

Open industry standards for high-availability systems

by Ajay Kamalvanshi, Sayandeb Saha, and Timo Jokiahho,
Service Availability Forum

The Service Availability Forum (SA Forum) recently released new specifications. The primary theme of this release is to make the interfaces implementable and to make the implementation more consistent from the application developer's perspective.

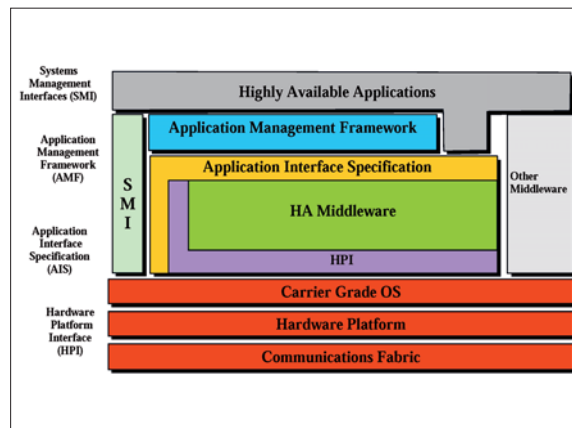


Figure 1. Service availability interfaces

■ The trend in the computer and communication industries to use modular and commercial off-the-shelf (COTS) hardware components is continuing. Today, equipment providers have a choice of hardware components from numerous vendors to build highly reliable products with less effort and shorter time-to-market. While COTS hardware components are readily available now, the software components have lagged behind primarily due to lack of open industry standards. Most of the software for highly available systems is developed using a proprietary middleware and clustering interface resulting in costly systems that are difficult to maintain and take longer time to deploy new services. These cluster software solutions are developed as monolithic software from bottom-up with proprietary interfaces to manage the underlying hardware and to build services on top of the cluster. The need for reducing complexity of software and developing modular and re-usable software component led to the creation of open industry standards.

The Service Availability Forum (SA Forum), an association of companies providing network equipment and system software, was formed to define open industry standards for highly available systems. Since its formation, the SA Forum has developed and published interface specifications for hardware platform management, clustering middleware, and for managing

the highly available systems and applications illustrated in figure 1. The SA Forum recently released new specifications. The primary theme of this release is to make the interfaces implementable and to make the implementation more consistent from the application developer's perspective. This is achieved by removing the ambiguities that could result in different interpretations of the specifications by vendors, defining missing features, and providing guidelines for options where multiple choices occur. This release contains a new mapping specification to map the hardware platform interface (HPI) to PICMG's AdvancedTCA platform and an updated application interface specification (AIS). In addition, a distributed system management interface for AIS has been released for the first time.

The HPI is a generic interface to monitor and control underlying hardware platforms and is defined in SAI-HPI-B.01.01. The HPI specification abstracts hardware platform characteristics into a data model consisting of entities and resources. This model has two views: a physical view with each physical component represented as an entity with its associated path reflecting its position in the containment hierarchy and a management view with each manageable component represented as a resource. Each resource has a set of management instruments that are used to manage a system com-

ponent. The management software that implements this specification exposes the management view. It provides APIs to discover the component hierarchy and to perform management and control operations. The HPI specification is focused on defining interfaces without providing or recommending any specific model for a commercial hardware platform.

The recently released HPI-to-AdvancedTCA mapping specification, SAIM-HPI-B.01.01-ATCA, is the SA Forum's first mapping specification that will help HPI implementers to provide a consistent logical model for systems based on AdvancedTCA. It will also help the HPI users such as cluster middleware and system management software to obtain a vendor-independent hardware platform view of an AdvancedTCA system. Using an example of the AdvancedTCA system as shown in figure 2, the specification proposes a three-tiered model to create system topology and containment hierarchy of all entities.

The top-level part is used for identification of the chassis in operator's premises and contains information such as shelf and rack location. The middle-level represents shelf and field-replaceable units (FRUs) such as physical slots that map to site types and locations of the AdvancedTCA shelf. These can be found in the AdvancedTCA address table and derived from

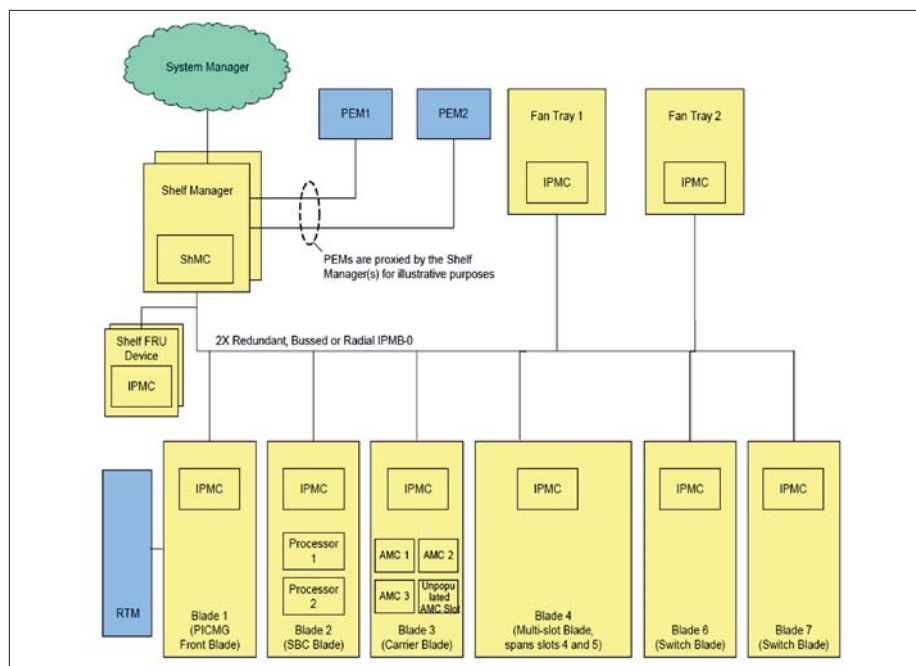


Figure 2: Example AdvancedTCA system

the AdvancedTCA IPMI command “get address info”. The bottom-level entities represent physical functional entities of the shelf such as FRUs (front boards, fans, PEMs), mezzanine boards

(PMCs, AMCs), and alarm modules that are installed in slots represented by the middle-level entities. A one-to-one mapping is provided in the specification between AdvancedTCA

entities to HPI entities. For the management view, the mapping specification also provides a clear set of rules on what types of entities are contained in other types of entities. These rules, referred to as entity schema or entity containment tree, will promulgate use of a well-known standard system organization for management as illustrated in figure 3.

For example, an AdvancedTCA system consists of a rack which contains a chassis. The chassis may contain a shelf manager slot that contains a shelf manager. The chassis contain multiple blade slots. The blade slot may contain a front blade that may contain a processor, and so on. In addition, the chassis may contain power supplies and cooling units, and cooling units may contain fans.

Furthermore, the mapping specification describes the details of how the general AdvancedTCA resources such as shelf, slot, shelf FRU device, and shelf manager map to HPI resources. The functional aspects such as hotswap (state machine mapping), electronic keying, and cooling are discussed in detail. Finally, mapping of data types such as sensors, event, event log, and inventory data records to HPI data types completes the specification.

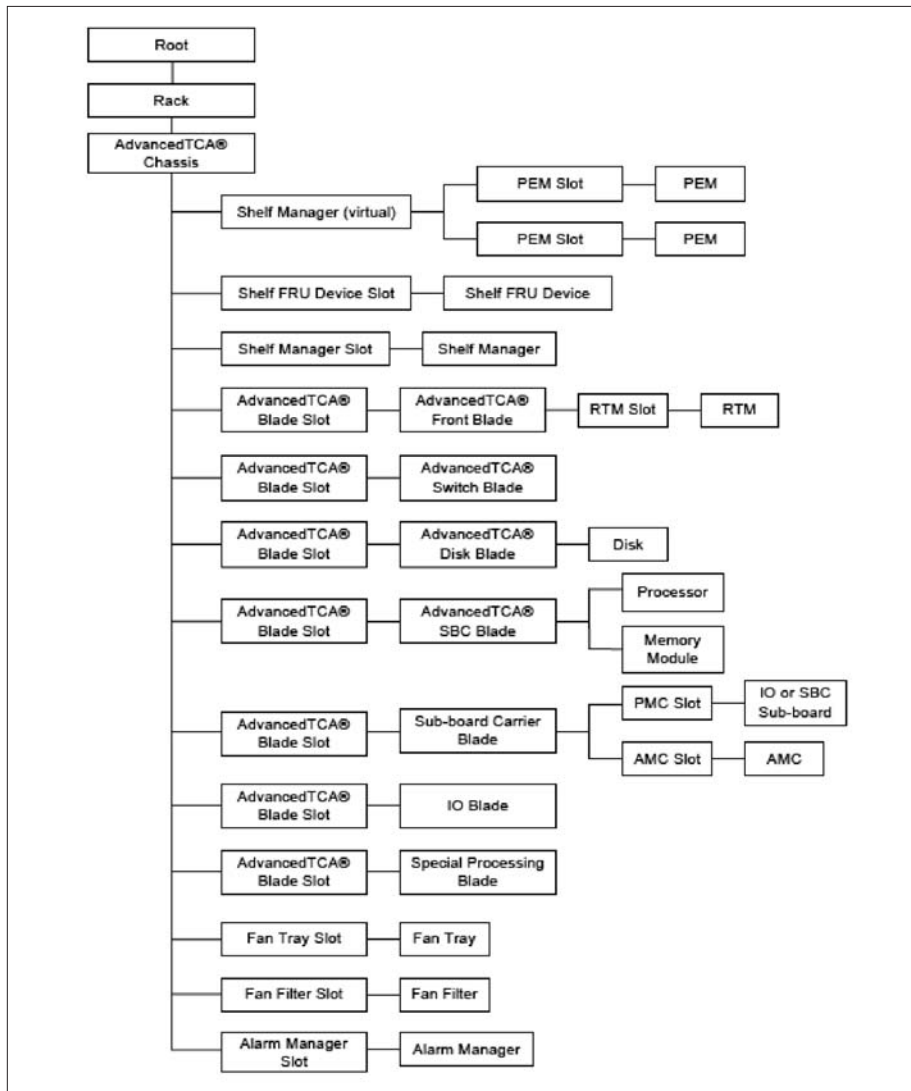


Figure 3. Entity schema

The SA Forum’s application interface specification (AIS) defines the interface between highly available application and cluster middleware using application programming interfaces (APIs). This interface is similar in look-and-feel to the POSIX interface, which provides operating system abstraction to UNIX applications. The AIS allows programmers to write highly available application software that is portable across different vendor’s implementations of the AIS specifications allowing TEMs, NEPs and other telcos to choose an implementation that is more suitable for their target application with the added benefit of not getting locked to a particular vendor’s proprietary middleware.

The AIS specification consists of availability management framework (AMF) and other accessory cluster services, referred to as the SA services. The AMF coordinates redundant resources within a cluster to provide service availability with no single point of failure. It does so by monitoring the health of system components (hardware and software) and dynamical-

ly assigning work-load and software resources based on system configuration to ensure continuous service availability. The APIs provided by AMF include registration, de-registration, health monitoring, availability-related control and state management, protection group handling and error handling. The cluster services or SA services provide basic HA functionality on which AMF and applications are implemented. These services include cluster membership service which maintains and provides membership information about nodes in the cluster, checkpoint service for synchronizing data between applications running on different nodes, a distributed messaging service, a publish-subscribe event service and a distributed lock service.

The latest version of AIS specification, AIS B.02.01, conceptually follows the structural organization of its predecessor, AIS B.01.01. However, instead of referring a document as a separate volume, each document uses its conventional name consisting of common prefix of

“SAI-AIS” followed by an abbreviation of the basic function or service it describes. For example, SAI-AIS-CKPT-B.02.01 describes the checkpoint service.

The latest release also includes an overview document, SAI-Overview-B.02.01, that sets the background to read all the other documents. It describes the document organization structure, programming model, naming conventions, standard predefined types, return values and constants, abbreviations, terms and concepts. This document also illustrates information model for AMF and other AIS services with conceptual views and class diagrams to explain relationships among various logical entities used in the specifications.

The latest release contains three new services in addition to the existing SA services: notification, log and information model management (IMM). The notification service, SAI-AIS-NTF-A.01.01 – note all new specifications are “A” specification compared to more mature, older “B” specifications – is closely based on X.700 series of international telecom union telecommunication standardization sector’s (ITU-T’s) specification on fault management (M-EVENT-REPORT service) and is centered around the concept of status and incident reporting. It uses the term notification to clearly distinguish itself from the existing AIS event service. Like AIS event service, notification service uses the publish-subscribe-pattern, where a producer publishes the occurrence of an incident and the subscribed consumers receive the information when an incident occurs. However, these incidents have a pre-defined well-known syntax and indicate alarm conditions (service affecting or security related), object creation or deletion, state change and attribute value change. The new APIs are provided to publish the notification, subscribe to notification and read historic notifications.

The second new AIS service, log service defined in SAI-AIS-LOG-A.01.01, is the first serviceability service for troubleshooting and root cause analysis. It provides APIs for application and middleware implementation to store an ordered set of information, log record, to four different types of log streams such as alarm, notification, system, and application. All notifications produced by the SAF notification service are logged in the respective well-known log streams that are made available by the SAF LOG service. Additional APIs are included to read the log streams. These APIs will help in developing tools that system administrators can use to view cluster-significant, function-based information.

The third new AIS service, IMM service defined in SAI-AIS-IMM-A.01.01, provides APIs to manage and manipulate the SAF AIS informa-

tion model. The information model, specified in UML, describes runtime and configuration objects in various specifications such as AMF (components, service units, and service instance), checkpoint service (checkpoint) and message (message queue). Each object in the information model has attributes and administrative operations, and the management application can create, access and manage these objects using the defined APIs. Additionally, the IMM service also allows system administrators to invoke various administrative operations on objects that are controlled by the individual AIS services.

In addition to three new services, a chapter on the information model is included in the overview document. This chapter provides three global views of object classes: HPI view, cluster view and AMF view. These views present a comprehensive high level view of all the SA Forum objects and will help the middleware vendors to implement SA Forum specifications. It will also help the application developers to correctly interpret various concepts used in the specifications and design portable applications. Some other new additions in the specification include management of proxy and proxied components, extensive changes to the AMF

state model, methods to integrate non-SA-aware components, clarification of memory ownership rules, auto-repair concept, definition of administrative APIs for AMF and LOG service and more. Additionally, the latest release of AMF and all other AIS specifications contains a new section at the very end that defines the various alarms, state change, object change and attribute change notifications that are produced by these services. These are described based on the syntax provided by the new SAI-AIS-NTF-A.01.01 specification described previously in this article. The specifications also clarify the ambiguities and missing information in several sections.

The SA Forum has also released system management interfaces, SAI-AIS-SNMP-A.01.01, for the first time. As the name suggests, the management is based on internet engineering task force's (IETF's) simple network management protocol (SNMP) containing three fundamental components: managed objects, management application and agent. The managed objects are collectively referred to as a management information base (MIBs) and are defined using structure for management information version 2 (SMIV2). The management application, also known as manager, requests manage-

ment operations to agent and asynchronously receives traps from agents. The agent runs on a cluster node and performs operations on the resources via managed object implementation. While the SNMP agent runs on few nodes in the cluster, it can obtain all the cluster-wide information of AMF and AIS services.

The system management specification contains SA Forum global and AIS textual conventions such as SAF predefined types (SafTimeStamp, SafUnsigned64), traps, errorReturnCode, probableCause, etc. In addition, MIBs for AMF and other SA services are specified. A private enterprise object identifier (OID) is allocated by IANA and all the MIBs are defined with this prefix, iso.org.dod.internet.private.enterprise.saforum. Each MIB has common scalar objects for the specification version that agent support, vendor name and vendor product revision name.

The AMF MIB contains tables for logical system entities such as applications, nodes, service groups, service units, service instances, components, component service instances and proxy-proxied components along with other objects that need to be manipulated for executing various administrative operations offered by AMF. ■