

BladeCenter to HPI Mapping

April 28, 2006

Table of Contents

| | | |
|-------|---|----|
| 1 | Introduction..... | 4 |
| 1.1 | OpenHPI BladeCenter Plugin Configuration File..... | 4 |
| 1.2 | BladeCenter References..... | 5 |
| 2 | BladeCenter Resources..... | 6 |
| 2.1 | BladeCenter non-Standard Entity Type Definitions..... | 7 |
| 2.2 | Entity Path Examples..... | 8 |
| 2.3 | Virtual Management Module..... | 8 |
| 2.4 | BladeCenter Model Resource Differences..... | 9 |
| 3 | Resource RDR Mappings..... | 10 |
| 3.1 | Chassis RDRs..... | 10 |
| 3.2 | Slot RDRs..... | 11 |
| 3.2.1 | Multi-slot Blades..... | 12 |
| 3.2.2 | Special Sensor Numbers..... | 12 |
| 3.2.3 | Power Module Slot RDRs..... | 12 |
| 3.3 | Management Module RDRs | 12 |
| 3.3.1 | Virtual Management Module RDRs..... | 12 |
| 3.3.2 | Physical Management Module RDRs..... | 13 |
| 3.4 | Blade RDRs..... | 15 |
| 3.5 | Blade Expansion Module RDRs..... | 18 |
| 3.6 | BEM DASD RDRs..... | 19 |
| 3.7 | I/O Module RDRs..... | 19 |
| 3.8 | Power Module RDRs..... | 19 |
| 3.9 | Blower RDRs..... | 20 |
| 3.10 | Media Tray RDRs..... | 21 |
| 4 | Hot Swap Operations..... | 22 |
| 4.1 | Simple Hot Swap..... | 22 |
| 4.2 | Three State Hot Swap..... | 22 |
| 5 | Event Processing..... | 25 |
| 5.1 | BladeCenter Hardware Event Log..... | 25 |
| 5.1.1 | Non-Writable..... | 25 |
| 5.1.2 | Variable Length..... | 25 |
| 5.1.3 | Wrap Characteristics..... | 25 |
| 5.1.4 | Too Much Stuff..... | 26 |
| 5.2 | OpenHPI BladeCenter Resource Event Log..... | 26 |
| 5.2.1 | Initial Bring-up..... | 26 |
| 5.3 | OpenHPI Domain Event Log..... | 27 |
| 5.4 | OpenHPI Domain Alarm Table..... | 27 |

Index of Tables

| | |
|--|----|
| Table 1.1: OpenHPI BladeCenter Plugin Configuration Variables..... | 4 |
| Table 2.1: BladeCenter Resources..... | 7 |
| Table 2.2: BladeCenter non-Standard Entity Path Definitions..... | 7 |
| Table 2.3: BladeCenter Model Resource Differences..... | 9 |
| Table 3.1: Chassis Sensor RDR Summary..... | 10 |
| Table 3.2: Chassis Control RDR Summary..... | 11 |
| Table 3.3: Chassis Inventory RDR Summary..... | 11 |
| Table 3.4: Slot Sensor RDR Summary..... | 11 |
| Table 3.5: VMM Sensor RDR Summary..... | 13 |
| Table 3.6: VMM Control RDR Summary..... | 13 |
| Table 3.7: Physical MM Inventory RDR Summary..... | 14 |
| Table 3.8: Blade Sensor RDR Summary..... | 17 |
| Table 3.9: Blade Control RDR Summary..... | 17 |
| Table 3.10: Blade Inventory RDR Summary..... | 18 |
| Table 3.11: BEM Sensor RDR Summary..... | 18 |
| Table 3.12: BEM DASD Sensor Summary..... | 19 |
| Table 3.13: I/O Module Sensor RDR Summary..... | 19 |
| Table 3.14: I/O Module Inventory RDR Summary..... | 19 |
| Table 3.15: Power Module Sensor RDR Summary..... | 20 |
| Table 3.16: Power Module Inventory RDR Summary..... | 20 |
| Table 3.17: Fan Sensor RDR Summary..... | 21 |
| Table 3.18: Media Tray Sensor RDR Summary..... | 21 |
| Table 3.19: Media Tray Inventory RDR Summary..... | 21 |

1 Introduction

BladeCenter HPI support is through the open source project – OpenHPI. OpenHPI supports HPI with a common set of core infrastructure code and a series of code plugins that are hardware and/or protocol specific. One of these plugins supports the IBM BladeCenter products.

HPI applications may run multiple plugins (e.g. to support ATCA and BladeCenter systems) or multiple instances of the same plugin (e.g. to support multiple BladeCenter chassis). For details on OpenHPI, refer to <http://openhpi.sourceforge.net>.

The OpenHPI BladeCenter plugin code communicates with the BladeCenter's Management Module using SNMP. No matter where the HPI application resides – on a laptop, on a standalone server, or on one of the blades in the BladeCenter itself – an external network path must exist to the BladeCenter's Management Module. The OpenHPI BladeCenter plugin mines the SNMP information available in the BladeCenter's SNMP MIB and translates that information into HPI structures, states, and events.

1.1 OpenHPI BladeCenter Plugin Configuration File

The HPI BladeCenter plugin is configured with the OpenHPI file – ***openhpi/examples/openhpi.conf***. Here's an example of the variables:

| Variable | Default Value | Description |
|-------------------|-----------------------|---|
| entity_root | "{SYSTEM_CHASSIS, 4}" | Defines the last part of each resource's Entity Path. {SAHPI_ENT_ROOT, 0} is assumed at the end of this definition. Applications can run multiple instances of the plugin to support multiple BladeCenters. To keep track of which BladeCenter is being addressed, change the location number (4) to a unique number for each plugin instance definition. |
| host | "192.168.70.125" | BladeCenter Management Module (MM) IP address. |
| version | "1" | SNMP version. SNMP versions 1 and 3 are supported. |
| community | "public" | SNMP v1 community name. Required if using SNMP v1; must match "Community Name" configured in the MM. |
| security_name | "snmpv3_user" | SNMP v3 user name. Required if using SNMP v3; must match one of the "Login IDs" configured in the MM. |
| context_name | "context_name" | SNMP v3 context name. Required if MM's "Context name" field is defined for the SNMP v3 "Login ID". |
| security_level | "authNoPriv" | SNMP v3 security level. noAuthNoPriv, authNoPriv, and authPriv supported. |
| passphrase | "" | SNMP v3 authentication password. Required if using SNMP v3 and security_level is authNoPriv or authPriv; must match "Password" defined in MM for the SNMP v3 "Login ID". |
| auth_type | "MD5" | SNMP v3 authentication password encoding type. Required if using SNMP v3 and security_level is authNoPriv or authPriv. Must match "Authentication Protocol" defined in the MM. Values of MD5, SHA, and None are supported. |
| privacy_passwd | "" | SNMP v3 privacy password. Must match "Privacy password" defined in the MM. |
| privacy_protocol | "None" | SNMP v3 privacy protocol. Must match "Privacy protocol" defined in the MM. Values of None and DES are supported. If security_level is authPriv, code defaults privacy_protocol to DES since there is currently no other privacy protocol choice. |
| count_per_getbulk | "32" | SNMP v3. Optional. SNMP_MSG_GETBULK commands can be used to increase performance. This variable sets the MAX OIDs allowable per MSG_GETBULK command. <ul style="list-style-type: none"> ● A value of "0" disables the use of the SNMP_MSG_GETBULK SNMP command. ● Positive values less than 16 default to "16", since values less than 16 don't necessarily increase performance. ● Too high of a value (> 40) can cause SNMP commands to BladeCenter to timeout. ● Default is "32". |

1.1: OpenHPI BladeCenter Plugin Configuration Variables

1.2 BladeCenter References

- In the spirit of open source, the best documentation source (and generally the most accurate) is the code. When you download the OpenHPI code, BladeCenter plugin code is located in the *openhpi/plugins/snmp_bc* directory. Two of the key files in that directory are:
 - *snmp_bc_resources.c* - describes the Resources and RDRs supported.
 - *snmp_bc_event.map* - describes which BladeCenter events are explicitly mapped to HPI events.
- BladeCenter Management Module code can be downloaded from <http://www.ibm.com>. Click on "Downloads and drivers". When you download the code, you'll also get the BladeCenter's SNMP MIB.
- <http://www-03.ibm.com/systems/bladecenter> contains lots of good BladeCenter information.

2 BladeCenter Resources

Table 2.1: BladeCenter Resources describes an abbreviated RPT table for supported BladeCenter resources.

| Resource | Entity Path | Capabilities | Severity |
|--------------------------------|---|---|-----------------|
| BladeCenter Chassis | {entity_root} | SAHPI_CAPABILITY_CONTROL SAHPI_CAPABILITY_INVENTORY_DATA SAHPI_CAPABILITY_RDR SAHPI_CAPABILITY_RESOURCE SAHPI_CAPABILITY_SENSOR | SAHPI_CRITICAL |
| Virtual Management Module | {SAHPI_ENT_SYS_MGMNT_MODULE, 0} {entity_root} | SAHPI_CAPABILITY_CONTROL SAHPI_CAPABILITY_EVENT_LOG SAHPI_CAPABILITY_RDR SAHPI_CAPABILITY_RESOURCE SAHPI_CAPABILITY_SENSOR | SAHPI_CRITICAL |
| Management Module Slot x | {BLADECENTER_SYS_MGMNT_MODULE_SLOT, x} {entity_root} | SAHPI_CAPABILITY_RDR SAHPI_CAPABILITY_RESOURCE SAHPI_CAPABILITY_SENSOR | SAHPI_CRITICAL |
| Management Module x | {SAHPI_ENT_SYS_MGMNT_MODULE, x} {BLADECENTER_SYS_MGMNT_MODULE_SLOT, x} {entity_root} | SAHPI_CAPABILITY_FRU SAHPI_CAPABILITY_INVENTORY_DATA SAHPI_CAPABILITY_MANAGED_HOTSWAP SAHPI_CAPABILITY_RDR SAHPI_CAPABILITY_RESET SAHPI_CAPABILITY_RESOURCE SAHPI_CAPABILITY_SENSOR | SAHPI_MAJOR |
| Blade Slot x | {SAHPI_ENT_PHYSICAL_SLOT, x} {entity_root} | SAHPI_CAPABILITY_RDR SAHPI_CAPABILITY_RESOURCE SAHPI_CAPABILITY_SENSOR | SAHPI_CRITICAL |
| Blade x | {SAHPI_ENT_SBC_BLADE, x} {SAHPI_ENT_PHYSICAL_SLOT, x} {entity_root} | SAHPI_CAPABILITY_CONTROL SAHPI_CAPABILITY_FRU SAHPI_CAPABILITY_INVENTORY_DATA SAHPI_CAPABILITY_MANAGED_HOTSWAP SAHPI_CAPABILITY_POWER SAHPI_CAPABILITY_RDR SAHPI_CAPABILITY_RESET SAHPI_CAPABILITY_RESOURCE SAHPI_CAPABILITY_SENSOR | SAHPI_MAJOR |
| Blade Expansion Module (BEM) x | {SAHPI_ENT_SYS_EXPANSION_BOARD, x} {SAHPI_ENT_SBC_BLADE, x} {SAHPI_ENT_PHYSICAL_SLOT, x} {entity_root} | SAHPI_CAPABILITY_RDR SAHPI_CAPABILITY_RESOURCE SAHPI_CAPABILITY_SENSOR | SAHPI_MAJOR |
| BEM DASD | {SAHPI_ENT_DISK_DRIVE} {SAHPI_ENT_SYS_EXPANSION_BOARD, x} {SAHPI_ENT_SBC_BLADE, x} {SAHPI_ENT_PHYSICAL_SLOT, x} {entity_root} | SAHPI_CAPABILITY_RDR SAHPI_CAPABILITY_RESOURCE SAHPI_CAPABILITY_SENSOR | SAHPI_MAJOR |
| I/O Module Slot x | {BLADECENTER_INTERCONNECT_SLOT, x} {entity_root} | SAHPI_CAPABILITY_RDR SAHPI_CAPABILITY_RESOURCE SAHPI_CAPABILITY_SENSOR | SAHPI_CRITICAL |

| Resource | Entity Path | Capabilities | Severity |
|---------------------|--|---|-----------------|
| I/O Module x | {SAHPI_ENT_INTERCONNECT, x} {BLADECENTER_INTERCONNECT_SLOT, x} {entity_root} | SAHPI_CAPABILITY_FRU SAHPI_CAPABILITY_INVENTORY_DATA SAHPI_CAPABILITY_MANAGED_HOTSWAP SAHPI_CAPABILITY_POWER SAHPI_CAPABILITY_RDR SAHPI_CAPABILITY_RESET SAHPI_CAPABILITY_RESOURCE SAHPI_CAPABILITY_SENSOR | SAHPI_MAJOR |
| Power Module Slot x | {BLADECENTER_POWER_SUPPLY_SLOT, x} {entity_root} | SAHPI_CAPABILITY_RDR SAHPI_CAPABILITY_RESOURCE SAHPI_CAPABILITY_SENSOR | SAHPI_CRITICAL |
| Power Module x | {SAHPI_ENT_POWER_SUPPLY, x} {BLADECENTER_POWER_SUPPLY_SLOT, x} {entity_root} | SAHPI_CAPABILITY_FRU SAHPI_CAPABILITY_INVENTORY_DATA SAHPI_CAPABILITY_RDR SAHPI_CAPABILITY_RESOURCE SAHPI_CAPABILITY_SENSOR | SAHPI_MAJOR |
| Blower Slot x | {BLADECENTER_FAN_SLOT, x} {entity_root} | SAHPI_CAPABILITY_RDR SAHPI_CAPABILITY_RESOURCE SAHPI_CAPABILITY_SENSOR | SAHPI_CRITICAL |
| Blower x | {SAHPI_ENT_FAN, x} {BLADECENTER_FAN_SLOT, x} {entity_root} | SAHPI_CAPABILITY_FRU SAHPI_CAPABILITY_RDR SAHPI_CAPABILITY_RESOURCE SAHPI_CAPABILITY_SENSOR | SAHPI_MAJOR |
| Media Tray Slot x | {BLADECENTER_PERIPHERAL_BAY_SLOT, x} {entity_root} | SAHPI_CAPABILITY_RDR SAHPI_CAPABILITY_RESOURCE | SAHPI_CRITICAL |
| Media Tray x | {SAHPI_ENT_PERIPHERAL_BAY, x} {BLADECENTER_PERIPHERAL_BAY_SLOT, x} {entity_root} | SAHPI_CAPABILITY_FRU SAHPI_CAPABILITY_INVENTORY_DATA SAHPI_CAPABILITY_RDR SAHPI_CAPABILITY_RESOURCE SAHPI_CAPABILITY_SENSOR | SAHPI_MAJOR |

Table 2.1: BladeCenter Resources

2.1 BladeCenter non-Standard Entity Type Definitions

Unfortunately, the HPI B.01.01 standard does not support entity types for different types of hardware slots. To support various slot types, the OpenHPI BladeCenter plugin supports the following entity path extensions:

| BladeCenter non-Standard Entity Path Definition | Value |
|--|-----------------------------------|
| BLADECENTER_INTERCONNECT_SLOT | SAHPI_ENT_CHASSIS_SPECIFIC + 0x10 |
| BLADECENTER_POWER_SUPPLY_SLOT | SAHPI_ENT_CHASSIS_SPECIFIC + 0x11 |
| BLADECENTER_PERIPHERAL_BAY_SLOT | SAHPI_ENT_CHASSIS_SPECIFIC + 0x12 |
| BLADECENTER_SYS_MGMT_MODULE_SLOT | SAHPI_ENT_CHASSIS_SPECIFIC + 0x13 |
| BLADECENTER_BLOWER_SLOT | SAHPI_ENT_CHASSIS_SPECIFIC + 0x14 |

Table 2.2: BladeCenter non-Standard Entity Path Definitions

These extensions are defined in the OpenHPI file **SaHpiBladeCenter.h**.

2.2 Entity Path Examples

{entity_root} is defined in OpenHPI's *openhpi.conf* file. For example, if entity_root = "{SYSTEM_CHASSIS, 2}" in *openhpi.conf*, then {entity_root} is:

```
{SAHPI_ENT_SYSTEM_CHASSIS, 2}  
{SAHPI_ENT_ROOT, 0}
```

For the following examples, assume entity_root = "{SYSTEM_CHASSIS, 1}" is defined in *openhpi.conf*.

Example 1: Blade in slot 3

```
{SAHPI_ENT_SBC_BLADE, 3}  
{SAHPI_ENT_PHYSICAL_SLOT, 3}  
{SAHPI_ENT_SYSTEM_CHASSIS, 1}  
{SAHPI_ENT_ROOT, 0}
```

Example 2: SCSI Blade Expansion Module attached to blade in slot 3

```
{SAHPI_ENT_SYS_EXPANSION_BOARD, 3}  
{SAHPI_ENT_SBC_BLADE, 3}  
{SAHPI_ENT_PHYSICAL_SLOT, 3}  
{SAHPI_ENT_SYSTEM_CHASSIS, 1}  
{SAHPI_ENT_ROOT, 0}
```

Example 3: Management Module 2

```
{SAHPI_ENT_SYS_MGMNT_MODULE, 2}  
{BLADECENTER_SYS_MGMNT_MODULE_SLOT, 2}  
{SAHPI_ENT_SYSTEM_CHASSIS, 1}  
{SAHPI_ENT_ROOT, 0}
```

Notes:

- All HPI location values start at 1; except for the Virtual Management Module resource, which has an HPI location value of 0.
- HPI location numbers for the slot and the entity plugged into the slot are always equal.
- Although BladeCenter Blade Expansion Modules (BEMs) take up a blade slot themselves, a BEM's HPI location number always agrees with the location number of its host blade.

2.3 Virtual Management Module

BladeCenter supports two dedicated Management Modules (MMs) which serve as the management control point for the entire chassis. The Management Modules plug into their own slots (they don't take up a blade slot) and are hot swappable. There must be at least one Management Module in the chassis for proper operations. The second MM is optional.

When the second MM is installed, it behaves as an active standby. The two Management Modules ping each other over an internal bus. If the primary MM fails, the standby takes over. With the exception of a few events, the takeover is completely automatic and transparent.

To model this behavior, BladeCenter supports a Virtual Management Module resource (VMM). It

is distinguished by an HPI Entity Path location code of 0 (all physical resources have location codes starting with 1). The primary Management Module monitors the BladeCenter Event Log and several sensors. These are preserved over an MM failover and are associated with the VMM resource.

RDRs specific to an MM, like inventory, are associated with the individual physical MM resource.

2.4 BladeCenter Model Resource Differences

The OpenHPI BladeCenter plugin dynamically determines what BladeCenter model it's communicating with and creates the necessary HPI entities. There isn't anything the HPI application need do or configure for this to happen. But just for fun, here's the major resource differences between the various BladeCenter models:

| Resource | BladeCenter | BladeCenter H | BladeCenter T |
|-----------------|--------------------|----------------------|----------------------|
| Blades | 14 | 14 | 8 |
| I/O Modules | 4 | 10 | 4 |
| Blowers | 2 | 2 | 4 |

Table 2.3: BladeCenter Model Resource Differences

3 Resource RDR Mappings

A couple of general BladeCenter RDR observations:

- HPI applications cannot disable individual sensors (e.g. `SensorRec.EnableCtrl = SAHPI_FALSE` for all sensors).
- HPI applications cannot disable individual events for a given sensor (e.g. `SensorRec.EventCtrl = SAHPI_SEC_READ_ONLY` for all sensors).
- HPI applications cannot set thresholds (e.g. `SensorRec.ThresholdDefn.WriteThold = 0` for all threshold sensors).
- HPI applications cannot write inventory information. `SA_ERR_HPI_READ_ONLY` is returned for `saHpilDrAreaAdd()`, `saHpilDrAreaDelete()`, `saHpilDrFieldAdd()`, and `saHpilDrFieldDelete()`.
- To assist HPI applications in identifying individual voltage sensors, the Nominal value is defined to be the same as the voltage type (e.g. The 3.3 voltage sensor has a Nominal value of 3.3); else the HPI application needs to parse the RDR's IdString.
- For a complete description of how all the RDR fields are defined, refer to the OpenHPI file *snmp_bc_resources.c*.

3.1 Chassis RDRs

| Sensor | Type | Category | Events | Data Format |
|--|--------------------------------|---------------------|--|------------------------|
| Ambient Air Temperature | SAHPI_TEMPERATURE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_CRIT SAHPI_ES_UPPER_MAJOR | FLOAT64 (Degrees C) |
| I/O Module Redundancy | SAHPI_PLATFORM_ALERT | SAHPI_EC_REDUNDANCY | SAHPI_ES_FULLY_REDUNDANCY SAHPI_ES_REDUNDANT_LOST | Event only |
| Power Module Redundancy | SAHPI_PLATFORM_ALERT | SAHPI_EC_REDUNDANCY | SAHPI_ES_FULLY_REDUNDANCY SAHPI_ES_REDUNDANT_LOST | Event only |
| Power Domain 1 Redundancy | SAHPI_PLATFORM_ALERT | SAHPI_EC_REDUNDANCY | SAHPI_ES_FULLY_REDUNDANCY SAHPI_ES_REDUNDANT_LOST | Event only |
| Power Domain 2 Redundancy | SAHPI_PLATFORM_ALERT | SAHPI_EC_REDUNDANCY | SAHPI_ES_FULLY_REDUNDANCY SAHPI_ES_REDUNDANT_LOST | Event only |
| Chassis Total Maximum Power Capability | SAHPI_OTHER_UNITS_BASED_SENSOR | SAHPI_EC_THRESHOLD | None | UINT64 (Watts) |
| Chassis Total Assigned Power | SAHPI_OTHER_UNITS_BASED_SENSOR | SAHPI_EC_THRESHOLD | None | UINT64 (Watts) |
| Chassis Total Minimum Power Capability | SAHPI_OTHER_UNITS_BASED_SENSOR | SAHPI_EC_THRESHOLD | None | UINT64 (Watts) |
| Chassis Filter (BCT only) | SAHPI_ENT_TEMPERATURE | SAHPI_EC_SEVERITY | SAHPI_ES_OK SAHPI_ES_INFORMATIONAL SAHPI_ES_MINOR_FROM_OK SAHPI_ES_MAJOR_FROM_LESS SAHPI_ES_CRITICAL | Event only |

Table 3.1: Chassis Sensor RDR Summary

| Control | Output Type | Type | Comments |
|--|--------------------|--------------------------|---------------------------------|
| Chassis Location LED (Writable Blue LED to illuminate chassis location) | SAHPI_CTRL_LED | SAHPI_CTRL_TYPE_DISCRETE | 0= Off; 1= solid on; 2=blinking |

Table 3.2: Chassis Control RDR Summary

| Inventory | Inventory Area Type | Supported Fields |
|-------------------|---------------------------------|---|
| Chassis Inventory | SAHPI_IDR AREATYPE_CHASSIS_INFO | SAHPI_IDR_FIELDTYPE_CHASSIS_TYPE SAHPI_IDR_FIELDTYPE_MFG_DATETIME SAHPI_IDR_FIELDTYPE_MANUFACTURER SAHPI_IDR_FIELDTYPE_PRODUCT_NAME SAHPI_IDR_FIELDTYPE_PRODUCT_VERSION SAHPI_IDR_FIELDTYPE_SERIAL_NUMBER SAHPI_IDR_FIELDTYPE_PART_NUMBER |

Table 3.3: Chassis Inventory RDR Summary

3.2 Slot RDRs

The Slot State sensor is a presence sensor which allows the HPI application to sense if there is something plugged into a physical slot. If there is a resource in the slot, the Slot State sensor returns an event state of SAHPI_ES_PRESENT and the Resource ID (RID) of the occupying resource. The RID is returned in the least significant 32 bits of an SAHPI_SENSOR_READING_TYPE_UINT64 value. The most significant 32 bits of the UINT64 value are set to 0. If there is no resource in the slot, an event state of SAHPI_ES_ABSENT and a RID of SAHPI_UNSPECIFIED_RESOURCE_ID is returned.

The Maximum Power Capability sensor reports the maximum power that the resource plugged into the slot could consume if it was fully populated (CPUs, memory chips, expansion cards, etc). The Assigned Power Capability sensor reports the power currently being supplied to the slot for its resource. The Minimum Power Capability sensor reports the minimum power that can be allocated to the slot's resource based on the resource's maximum CPU throttling capabilities.

| Sensor | Sensor Number | Type | Category | Events | Data Format |
|-------------------------------|---------------------------------------|--------------------------------|--------------------|---------------|--------------------|
| Slot State | BLADECENTER_SENSOR_NUM_SLOT_STATE | SAHPI_ENTITY_PRESENCE | SAHPI_EC_PRESENCE | None | UINT64 (RID) |
| Slot Maximum Power Capability | BLADECENTER_SENSOR_NUM_MAX_POWER | SAHPI_OTHER_UNITS_BASED_SENSOR | SAHPI_EC_THRESHOLD | None | UINT64 (Watts) |
| Slot Assigned Power | BLADECENTER_SENSOR_NUM_ASSIGNED_POWER | SAHPI_OTHER_UNITS_BASED_SENSOR | SAHPI_EC_THRESHOLD | None | UINT64 (Watts) |
| Slot Minimum Power Capability | BLADECENTER_SENSOR_NUM_MIN_POWER | SAHPI_OTHER_UNITS_BASED_SENSOR | SAHPI_EC_THRESHOLD | None | UINT64 (Watts) |

Table 3.4: Slot Sensor RDR Summary

3.2.1 Multi-slot Blades

In BladeCenter, blades may span multiple slots. In this case, each Slot State sensor in the slot span reports that the slot is occupied and returns the same RID. Blades are the only resource that span multiple slots.

For multi-slot blades, the Maximum, Assigned, and Minimum Power Capability values are spread evenly among the slots.

3.2.2 Special Sensor Numbers

The sensor numbers for Slot RDRs are hard coded and defined in the OpenHPI file ***SaHpiBladeCenter.h***. These sensor numbers are the same as those defined in the *HPI-to-AdvancedTCA Mapping Specification* (which defines similar sensors). The attempt is to provide the HPI application that needs to support both BladeCenter and ATCA chassis with as consistent an interface as possible.

3.2.3 Power Module Slot RDRs

Generally the Power Module Slot resource reports 0 for the Power Module's Slot Maximum, Assigned, and Minimum Power Capability Sensors. In some BladeCenter models, the power modules themselves consume power. For example, the BladeCenter H power modules contain fan packs which cool the power modules. These fan packs draw current. The amount of which is reported by the Power Module's Slot Maximum, Assigned, Minimum sensors.

3.3 Management Module RDRs

3.3.1 Virtual Management Module RDRs

The VMM Redundancy sensor is an event only sensor that notifies the HPI application that MM redundancy has been lost or recovered.

The Active MM and Standby MM sensors returns and event state of SAHPI_ES_PRESENT and the Resource ID of the physical MM that serves as the active or standby MM. The RID is returned in the least significant 32 bits of an SAHPI_SENSOR_READING_TYPE_UINT64 value. The most significant 32 bits of the UINT64 value are set to 0. If there is no standby MM, the Standby MM sensor returns SAHPI_UNSPECIFIED_RESOURCE_ID and an event state of SAHPI_ES_ABSENT.

The MM Air Temperature sensor reads the temperature as recorded at the primary MM.

A variety of sensors monitor the voltage distribution coming from the power modules and being distributed in the chassis to the various hardware components.

A variety of sensors monitor internal chassis I2C buses used to determine the presence and status of various hardware components. Events from these sensors don't always mean that the component has failed; just that the primary MM cannot access it.

A control is provided to allow the HPI application to force a MM failover. Setting a value of SAHPI_CTRL_STATE_PULSE_ON forces the failover. This control returns SA_ERR_HPI_INVALID_REQUEST if there are not redundant MMs in the chassis. SAHPI_CTRL_STATE_OFF is always returned if the control's state is read.

| Sensor | Type | Category | Events | Data Format |
|---|--------------------------------|-----------------------|---|--------------------|
| System Management Bus Operational State | SAHPI_OPERATIONAL | SAHPI_EC_AVAILABILITY | SAHPI_ES_RUNNING SAHPI_ES_DEGRADED SAHPI_ES_OFF_LINE | Event Only |
| MM Redundancy | SAHPI_OPERATIONAL | SAHPI_EC_REDUNDANCY | SAHPI_ES_FULLY_REDUNDANT SAHPI_ES_NON_REDUNDANT_SUFFICIENT_RESOURCES | Event only |
| Active MM | SAHPI_ENTITY_PRESENCE | SAHPI_EC_PRESENCE | None | UINT64 (RID) |
| Standby MM | SAHPI_ENTITY_PRESENCE | SAHPI_EC_PRESENCE | None | UINT64 (RID) |
| MM Air Temperature | SAHPI_TEMPERATURE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR | UINT64 (Degrees C) |
| System 1.8 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_CRIT SAHPI_ES_LOWER_CRIT | FLOAT64 (Volts) |
| System 2.5 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_CRIT SAHPI_ES_LOWER_CRIT | FLOAT64 (Volts) |
| System 3.3 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_CRIT SAHPI_ES_LOWER_CRIT | FLOAT64 (Volts) |
| System 5 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_CRIT SAHPI_ES_LOWER_CRIT | FLOAT64 (Volts) |
| System -5 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_CRIT SAHPI_ES_LOWER_CRIT | FLOAT64 (Volts) |
| System 12 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_CRIT SAHPI_ES_LOWER_CRIT | FLOAT64 (Volts) |
| Midplane Maximum Power Capability | SAHPI_OTHER_UNITS_BASED_SENSOR | SAHPI_EC_THRESHOLD | None | UINT64 (Watts) |
| Midplane Assigned Power | SAHPI_OTHER_UNITS_BASED_SENSOR | SAHPI_EC_THRESHOLD | None | UINT64 (Watts) |
| Midplane Minimum Power Capability | SAHPI_OTHER_UNITS_BASED_SENSOR | SAHPI_EC_THRESHOLD | None | UINT64 (Watts) |

Table 3.5: VMM Sensor RDR Summary

| Control | Output Type | Type | Comments |
|----------------|--------------------|-------------------------|---|
| MM Failover | SAHPI_CTRL_GENERIC | SAHPI_CTRL_TYPE_DIGITAL | SAHPI_CTRL_STATE_PULSE_ON Forces MM failover |

Table 3.6: VMM Control RDR Summary

3.3.2 Physical Management Module RDRs

| <i>Inventory</i> | <i>Inventory Area Type</i> | <i>Supported Fields</i> |
|-------------------------|-----------------------------------|--|
| MM Inventory | SAHPI_IDR_AREATYPE_BOARD_INFO | SAHPI_IDR_FIELDTYPE_MFG_DATETIME SAHPI_IDR_FIELDTYPE_MANUFACTURER SAHPI_IDR_FIELDTYPE_PRODUCT_VERSION SAHPI_IDR_FIELDTYPE_PART_NUMBER |

Table 3.7: Physical MM Inventory RDR Summary

3.4 Blade RDRs

| Sensor | Type | Category | Events | Data Format |
|--|--------------------------|-----------------------|--|---|
| Blade Operational Status | SAHPI_OPERATIONAL | SAHPI_EC_AVAILABILITY | SAHPI_ES_RUNNING SAHPI_ES_OFF_LINE SAHPI_ES_DEGRADED SAHPI_ES_INSTALL_ERROR | INT64 0 = unknown 1 = good 2 = warning 3 = bad 4 = kernelmode 5 = discovering 6 = comm error 7 = no power 8 = flashing 9 = init error |
| Blade Management Bus Operational State | SAHPI_OPERATIONAL | SAHPI_EC_AVAILABILITY | SAHPI_ES_RUNNING SAHPI_ES_OFF_LINE | Event Only |
| Blade NMI Status | SAHPI_CRITICAL_INTERRUPT | SAHPI_EC_STATE | SAHPI_ES_STATE_ASSERTED SAHPI_ES_STATE_DEASSERTED | Event Only |
| Blade CPU 1 Temperature | SAHPI_TEMPERATURE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_CRIT SAHPI_ES_UPPER_MAJOR | FLOAT64 (Degrees C) |
| Blade CPU 2 Temperature | SAHPI_TEMPERATURE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_CRIT SAHPI_ES_UPPER_MAJOR | FLOAT64 (Degrees C) |
| Blade CPU 3 Temperature | SAHPI_TEMPERATURE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_CRIT SAHPI_ES_UPPER_MAJOR | FLOAT64 (Degrees C) |
| Blade CPU 4 Temperature | SAHPI_TEMPERATURE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_CRIT SAHPI_ES_UPPER_MAJOR | FLOAT64 (Degrees C) |
| Blade CPU 1 Core Voltage | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| Blade CPU 2 Core Voltage | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| Blade CPU 3 Core Voltage | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| Blade CPU 4 Core Voltage | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| Blade VRM Voltage | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| Blade 0.9 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| Blade 1.2 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| Blade Standby 1.2 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| Blade 1.25 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| Blade 1.5 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| Blade Standby 1.5 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| Blade 1.8 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |

| Sensor | Type | Category | Events | Data Format |
|---------------------------------|-------------------|--------------------|--|--------------------|
| Blade Standby 1.8 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| Blade 2.5 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| Blade Standby 2.5 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| Blade 3.3 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| Blade Standby 3.3 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| Blade 5 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| Blade Standby 5 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| Blade -5 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| Blade 12 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| Blade Standby 12 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| Blade Memory Bank 1 Temperature | SAHPI_TEMPERATURE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_CRIT SAHPI_ES_UPPER_MAJOR | FLOAT64 (Volts) |
| Blade Memory Bank 2 Temperature | SAHPI_TEMPERATURE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_CRIT SAHPI_ES_UPPER_MAJOR | FLOAT64 (Volts) |
| Blade Memory Bank 3 Temperature | SAHPI_TEMPERATURE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_CRIT SAHPI_ES_UPPER_MAJOR | FLOAT64 (Volts) |
| Blade Memory Bank 4 Temperature | SAHPI_TEMPERATURE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_CRIT SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| Blade Battery Voltage | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_CRIT SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |

Table 3.8: Blade Sensor RDR Summary

| Control | Output Type | Type | Comments |
|--|--------------------|--------------------------|-------------------------------------|
| Blade Location LED (Writable Blue LED to illuminate blade location) | SAHPI_CTRL_LED | SAHPI_CTRL_TYPE_DISCRETE | 0 = Off; 1 = solid on; 2 = blinking |
| Blade BMC Reset | SAHPI_CTRL_GENERIC | SAHPI_CTRL_TYPE_DISCRETE | 1 = reset BMC |

Table 3.9: Blade Control RDR Summary

| Inventory | Inventory Area Type | Supported Fields |
|------------------|-------------------------------|---|
| Blade Inventory | SAHPI_IDR_AREATYPE_BOARD_INFO | SAHPI_IDR_FIELDTYPE_MFG_DATETIME SAHPI_IDR_FIELDTYPE_MANUFACTURER SAHPI_IDR_FIELDTYPE_PRODUCT_NAME SAHPI_IDR_FIELDTYPE_PRODUCT_VERSION SAHPI_IDR_FIELDTYPE_SERIAL_NUMBER SAHPI_IDR_FIELDTYPE_PART_NUMBER |

Table 3.10: Blade Inventory RDR Summary

3.5 Blade Expansion Module RDRs

| Sensor | Type | Category | Events | Data Format |
|------------------------|-------------------|-----------------------|--|------------------------|
| BEM Operational Status | SAHPI_OPERATIONAL | SAHPI_EC_AVAILABILITY | SAHPI_ES_RUNNING SAHPI_ES_OFF_LINE | Event Only |
| BEM Temperature | SAHPI_TEMPERATURE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_CRIT SAHPI_ES_UPPER_MAJOR | Event Only |
| BEM Voltage | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | Event Only |
| PEU2 Temperature | SAHPI_TEMPERATURE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_CRIT SAHPI_ES_UPPER_MAJOR | FLOAT64 (Degrees C) |
| PEU2 1 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| PEU2 3.3 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| PEU2 5 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| PEU2 12 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| PEU2 Standby 12 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| BIE Temperature | SAHPI_TEMPERATURE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_CRIT SAHPI_ES_UPPER_MAJOR | FLOAT64 (Degrees C) |
| BIE 1.5 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| BIE 3.3 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| BIE2 5 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |
| PEU 12 Volt | SAHPI_VOLTAGE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_LOWER_MAJOR | FLOAT64 (Volts) |

Table 3.11: BEM Sensor RDR Summary

3.6 BEM DASD RDRs

| Sensor | Type | Category | Events | Data Format |
|-------------------------------|-------------------|-----------------------|---------------------------------------|--------------------|
| BEM DASD 1 Operational Status | SAHPI_OPERATIONAL | SAHPI_EC_AVAILABILITY | SAHPI_ES_RUNNING SAHPI_ES_OFF_LINE | Event Only |
| BEM DASD 2 Operational Status | SAHPI_OPERATIONAL | SAHPI_EC_AVAILABILITY | SAHPI_ES_RUNNING SAHPI_ES_OFF_LINE | Event Only |
| BEM DASD 3 Operational Status | SAHPI_OPERATIONAL | SAHPI_EC_AVAILABILITY | SAHPI_ES_RUNNING SAHPI_ES_OFF_LINE | Event Only |

Table 3.12: BEM DASD Sensor Summary

3.7 I/O Module RDRs

| Sensor | Type | Category | Events | Data Format |
|---|-------------------|-----------------------|---|--|
| I/O Module Operational Status | SAHPI_OPERATIONAL | SAHPI_EC_AVAILABILITY | SAHPI_ES_RUNNING SAHPI_ES_OFF_LINE | INT64 0 = unknown 1 = good 2 = warning 3 = bad |
| I/O Module Management Bus Operational State | SAHPI_OPERATIONAL | SAHPI_EC_AVAILABILITY | SAHPI_ES_RUNNING SAHPI_ES_OFF_LINE | Event Only |
| I/O Module Temperature | SAHPI_TEMPERATURE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_CRIT SAHPI_ES_UPPER_MAJOR | Event Only |

Table 3.13: I/O Module Sensor RDR Summary

| Inventory | Inventory Area Type | Supported Fields |
|----------------------|-------------------------------|--|
| I/O Module Inventory | SAHPI_IDR_AREATYPE_BOARD_INFO | SAHPI_IDR_FIELDTYPE_MFG_DATETIME SAHPI_IDR_FIELDTYPE_MANUFACTURER SAHPI_IDR_FIELDTYPE_PRODUCT_VERSION SAHPI_IDR_FIELDTYPE_PART_NUMBER |

Table 3.14: I/O Module Inventory RDR Summary

3.8 Power Module RDRs

| Sensor | Type | Category | Events | Data Format |
|---|-------------------|-----------------------|---|--|
| Power Module Operational Status | SAHPI_OPERATIONAL | SAHPI_EC_AVAILABILITY | SAHPI_ES_RUNNING SAHPI_ES_OFF_LINE | INT64 0 = unknown 1 = good 2 = warning 3 = bad |
| Power Module Management Bus Operational State | SAHPI_OPERATIONAL | SAHPI_EC_AVAILABILITY | SAHPI_ES_RUNNING SAHPI_ES_OFF_LINE | Event Only |
| Power Module Temperature | SAHPI_TEMPERATURE | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR SAHPI_ES_UPPER_CRIT | Event Only |
| Power Module Fan Pack Operational Status | SAHPI_OPERATIONAL | SAHPI_EC_AVAILABILITY | SAHPI_ES_RUNNING SAHPI_ES_OFF_LINE | Event Only |
| Power Module Fan Pack Average Speed (Percent of Max) (BCH only) | SAHPI_FAN | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR | PERCENTAGE |
| Power Module Fan Pack Average RPM Speed (BCH only) | SAHPI_FAN | SAHPI_EC_UNSPECIFIED | None | FLOAT64 (RPM) |

Table 3.15: Power Module Sensor RDR Summary

| Inventory | Inventory Area Type | Supported Fields |
|------------------------|-------------------------------|--|
| Power Module Inventory | SAHPI_IDR_AREATYPE_BOARD_INFO | SAHPI_IDR_FIELDTYPE_MFG_DATETIME SAHPI_IDR_FIELDTYPE_MANUFACTURER SAHPI_IDR_FIELDTYPE_PRODUCT_VERSION SAHPI_IDR_FIELDTYPE_PART_NUMBER |

Table 3.16: Power Module Inventory RDR Summary

3.9 Blower RDRs

| Sensor | Type | Category | Events | Data Format |
|---|-------------------|-----------------------|---------------------------------------|--|
| Blower Operational Status | SAHPI_OPERATIONAL | SAHPI_EC_AVAILABILITY | SAHPI_ES_RUNNING SAHPI_ES_OFF_LINE | INT64 0 = unknown 1 = good 2 = warning 3 = bad |
| Blower Management Bus Operational State | SAHPI_OPERATIONAL | SAHPI_EC_AVAILABILITY | SAHPI_ES_RUNNING SAHPI_ES_OFF_LINE | Event Only |
| Blower Speed (Percent of Max) | SAHPI_FAN | SAHPI_EC_THRESHOLD | SAHPI_ES_UPPER_MAJOR | PERCENTAGE |
| Blower RPM Speed (BCH only) | SAHPI_FAN | SAHPI_EC_UNSPECIFIED | None | FLOAT64 (RPM) |

Table 3.17: Fan Sensor RDR Summary

3.10 Media Tray RDRs

| Sensor | Type | Category | Events | Data Format |
|---|-------------------|-----------------------|---------------------------------------|--------------------|
| Media Tray Operational Status | SAHPI_OPERATIONAL | SAHPI_EC_AVAILABILITY | SAHPI_ES_RUNNING SAHPI_ES_DEGRADED | Event Only |
| Media Tray Management Bus Operational State | SAHPI_OPERATIONAL | SAHPI_EC_AVAILABILITY | SAHPI_ES_RUNNING SAHPI_ES_OFF_LINE | Event Only |

Table 3.18: Media Tray Sensor RDR Summary

| Inventory | Inventory Area Type | Supported Fields |
|----------------------|-------------------------------|--|
| Media Tray Inventory | SAHPI_IDR_AREATYPE_BOARD_INFO | SAHPI_IDR_FIELDTYPE_MFG_DATETIME SAHPI_IDR_FIELDTYPE_MANUFACTURER SAHPI_IDR_FIELDTYPE_PRODUCT_VERSION SAHPI_IDR_FIELDTYPE_PART_NUMBER |

Table 3.19: Media Tray Inventory RDR Summary

4 Hot Swap Operations

BladeCenter hardware currently supports two hot swap models.

- Simple “Not present or Active” model
- Three State “Not present, Inactive, Active” model

4.1 Simple Hot Swap

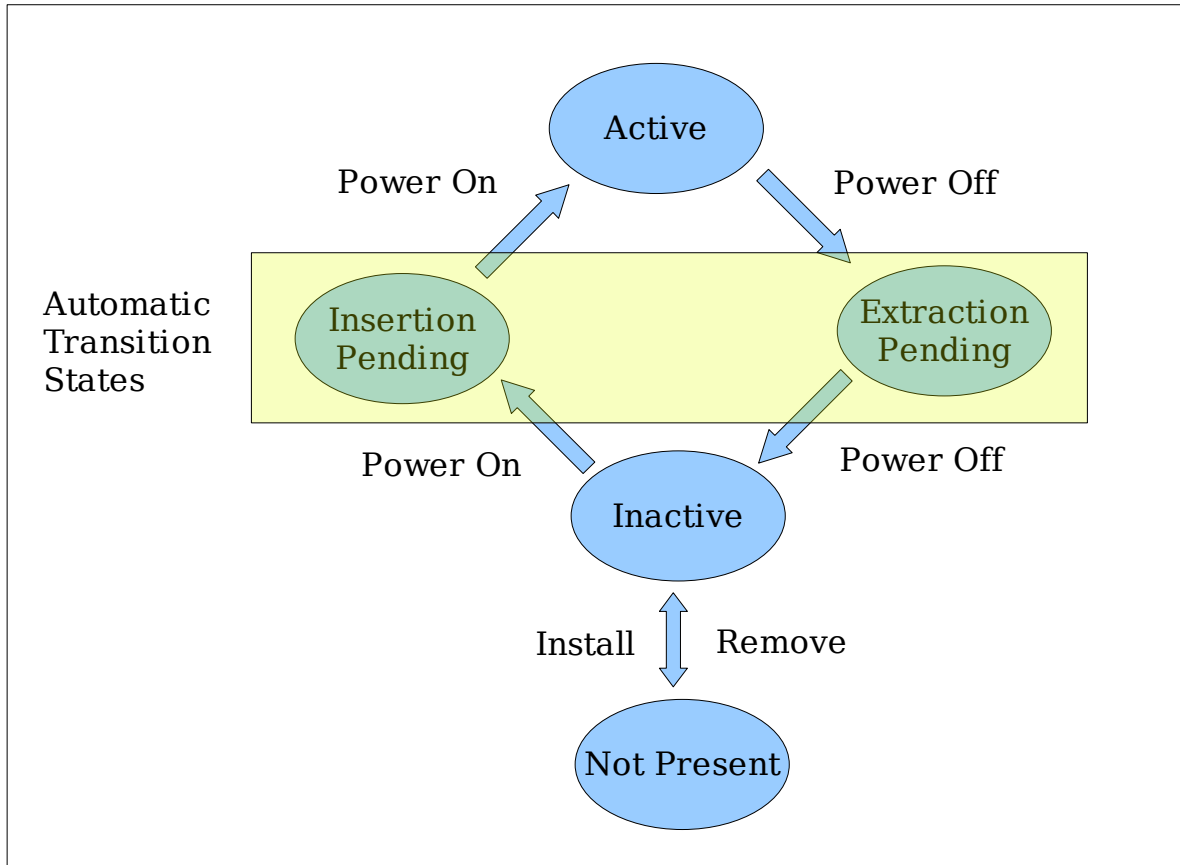
Resources such as fans, power supplies, and the media tray become active as soon as they are installed into the system. There is no HPI application action required to activate these resources. Nor is there any action required to make these resources ready to be removed. These resources are modeled using HPI's Simplified Hot Swap model. HPI hot swap events are generated whenever the resource is removed from or installed into the system.

Resources supporting HPI's Simplified Hot Swap model are distinguished in Table 2.1: BladeCenter Resources by a Capabilities definition that includes SAHPI_CAPABILITY_FRU but does **not** include SAHPI_CAPABILITY_MANAGED_HOTSWAP.

4.2 Three State Hot Swap

Resource such as blades, I/O modules, and Management Modules go through a three stage hot swap procedure – Not Present, Inactive, Active. When one of these resources is inserted into the chassis, it enters the Inactive state. In this state, the resource is in communication with the primary Management Module and has run self-tests to verify that it is ready for normal operations. It now takes an explicit action to activate the resource – either by pressing a local power-on hardware button or by issuing a remote power-on command. Likewise the resource can be power-off either locally or remotely to move the resource from the Active to the Inactive state. The resource can also be reset remotely – which acts like a power off/power on sequence.

So how to model this in HPI? HPI's Simplified Hot Swap model is too simple; while HPI's 5-state Managed Hot Swap model is too complex. To give HPI applications control over transitions between the Active and Inactive state, these resources are modeled using HPI's Managed Hot Swap model. The difference is that transitions from the Inactive state to the Active state and vice versa pass automatically through the HPI “Insertion Pending” and “Extraction Pending” states. See the figure below:



For example, when a blade is inserted into the chassis, it enters into the HPI Inactive state. An informational HPI hot swap event flows to the HPI application marking the transition from SAHPI_HS_STATE_NOT_PRESENT to SAHPI_HS_STATE_INACTIVE. If the application then issues saHpiResourcePowerStateSet() to the blade, it will receive an informational hot swap event marking the transition from SAHPI_HS_STATE_INACTIVE to SAHPI_HS_STATE_INSERTION_PENDING followed immediately by another HPI event marking the transition from SAHPI_HS_STATE_INSERTION_PENDING to SAHPI_HS_STATE_ACTIVE. The automatic transitions through HPI's Insertion Pending and Extraction Pending states means:

- saHpiHotSwapPolicyCancel() always returns SA_ERR_HPI_INVALID_REQUEST
- saHpiResourceActiveSet() always returns SA_ERR_HPI_INVALID_REQUEST
- saHpiResourceInactiveSet() always returns SA_ERR_HPI_INVALID_REQUEST
- saHpiAutoInsertTimeoutGet() always returns SAHPI_TIMEOUT_IMMEDIATE
- saHpiAutoInsertTimeoutSet() always returns SA_ERR_HPI_READ_ONLY
- saHpiAutoExtractTimeoutGet() always returns SAHPI_TIMEOUT_IMMEDIATE
- saHpiAutoExtractTimeoutSet() always returns SA_ERR_HPI_READ_ONLY

BladeCenter blades, I/O modules, and Management Modules do not possess hot swap indicator LEDs so:

- saHpiHotSwapIndicatorStateGet() always returns SA_ERR_HPI_CAPABILITY
- saHpiHotSwapIndicatorStateSet() always returns SA_ERR_HPI_CAPABILITY

The HPI application may get HPI events re-announcing a state. For example, when in ACTIVE state, BladeCenter may generate another hot swap event stating that its in the ACTIVE state. Such re-announcements are informational and carry matching values for previous and current state.

5 Event Processing

HPI defines a simple polling interface for events. Internally, however, things get a little complex. There are actually four major logs that are involved in the processing of events:

- BladeCenter Hardware Event Log
- OpenHPI BladeCenter Plugin Resource Event Log
- OpenHPI Domain Event Log
- OpenHPI Domain Alarm Table

5.1 BladeCenter Hardware Event Log

At the base, is the BladeCenter's Hardware Event Log. This log is persistent in the BladeCenter hardware and is preserved over MM failovers and failures. All the noteworthy (plus some not so noteworthy) events in the BladeCenter system are logged here. The log is accessible externally through SNMP commands.

In a simplified event flow, an HPI application issues *saHpiEventGet()* which flows through the OpenHPI Infrastructure code to the OpenHPI BladeCenter plugin code. The OpenHPI BladeCenter plugin code issues an SNMP command to the BladeCenter Management Module to retrieve an event from the BladeCenter's Hardware Event Log. The BladeCenter event is then translated into an HPI event and returned the event to the OpenHPI Infrastructure code. The OpenHPI Infrastructure code then logs the event in its Domain Event Log, so that its available to all subscribed users. If the severity is high enough, the event is also logged to the Domain Alarm Table.

All HPI Resource Event Log commands, *saHpiEventLogxxx()*, deal with the BladeCenter Hardware Log and its characteristics and are supported by the OpenHPI BladeCenter plugin code.

All HPI Domain Event Log commands, *saHpiEventGet()* and *saHpiEventAdd()*; and all HPI Domain Alarm Table commands, *saHpiAlarmxxx()*, are supported entirely by the OpenHPI Infrastructure code. More on these in a bit.

5.1.1 Non-Writable

The BladeCenter Hardware Event Log is not writable from external sources. *SA_ERR_HPI_INVALID_CMD* is returned for the command *saHpiEventLogEntryAdd()*.

The HPI Infrastructure code, however, does support writable user-defined events to the Domain Event Log and the Domain Alert Table. So the commands *SaHpiEventAdd()* and *SaHpiAlarmAdd()* are supported.

5.1.2 Variable Length

The BladeCenter Hardware Event Log is variable in size. The size varies depending upon what specific events are logged. Some are larger than others. Typically the number of events range from around 350 to almost 800 events. HPI assumes fix length logs so the OpenHPI BladeCenter plugin fibs a bit and returns a value of 768 for the maximum length of the log. 768 is based on the smallest BladeCenter event message.

5.1.3 Wrap Characteristics

When the BladeCenter Hardware Event Log gets full, it wraps. The oldest events are replaced

with the newer ones. Generally this isn't a problem, as we'll see, since the OpenHPI BladeCenter plugin code keeps a copy of the BladeCenter Hardware Event Log in its memory – which does not wrap. So events that are replaced in the hardware event log are generally still available in the plugin's memory cache.

5.1.4 Too Much Stuff

Lots of stuff is logged in the BladeCenter's Hardware Event Log – not all of it useful to HPI applications or easily translated into HPI structures. However, no event in the hardware event log is lost or ignored. If the BladeCenter plugin code doesn't know what to do with an event (or doesn't think it's useful), it translates the event into an HPI OEM event with the event's message put into the Text Buffer of the OEM event. An application could parse the Text Buffer of OEM events if desirable; but, in general, HPI applications should simply ignore HPI OEM events.

A list of the hardware events explicitly mapped into non-OEM HPI events are listed in the OpenHPI file – *snmp_bc_events.map*.

5.2 OpenHPI BladeCenter Resource Event Log

HPI's Resource Event Log (REL) for BladeCenter is supported in the OpenHPI's BladeCenter plugin code. The REL is associated with the Virtual Management Module resource. The plugin code keeps the REL synchronized with the BladeCenter Hardware Event Log.

To improve performance, the OpenHPI BladeCenter plugin code generally reads more than one event from the BladeCenter Hardware Event Log at a time. When an HPI application requests an event, the BladeCenter plugin determines if there are any new events to read from the hardware. If there are, the plugin reads *all* the new events in the BladeCenter Hardware Event Log at once. It stores the new events in the REL and feeds them to the OpenHPI Infrastructure code. If there are no new events to be read from the hardware but there are unprocessed events in the REL, the event is returned from the REL memory. This improves SNMP network efficiencies and reduces the chances that events are lost due to the hardware log wrapping.

5.2.1 Initial Bring-up

The first time HPI comes up, the BladeCenter plugin code reads all of the BladeCenter's Hardware Event Log to create the REL. This may take a very long time (minutes) - there can be hundreds of events to read. There are a couple of things one can do to improve bring-up performance:

- Clear the Event Log before requesting the first event
Issuing *saHpiEventLogClear()* causes the BladeCenter Hardware Event Log to be cleared. Of course, you'll lose historical data (which may be stale).
- Use SNMP v3 and configure the “count_per_getbulk” variable in *openhpi.conf* to be as large as possible. Using SNMP GET-BULK commands to retrieve the initial set of events can dramatically reduce bring-up time. The maximum value for “count_per_getbulk” depends on the Management Module code level. Consult *openhpi/plugins/snmp_bc/README* for the details.

Besides performance, the other area of concern is stale data. When the BladeCenter Hardware Event Log is first read, it can contain events for resources that are no longer in the chassis (e.g. removed blades). The OpenHPI BladeCenter plugin code won't propagate events to the OpenHPI Infrastructure code until after the initial bring-up and discovery is finished. If the HPI application wants to read historical data, it can issue *saHpiEventLogGet()*; if it just wants events after bring-up, it can issue *saHpiEventGet()*.

5.3 OpenHPI Domain Event Log

This is the log that the HPI application reads when issuing ***saHpiEventGet()***. Events can be added to this log using ***saHpiEventAdd()***.

Generally the OpenHPI Domain Event Log and the plugin's REL have more events than the application has read. When the application requests an event, it can usually be retrieved from memory instead of having to request the event from the hardware. This greatly improves performance. OpenHPI can also be configured to run in a threaded mode. In this mode, a separate thread polls the plugins for events every 3 seconds. The idea is to off-load the time getting and translating events directly from the hardware.

This design should also reduce concern about log wrapping. Unless the hardware event log can be overflowed in 3 seconds, events will be picked up and saved in memory until the HPI application gets around to processing them. The amount of memory used to store events can be capped by the user, if desired, through the OPENHPI_EVT_QUEUE_LIMIT configuration option in ***openhpi.conf***.

5.4 OpenHPI Domain Alarm Table

For events with severities greater than or equal to SAHPI_MINOR, the OpenHPI Infrastructure code stores and tracks in the Domain Alarm Table (DAT). As events are recovered, they are removed from this table. User applications can add events to the DAT with the command – ***saHpiAlarmAdd()***. The size of the DAT can be limited, if desired, by the OPENHPI_DAT_SIZE_LIMIT configuration option in ***openhpi.conf***.