# *The Service Availability Forum* Platform Interface Overview and Model

Zhang, xiaolu

In March. 2003
Seminar Paper on High Availability and Timeliness in Linux
Department of Computer Science
Helsinki University

## ABSTRACT

The Service Availability Forum Platform Interface (HPI) is an interface between the operating system or hardware platform and the Service Availability middleware. HPI provides an industry standard interface to monitor and control highly available telecommunications system platforms. . The use of HPI enables carrier-grade systems to run on cost-effective commercial off-the-shelf (COTS) building blocks, while making management of middleware independent of any particular hardware platform. This paper presents a general description of HPI model.

## General Terms

Service Availability (HA), hot-swappable components, carrier-grade,
The Service Availability Forum Platform Interface (HPI), entity identifier,
entity path, resource, domain, middleware, hardware

# 1 Introduction

A Service Availability solution requires a system be highly available and provide continuity of service. In order to provide continuous service, application software must be able to execute continuously. Even with the most reliable hardware available today, this requires a strategy that permits continued operation in the presence of hardware failures. Therefore, system platforms must provide redundant components and methods for allocating application processing to those components. The ability to repair the hardware platform while it continues to operate is required in systems that are to provide continuous service. This implies a need to have "hot-swappable" components in the system. Often, system platforms include the ability to actually change the hardware configuration, in addition to just replacing failed components, while the system continues to run. Thus, implementation of a service availability solution requires following platform management capabilities:

- Monitoring the environment of system components , including measurement of ambient temperatures, component temperatures, and input voltage levels.
- Monitoring the performance of system components such as fan speeds, power supply output voltages .
- Setting various configuration or operational characteristics of hardware components in the system
- Reporting "inventory data" including component model numbers, serial numbers, revision levels for field replaceable components.
- Detecting and managing the hot-swap actions of components in the system, including controlling power to individual components.
- Operating watchdog timers, which can automatically reset or power cycles a board if the software running on that board stops.

In order to correctly control processing of the continuously available application, The Service Availability Middleware must understand how to interact with the hardware platform, and utilise its platform management capabilities. While various approaches to standardisation of management capabilities have been developed, no existing platform management standards provide a strong, common model of a high-availability system with redundant, hot-swappable components. The Service Availability Forum Platform Interface called "Hardware Platform Interface( HPI) " is designed to solve this problem.

# 2 The overview of HPI

## 2.1 Standard carrier-grade interface

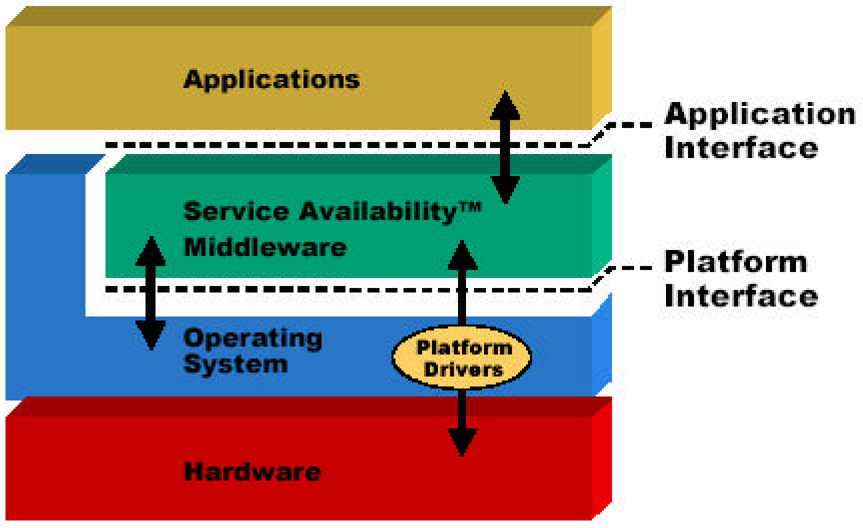

Figure 1: The Service Availability™ Forum Interfaces

The Service Availability Forum (SAF) is developing two layers of standard Carrier-grade interfaces: an application interface and HPI (Figure 1). The SAF application interface provides access to standard set of tools for application software to use in order to distribute its processing over multiple computing elements, and to respond to failures of those elements without loss of service delivery or continuity to any user.

Middleware that conforms to these specifications called Service Availability middleware. Service Availability middleware provides tools to application software in part by monitoring and controlling the physical components of a high-availability computing platform via HPI.

By using a standard interface to manage the physical platform, Service Availability middleware can be written that is independent of any particular hardware. This, in turn, allows application developers to choose the best hardware platform and the best Service Availability middleware to fit the needs

## 2.2 Role and benefits of HPI

The Service Availability middleware has the responsibility of monitoring and controlling the hardware platform, and providing services to the application software that are independent of any particular hardware. The middleware does this by using the standard platform interface to discover what capabilities the hardware platform provides, then mapping these capabilities into a standard system model, which is maintained by the Service Availability middleware and presented to application software. HPI is an independent interface usable by any software package that needs to monitor and control the hardware platform. this standard interface, vendors make their hardware manageable by any software written to use the interface. In particular, the adoption of the HPI specification by platform vendors provides a migration path from purely proprietary systems in use today to completely open solutions in the future. Today, application software often includes availability management functionality embedded within it, and the application is then adapted to a specific hardware platform.

By using a standard interface to manage the physical platform, Service Availability middleware can be written that is independent of any particular hardware. This, in turn, allows application developers to choose the best hardware platform and the best Service Availability middleware to fit their needs. Using HPI, following benefits can be provided:

- Service Availability middleware packages can be developed more quickly, and more economically, because there is no need for the middleware developers to devote significant resources to "porting" their product to various hardware platforms.

- Platform vendors may leverage more Service Availability middleware options by developing a single standard platform interface that is immediately usable by different middleware implementations.

- Platform vendors are free to innovate, because new hardware features may be quickly

- Users will see a more consistent model of the hardware platform for management as it is "filtered" through the standard interface and middleware

## 3 Description of HPI model

HPI provides a platform-independent interface to platform-specific management services. It represents the platform-specific characteristics of the system in an abstract model, and then provides standard methods of monitoring or controlling that model .

The components of the system are represented by the HPI as ***entities***. An entity's manageability is modelled in HPI by controls, sensors, entity inventory repositories, and watchdog timers, which are defined in resource data records (RDRs) associated with the entity. These controls, sensors, entity inventory repositories and watchdog timers are the mechanisms by which HPI users can control and receive information about the state of the system.

Entity management via the HPI may include any combination of the following functions :

- Reading values related to the operation or health of a component. This ability to read operational or health data is modelled via **Sensors** associated with the entity.

- Controlling aspects of the operation of a component. This ability to control a component is modelled via **Controls** associated with the entity, plus a special function to control the resetting of a component.

- Reporting inventory and static configuration data. This data is reported via **Entity Inventory Records** associated with the entity.

- Operating watchdog timers on components. **Watchdog timers** may cause implementation-defined actions to occur when the timers expire. The ability to operate watchdog timers is modelled via Watchdog Timers associated with the entity.

## 3.1 Entity

Each entity has a unique identifier. An entity identifier consists of the combination of an ***entity type*** and an ***entity instance***. The entity type describes the type of hardware component, while the entity instance allows multiple occurrences of a particular hardware component to be distinguished. the HPI model uses a list of entity type/entity instance pairs called an ***entity path***, to fully identify each component. The path is ordered from the entry itself, to the root of the system hierarchy. Figure 2 shows an example of a system platform with example entity paths for a few components.

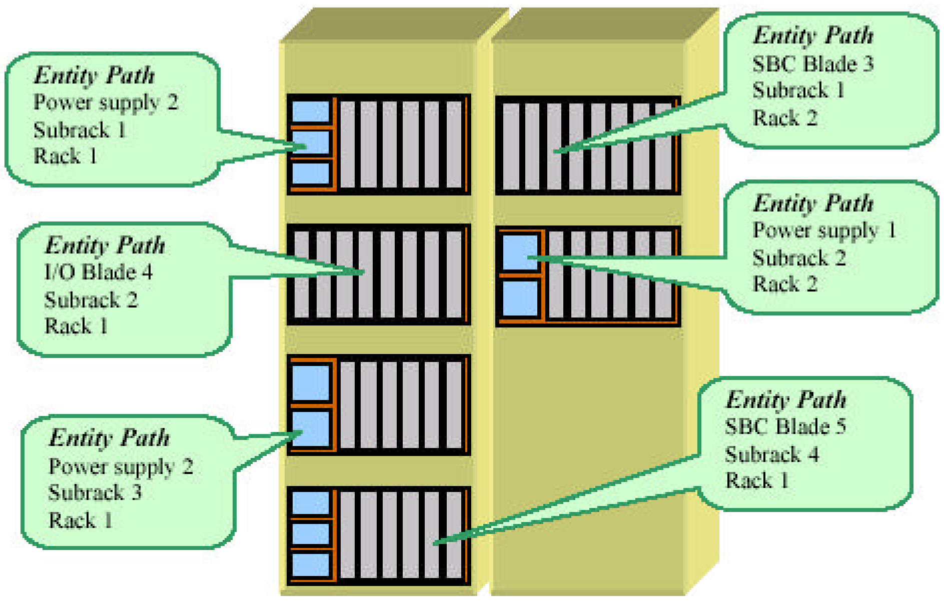

Figure 2: HPI Physical View

## 3.2 Management capabilities

Management capabilities of system entities are modelled in the HPI with **sensors**, **controls, entity inventory repositories and watchdog timers.** One of the key things that the platforms used for carrier-grade solutions is their inclusion of **hot-swappable** components. At a minimum, platforms generally have the ability to remove and replace failed components while the rest of the system remains operational. More commonly, they include a general capability to have components inserted or removed while the system runs-for reconfiguration or repair purposes.

### *3.2.1 Entity Inventory Repositories*

Each physical entity in the system may have inventory data associated with it, readable via the HPI interface. This inventory data, such as manufacturer, model number, revision level, serial number, and static configuration information, is accessible by reading records from an entity inventory repository. The HPI user may read the inventory data from any repository, and may also update the data in a repository.

### 3.2.1 Sensors

**Sensor** is used to minter any physical characteristic of an entity. Sensors usually are used to model values that may change over time, like temperatures, voltages etc. They also can be used to report static configuration information such as the setting of switches on a component**.**

**Sensors** also have associated *event states*. When changes in the reading occur, the sensor can assert or de-assert one or more of event states. When a sensor asserts or de-asserts an event state, an HPI event message may be created and sent to an HPI user or put in a log.

### *3.2.3 Controls*

While a sensor is used to monitor physical entities, **control** is used to send a command to an entity. Controls are abstract management capabilities that are tied to actual capabilities by the HPI implementation. Six types of controls are defined to handle different sorts of data that may need to be sent to the entity. The defined control types are:

- Digital – send on/off type settings
- Discrete – send multiple-bit settings where the meanings of different bits or patterns is implementation specific
- Analogy – send an analogue value
- Stream – send a repeating pattern of on/off bits (e.g., to make an audible alarm beep in a fixed pattern)
- Text – send text to a display device

### *3.2.4 Watchdog Timers*

A **watchdog timer** management capability is used to control physical watchdog timers that may be implemented on physical entities. Specialised function calls are available to configure and start a watchdog timer, to send “keep-alive” heartbeats to it, and to define what actions are taken when it expires.

## 3.3 Resource

Access to the entities within the system is via management access points in the system infrastructure, represented in the HPI as ***resources***. A ***Resource*** is simply a collection of management abilities associated with one or more entities. ***Management abilities*** are modelled as belonging to a single resource when they share common accessibility in the systems. For example, if the management infrastructure in a platform contains several management controllers, each of which provides management services for different sets of boards, fans, power supplies, etc., then each of these management controllers may be modelled as a separate resource containing the sensors and controls that correspond to the actual management services provided by that controller. If one of the management controllers fails or is removed from the system, the interface can report that the corresponding resource is no longer available, and all of the specific sensors, controls, etc., hosted by that management controller are therefore not accessible.

A resource does not have to correspond to a physical management controller, however. There are other reasons the platform interface may make parts of the platform management infrastructure available or unavailable at different times, or to different users. One important reason is in support of ***hot-swappable*** components. Another reason is to give certain users management control over just part of the overall system. This can be particularly useful in multi-tenant systems where parts of a high-availability system platform are leased by a service provider to independent customers.

### 3.4 Domain

The HPI view of a system is divided into one or more ***domains***, where a domain provides access to some set of the resources within the system. A domain represents some part of the system that is capable of being managed by a single HPI user.

A resource is a member of one or more **domains** in the HPI model. Access to all of the management instruments contains in resource is permitted to all users who are able to access a domain that contains the resource. Resources may be members of more than one domain at the same time, permitting different sets of users simultaneous access.

### 3.5 Sessions

A user of the HPI accesses the system through sessions, where each session is opened in a domain. A session provides access to resources that are visible in the domain upon which the session is opened. When a user initiates an HPI session, a domain identifier must be provided, and a session identifier is returned. A user of HPI only access resources that are visible in that domain. Within a given session, the user will be supplied with all events generated by resources in the associated domain. One domain can have multiple sessions open on it; and any HPI user may have multiple sessions open at once.

## 4 Using HPI interface

The HPI is made available in a series of C language library calls and a header file provided by a platform vendor. The header file is taken directly from the SAF specification, and except for assigning a few basic data types to the appropriate types for the processor family, no changes are required by the vendor. The library functions, however must be written specifically for each system platform to map the HPI model and functionality to the actual hardware platform capabilities. The platform vendor must provide libraries appropriate for whatever operating system and compiler is used on the platform.

Service Availability middleware, or other software user software, invokes the library by including the header file in its software modules, and making calls to the appropriate library functions.

After establishing a session to a domain, use software can use function calls provided in the interface to discover all the resources that are members of the domain , and discover all management capabilities and entities visible and accessible via domain, as well as any entity inventory repositories.

Any call to the HPI may cause a context switch from the calling process, and may thus be interrupted by normal OS means. A particular HPI may implement most or all calls as remote procedure calls, or may suspend the calling process while it fetches requested information from the management infrastructure.

## 5 Conclusion

The Service Availability Platform Interface is an interface between the operating system or hardware platform and the Service Availability middleware. The interface is built on a model that provides an abstract view of the system, while still allowing full access to whatever management capabilities it may contain. The Service Availability Platform Interface provides a standard interface for the rich management capabilities found in the carrier-grade hardware platforms used with applications that must provide continuous availability.

The Service Availability Platform Interface enables the design of highly reliable infrastructure products, without the limitations of proprietary interfaces. Benefits of the Service Availability Platform Interface include shorter development cycles, development cost savings, lower total cost of ownership, improved design flexibility, reduced development risk and faster innovation.

The use of HPI enables carrier-grade systems to run on cost-effective commercial off-the-shelf (COTS) building blocks, while making management middleware independent of any particular hardware platform. HPI is destined to become a significant requirement for open architecture systems in the near future.

## References

- Service Availability Specification 2003
  http://www.saforum.org/specification

- GoAhead Service Availability Forum 2002
  http://www.goahead.com/sr/sravailability.htm
- Collin Holland (11/8/2002) Service Availability Forum's Platform Interface spec released to ease infrastructure design
  http://www.embedded.com/story/OEG20021108S0021
- What is the Service Availability™ Solution 2001
  http://data.goahead.com/sr/11652g%20White%20Paper.pdf