed value and read the register content again. The register is regarded as read-only if the values from the two continuous read are same. However, the original register content isn't written back and it causes corruption on pref window register if it's writable.

  This fixes the above issue by writing the original content back to the register at the end.

5.1.27 skiboot-5.2.3

skiboot-5.2.3 was released on Thursday June 30th, 2016.

skiboot-5.2.3 is the 4th stable release of skiboot 5.2, the new stable release of skiboot, which takes over from the 5.1.x series which was first released August 17th, 2015.

Skiboot 5.2.3 replaces skiboot-5.2.2 as the current stable version, which was released on May 5th, 2016. Over skiboot-5.2.2, skiboot 5.2.3 contains one important bug fix regarding parsing data from the OCC regarding CPU frequency tables, which could lead to no CPU frequency scaling.

skiboot-5.2.3 contains all bug fixes as of skiboot-5.1.16.

This is the second release that will follow the (now documented) Skiboot stable rules - see doc/stable-skiboot-rules.txt.

Over skiboot-5.2.2, the following fixes are included:

OpenPOWER platforms

- occ: Filter out entries from Pmin to Pmax in pstate table (cherry picked from commit eca02ee2e62cee115d921a01cea061782ce47cc7) Without this fix, with newer OCC firmware on some OpenPOWER machines, we would fail to parse the table from the OCC, which meant the host OS would not get a table of supported CPU frequencies.

General

- pci: Do a dummy config write to devices to establish bus number (cherry picked from commit f46c1e506d199332b0f9741278c8ec35b3e39135)

  On PCI Express, devices need to know their own bus number in order to provide the correct source identification (aka RID) in upstream packets they might send, such as error messages or DMAs.

  However while devices know (and hard wire) their own device and function number, they know nothing about bus numbers by default, those are decoded by bridges for routing. All they know is that if their parent bridge sends a "type 0" configuration access, they should decode it provided the device and function numbers match.

  The PCIe spec thus defines that when a device receive such a configuration access and it's a write, it should "capture" the bus number in the source field of the packet, and re-use as the originator bus number of all subsequent outgoing requests.

  In order to ensure that a device has this bus number firmly established before it's likely to send error packets upstream, we should thus do a dummy configuration write to it as soon as possible after probing.
- Fix GCC 6 warning in backtrace code (cherry picked from commit 793f6f5b32c96f2774bd955b6062c74a672317ca)
- Backport of user visible typo fixes partial cherry picked from 4c95b5e04e3c4f72e4005574f67cd6e365d3276f

Utilities

- Fix ARM build failure with parallel make

5.1.28 skiboot-5.2.4

skiboot-5.2.4 was released on Tuesday July 12th, 2016.

This is the 5th stable release of skiboot 5.2, the new stable release of skiboot (first release with 5.2.0 on March 16th 2016).

Skiboot 5.2.4 replaces skiboot-5.2.3 as the current stable version, which was released on June 30th 2016. Over skiboot-5.2.3, skiboot 5.2.4 contains bug fixes to make skiboot more resilient to errors in the XSCOM engine and some build improvements for the pflash utility.

skiboot-5.2.4 contains all bug fixes as of skiboot-5.1.16.

This is the second release that will follow the (now documented) Skiboot stable rules - see doc/stable-skiboot-rules.txt.

Over skiboot-5.2.3, the following fixes are included:

All platforms

- Make the XSCOM engine code more resilient to errors:
  - hw/xscom: Reset XSCOM engine after querying sleeping core FIR
  - hw/xscom: Reset XSCOM engine after finite number of retries when busy

Userspace utilities

- pflash build improvements

5.1.29 skiboot-5.2.5

skiboot-5.2.5 was released on Thursday July 28th, 2016.

skiboot-5.2.5 contains all bug fixes as of skiboot-5.1.17.

This is the second release that will follow the (now documented) Skiboot stable rules - see doc/stable-skiboot-rules.txt.

Over skiboot-5.2.4, the following fixes are included:

- pflash: Fix the makefile (cherry picked from commit fd599965f723330da5ec55519c20cdb6aa2b3a2d)
- pflash: Clean up makefiles and resolve build race (cherry picked from commit c327eddd9b291a0e6e54001fa3b1e547bad3fca2)
- FSP/ELOG: Fix OPAL generated elog resend logic (cherry picked from commit a6d4a7884e95cb9c918b8a217c11e46b01218358)
- FSP/ELOG: Fix possible event notifier hangs (cherry picked from commit e7c8cba4ad773055f390632c2996d3242b633bf4)
- FSP/ELOG: Disable event notification if list is not consistent (cherry picked from commit 1fb10de164d3ca034193df81c1f5d007aec37781)
- FSP/ELOG: Improve elog event states (cherry picked from commit cec5750a4a86ff3f69e1d8817eda023f4d40c492)
- FSP/ELOG: Fix OPAL generated elog event notification (cherry picked from commit ec366ad4e2e871096fa4c614ad7e89f5bb6f884f)
- FSP/ELOG: Disable event notification during kexec (cherry picked from commit d2ae07fd97bb9408456279cec799f72cb78680a6)
- hw/xscom: Reset XSCOM engine after querying sleeping core FIR (cherry picked from commit 15cec493804ff14e6246eb1b65e9d0c7cb469a81)
- hw/xscom: Reset XSCOM engine after finite number of retries when busy (cherry picked from commit e761222593a1ae932cddbc81239b6a7cd98ddb70)
- xscom: Return OPAL_WRONG_STATE on XSCOM ops if CPU is asleep (cherry picked from commit 9c2d82394fd2303847cac4a665dee62556ca528a)

- fsp/console: Ignore data on unresponsive consoles (cherry picked from commit fd6b71fcc6912611ce81f455b4805f0531699d5e)
- SEL: Fix eSEL ID while logging eSEL event

5.1.30 skiboot-5.3.0

skiboot-5.3.0 was released on Tuesday August 2nd, 2016.

skiboot-5.3.0 is the first stable release of skiboot 5.3, the new stable release of skiboot, which will take over from the 5.2.x series which was first released Wednesday March 16th, 2016.

skiboot-5.3.0 contains all bug fixes as of skiboot-5.1.17 and skiboot-5.2.5.

Changes over skiboot-5.3.0-rc2: - Adopt libtool rules for soname versioning for libflash

See skiboot-5.3.0-rc2 and skiboot-5.3.0-rc1 release notes for a complete list of changes from skiboot-5.2.0.

5.1.31 skiboot-5.3.0-rc1

skiboot-5.3.0-rc1 was released on Monday July 25th, 2016

skiboot-5.3.0-rc1 is the first release candidate of skiboot 5.3, which will become the new stable release of skiboot following the 5.2 release, first released March 16th 2016.

skiboot-5.3.0-rc1 contains all bug fixes as of skiboot-5.1.16 and skiboot-5.2.4 (the existing stable releases).

For how the skiboot stable releases work, see doc/stable-skiboot-rules.txt in the skiboot source repository.

The current plan is to release skiboot-5.3.0 August 1st 2016.

Over skiboot-5.2, we have the following changes:

OPAL API/Device Tree

- **Reserve OPAL API numbers for XICS emulation for XIVE** Additionally, we put in some skeleton docs for what's coming, key points being that this is for P9 and above, relies on a device being present in the device tree and is modelled on the PAPR calls.
- interrupts: Remove #interrupt-cells from ICP nodes
- Stop adding legacy linux, phandle to device tree, just add phandle No Linux kernel has ever existed for powernv that only knows linux,phandle.

POWER9

- Add base POWER9 support In *NO WAY* is this geared towards real POWER9 hardware. Suitable for use in simulators *only*, and even then, only if you intensely know what you're doing.
- Document changes in OPAL API for POWER9 Some things are going to change, we start documenting them.
- cpu: supply ibm,dec-bits via devicetree
- power9: Add example device tree for phb4
- device-tree: Only advertise ibm, opal-v3 (not v2) on POWER9 and above

CAPI

- phb3: Test CAPI mode on both CAPP units on Naples
- hmi: Recover both CAPP units on Naples after malfunction alert
- chiptod: Sync timebase in both CAPP units on Naples
- phb3: Set CAPI mode for both CAPP units on Naples
- phb3: Load CAPP ucode to both CAPP units on Naples
- **phb3: Add support for CAPP DMA mode** The XSL used in the Mellanox CX4 card uses a DMA mode of CAPI, which requires a few registers configured specially. This adds a new mode to the OPAL_PCI_SET_PHB_CAPI_MODE API to enable CAPI in DMA mode.

PCI

- pci: Do a dummy config write to devices to establish bus number
- phb: Work around XSL bug sending PTE updates with wrong scope
- Support for PCI hotplug (if a platform supports it)

Garrison

- NVLink/NPU support
- Full garrison platform support.

BMC based platforms

- bt: use the maximum retry count returned by the BMC
- **SEL: Fix eSEL ID while logging eSEL event** Commit 127a7dac added eSEL ID to SEL event in reverse order (0700 instead of 0007). This code fixes this issue by adding ID in proper order.

Tests/Simulation

- test/hello_world: always use shutdown type zero
- make check: make test runs less noisy
- boot-tests: force booting from primary (non-golden) side
- mambo: Enable multicore configurations
- mambo: Flatten device tree at the end
- mambo: Increase memory to 4GB and change memory map
- Timebase quirk for slow simulators like AWAN and SIMICS
- chip: Add simics specific quirks
- mambo: Flash driver using bogus disk
- platform/mambo: Add a heartbeat time, making console more responsive
- mambo: Fix bt command and add little endian support

FSP platforms

- beginnings of support for SPIRA-S structure
- Handle mbox response with bad status:0x24 during FSP termination
- FSP: Validate fsp_msg response memory allocation
- FSP/ELOG: Fix OPAL generated elog event notification
- FSP/ELOG: Disable event notification during kexec Possible crash if error log timing around kexec is unfortunate
- fsp/console: Ignore data on unresponsive consoles

  Linux kernels from v4.1 onwards will try to request an irq for each hvc console using OPAL_EVENT_CONSOLE_INPUT, however because the IRQF_SHARED flag is not set any console after the first will fail. If there is data on one of these failed consoles OPAL will set OPAL_EVENT_CONSOLE_INPUT every time fsp_console_read is called, leading to RCU stalls in the kernel.

  As a workaround for unpatched kernels, cease setting OPAL_EVENT_CONSOLE_INPUT for consoles that we have noticed are not being read.

HMI

- hmi: Fix a bug where partial hmi event was reported to host.
- hmi: Add handling for NPU checkstops
- hmi: Only raise a catchall HMI if no other components have
- hmi: Rework HMI event handling of FIR read failure

Tools

- **external: Add a getsram command** The getsram command reads the OCC SRAM. This is useful for debug.
- bug fixes in flash utilities (pflash/gard)
- pflash: Allow building under yocto.
- external/opal-prd: Ensure that struct host_interfaces matches the thunk
- external/pflash: Handle incorrect cmd-line options better
- libflash: fix bug on reading truncated flash file
- pflash: add support for manipulating file rather than flash
- gard: fix compile error on ARM
- libflash: Add sanity checks to ffs init code.
- external: Add dynamically linked pflash

Mambo

- **Test device tree for kernel location** This can reduce the boot time since the kernel no longer needs to relocate itself when loaded directly at 0.

Generic

- hw/lpc: Log LPC SYNC errors as OPAL_PLATFORM_ERR_EVT errors
- Explicitly disable the attn instruction on all CPUs on boot.
- hw/xscom: Reset XSCOM engine after finite number of retries when busy
- hw/xscom: Reset XSCOM engine after querying sleeping core FIR
- core/timer: Add support for platform specific heartbeat
- Fix GCOV_COUNTERS ifdef logic for GCC 6.0
- core: Fix backtrace for gcc 6 fixes a compiler warning on GCC 6 and above
- **cpu: Don't call time_wait with lock held** Also make the locking around re-init safer, properly block the OS from restarting a thread that was caught for re-init.
- flash: Increase the maximum number of flash devices

Contributors

Extending the analysis done for the last few releases, we can see our trends in code review across versions:

Release	csets	Ack	Reviews	Tested	Reported
5.0	329	15	20	1	0
5.1	372	13	38	1	4
5.2-rc1	334	20	34	6	11
5.3-rc1	302	36	53	4	5

An increase in reviews this cycle is great!

Detailed statistics for 5.3.0-rc1 are below:

Processed 302 csets from 31 developers A total of 20887 lines added, 4540 removed (delta 16347)

Developers with the most changesets

Stewart Smith	82 (27.2%)
Gavin Shan	36 (11.9%)
Benjamin Herrenschmidt	28 (9.3%)
Michael Neuling	25 (8.3%)
Vasant Hegde	24 (7.9%)
Russell Currey	14 (4.6%)
Brad Bishop	12 (4.0%)
Vipin K Parashar	10 (3.3%)
Cédric Le Goater	9 (3.0%)
Shreyas B. Prabhu	8 (2.6%)
Jeremy Kerr	7 (2.3%)
Philippe Bergheaud	6 (2.0%)
Cyril Bur	5 (1.7%)
Mukesh Ojha	4 (1.3%)
Alistair Popple	4 (1.3%)
Ian Munsie	4 (1.3%)
Oliver O'Halloran	3 (1.0%)
Chris Smart	3 (1.0%)

Continued on next page

Table 5.1 – continued from previous page

Sam Mendoza-Jonas	2 (0.7%)
Joel Stanley	2 (0.7%)
Dinar Valeev	2 (0.7%)
Shilpasri G Bhat	2 (0.7%)
Patrick Williams	2 (0.7%)
Deb McLemore	1 (0.3%)
Balbir Singh	1 (0.3%)
Andrew Donnellan	1 (0.3%)
Suraj Jitindar Singh	1 (0.3%)
Frederic Bonnard	1 (0.3%)
Kamalesh Babulal	1 (0.3%)
Mamatha	1 (0.3%)
Mahesh Salgaonkar	1 (0.3%)

Developers with the most changed lines

Benjamin Herrenschmidt	7491 (34.4%)
Gavin Shan	4821 (22.1%)
Vasant Hegde	4740 (21.7%)
Stewart Smith	1294 (5.9%)
Michael Neuling	620 (2.8%)
Cédric Le Goater	470 (2.2%)
Jeremy Kerr	338 (1.6%)
Shreyas B. Prabhu	330 (1.5%)
Vipin K Parashar	305 (1.4%)
Russell Currey	295 (1.4%)
Alistair Popple	229 (1.1%)
Philippe Bergheaud	170 (0.8%)
Ian Munsie	133 (0.6%)
Dinar Valeev	126 (0.6%)
Brad Bishop	80 (0.4%)
Oliver O'Halloran	80 (0.4%)
Cyril Bur	62 (0.3%)
Frederic Bonnard	61 (0.3%)
Sam Mendoza-Jonas	32 (0.1%)
Chris Smart	27 (0.1%)
Shilpasri G Bhat	20 (0.1%)
Patrick Williams	18 (0.1%)
Suraj Jitindar Singh	17 (0.1%)
Mamatha	15 (0.1%)
Mukesh Ojha	8 (0.0%)
Mahesh Salgaonkar	8 (0.0%)
Joel Stanley	4 (0.0%)
Balbir Singh	4 (0.0%)
Kamalesh Babulal	2 (0.0%)
Deb McLemore	1 (0.0%)
Andrew Donnellan	1 (0.0%)

Developers with the most lines removed

Dinar Valeev	68 (1.5%)
Patrick Williams	10 (0.2%)
Mukesh Ojha	4 (0.1%)
Kamalesh Babulal	1 (0.0%)

Developers with the most signoffs (total 249)

Stewart Smith	236 (94.8%)
Vaidyanathan Srinivasan	6 (2.4%)
Benjamin Herrenschmidt	3 (1.2%)
Michael Neuling	2 (0.8%)
Oliver O'Halloran	1 (0.4%)
Vipin K Parashar	1 (0.4%)

Developers with the most reviews (total 53)

Andrew Donnellan	11 (20.8%)
Russell Currey	9 (17.0%)
Joel Stanley	7 (13.2%)
Alistair Popple	7 (13.2%)
Mukesh Ojha	5 (9.4%)
Cyril Bur	3 (5.7%)
Mahesh Salgaonkar	2 (3.8%)
Gavin Shan	2 (3.8%)
Vasant Hegde	2 (3.8%)
Stewart Smith	1 (1.9%)
Vaidyanathan Srinivasan	1 (1.9%)
Vipin K Parashar	1 (1.9%)
Frederic Barrat	1 (1.9%)
Cédric Le Goater	1 (1.9%)

Developers with the most test credits (total 4)

Andrew Donnellan	2 (50.0%)
Russell Currey	1 (25.0%)
Vaibhav Jain	1 (25.0%)

Developers who gave the most tested-by credits (total 4)

Michael Neuling	3 (75.0%)
Gavin Shan	1 (25.0%)

Developers with the most report credits (total 5)

Mukesh Ojha	2 (40.0%)
Russell Currey	1 (20.0%)
Pridhiviraj Paidipeddi	1 (20.0%)
Balbir Singh	1 (20.0%)

Developers who gave the most report credits (total 5)

Gavin Shan	2 (40.0%)
Stewart Smith	2 (40.0%)
Vasant Hegde	1 (20.0%)

5.1.32 skiboot-5.3.0-rc2

skiboot-5.3.0-rc2 was released on Thursday July 28th, 2016.

The current plan is to release skiboot-5.3.0 August 1st 2016.

Over skiboot-5.3.0-rc1, we have the following changes:

pflash

- pflash: Clean up makefiles and resolve build race
- pflash: use atexit for musl compatibility

General

- core/flash: Fix passing pointer instead of value

POWER9

- **mambo: Update Radix Tree Size as per ISA 3.0** In Linux we recently changed to this encoding, so we no longer boot. The associated Linux commit is b23d9c5b9c83c05e013aa52460f12a8365062cf4

FSP Platforms

- platforms/ibm-fsp: Fix incorrect struct member access and comparison
- FSP/MDST: Fix TCE alignment issue In some corner cases (like source memory size = 4097) we may endup doing wrong mapping and corrupting part of SYSDUMP.
- hdat/vpd: Add chip-id property to processor chip node under vpd

CAPI

- hw/phb3: Increase AIB TX command credit for DMA read in CAPP DMA mode

5.1.33 skiboot-5.3.1

skiboot-5.3.1 was released on Wednesday August 10th, 2016.

This is the 2nd stable release of skiboot 5.3, the new stable release of skiboot (first released with 5.3.0 on August 2nd, 2016).

Skiboot 5.3.1 replaces skiboot-5.3.0 as the current stable version. It contains a few minor bug fixes.

This release follows the Skiboot stable rules, see doc/stable-skiboot-rules.txt.

Over skiboot-5.3.0, the following fixes are included:

FSP systems:

- **FSP/ELOG: elog_enable flag should be false by default** This issue is one of the corner case, which is related to recent change went upstream and only observed in the petitboot prompt, where we see only one error log instead of getting all error log in /sys/firmware/opal/elog.

NVLink systems (i.e. Garrison):

- **npu: reword "error" to indicate it's actually a warning** Without this patch, you get spurious FirmWare Test Suite (FWTS) warnings about NVLink not working on machines that aren't fully populated with GPUs.
- **hmi: Clean up NPU FIR debug messages** With the skiboot log set to debug, the FIR (and related registers) were logged all in the same message. It was too much for one line, didn't clarify if the numbers were in hex, and didn't show leading zeroes.

General:

- asm: Fix backtrace for unexpected exception
- correct the log level from PR_ERROR down to PR_INFO for some skiboot log messages.

5.1.34 skiboot-5.3.2

skiboot-5.3.2 was released on Friday August 26th, 2016.

This is the 3rd stable release of skiboot 5.3, the new stable release of skiboot (first released with 5.3.0 on August 2nd, 2016).

Skiboot 5.3.2 replaces skiboot-5.3.1 as the current stable version. It contains a few minor bug fixes.

Over skiboot-5.3.1, the following fixes are included:

- opal/hmi: Fix a TOD HMI failure during a race condition. Rare race condition which meant we wouldn't recover from TOD error
- lpc: Log LPC SYNC errors as unrecoverable ones for manufacturing Only affects systems in manufacturing mode. No behaviour change when not in manufacturing mode.
- hw/phb3: Update capi initialization sequence The capi initialization sequence was revised in a circumvention document when a 'link down' error was converted from fatal to Endpoint Recoverable. Other, non-capi, register setup was corrected even before the initial open-source release of skiboot, but a few capi-related registers were not updated then, so this patch fixes it. The point is that a link-down error detected by the UTL logic will lead to an AIB fence, so that the CAPP unit can detect the error.

5.1.35 skiboot-5.3.3

skiboot-5.3.3 was released on Friday September 2nd, 2016.

This is the 4th stable release of skiboot 5.3, the new stable release of skiboot (first released with 5.3.0 on August 2nd, 2016).

Skiboot 5.3.3 replaces skiboot-5.3.2 as the current stable version. It contains two bug fixes for machines utilizing the NPU (i.e. Garrison)

Over skiboot-5.3.2, the following fixes are included:

- hw/npu: assert the NPU irq min is aligned.
- hw/npu: program NPU BUID reg properly The NPU BUID register was incorrectly programmed resulting in npu interrupt level 0 causing a PB_CENT_CRESP_ADDR_ERROR checkstop, and irqs from npus in odd chips being aliased to and processed as the interrupts from the corresponding npu on the even chips.

5.1.36 skiboot-5.3.4

skiboot-5.3.4 was released on Tuesday September 13th, 2016.

This is the 5th stable release of skiboot 5.3, the new stable release of skiboot (first released with 5.3.0 on August 2nd, 2016).

Skiboot 5.3.4 replaces skiboot-5.3.3 as the current stable version. It contains a couple of bug fixes, specifically around failing XSCOMs.

Over skiboot-5.3.3, the following fixes are included:

- xscom: Initialize the data to a known value in xscom_read In case of error, don't leave the data random. It helps debugging when the user fails to check the error code. This happens due to a bug in the PRD wrapper app.
- xscom: Map all HMER status codes to OPAL errors
- centaur: Mark centaur offline after 10 consecutive access errors This avoids spamming the logs when the centaur is dead and PRD constantly tries to access it
- nvlink: Fix bad PE number check in error inject code path (<= rather than <)

5.1.37 skiboot-5.3.5

skiboot-5.3.5 was released on Wednesday September 14th, 2016.

This is the 6th stable release of skiboot 5.3, the new stable release of skiboot (first released with 5.3.0 on August 2nd, 2016).

Skiboot 5.3.5 replaces skiboot-5.3.4 as the current stable version. It contains a couple of minor bug fixes: simply clarifying two error messages.

Over skiboot-5.3.4, the following fixes are included:

- centaur: print message on disabling xscoms to centaur due to many errors
- slw: improve error message for SLW timer stuck We still register dump, but only to in memory console buffer by default.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search