Service Availability™ Forum
Service Availability Interface

Overview SAI-Overview-B.05.03

SERVICE AVAILABILITY™ FORUM SPECIFICATION LICENSE AGREEMENT

The Service Availability™ Forum Application Interface Specification (the "Package") found at the URL http://www.saforum.org is generally made available by the Service Availability Forum (the "Copyright Holder") for use in developing products that are compatible with the standards provided in the Specification. The terms and conditions which govern the use of the Package are covered by the Artistic License 2.0 of the Perl Foundation, which is reproduced here.

The Artistic License 2.0

Copyright (c) 2000-2006, The Perl Foundation.

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

This license establishes the terms under which a given free software Package may be copied, modified, distributed, and/or redistributed.

The intent is that the Copyright Holder maintains some artistic control over the development of that Package while still keeping the Package available as open source and free software.

You are always permitted to make arrangements wholly outside of this license directly with the Copyright Holder of a given Package. If the terms of this license do not permit the full use that you propose to make of the Package, you should contact the Copyright Holder and seek a different licensing arrangement.

Definitions

"Copyright Holder" means the individual(s) or organization(s) named in the copyright notice for the entire Package.

"Contributor" means any party that has contributed code or other material to the Package, in accordance with the Copyright Holder's procedures.

"You" and "your" means any person who would like to copy, distribute, or modify the Package.

"Package" means the collection of files distributed by the Copyright Holder, and derivatives of that collection and/or of those files. A given Package may consist of either the Standard Version, or a Modified Version.

"Distribute" means providing a copy of the Package or making it accessible to anyone else, or in the case of a company or organization, to others outside of your company or organization.

"Distributor Fee" means any fee that you charge for Distributing this Package or providing support for this Package to another party. It does not mean licensing fees.

"Standard Version" refers to the Package if it has not been modified, or has been modified only in ways explicitly requested by the Copyright Holder.

"Modified Version" means the Package, if it has been changed, and such changes were not explicitly requested by the Copyright Holder.

"Original License" means this Artistic License as Distributed with the Standard Version of the Package, in its current version or as it may be modified by The Perl Foundation in the future.

"Source" form means the source code, documentation source, and configuration files for the Package.

"Compiled" form means the compiled bytecode, object code, binary, or any other form resulting from mechanical transformation or translation of the Source form.

Permission for Use and Modification Without Distribution

(1) You are permitted to use the Standard Version and create and use Modified Versions for any purpose without restriction, provided that you do not Distribute the Modified Version.

Permissions for Redistribution of the Standard Version

(2) You may Distribute verbatim copies of the Source form of the Standard Version of this Package in any medium without restriction, either gratis or for a Distributor Fee, provided that you duplicate all of the original copyright notices and associated disclaimers. At your discretion, such verbatim copies may or may not include a Compiled form of the Package.

(3) You may apply any bug fixes, portability changes, and other modifications made available from the Copyright Holder. The resulting Package will still be considered the Standard Version, and as such will be subject to the Original License.

Distribution of Modified Versions of the Package as Source

(4) You may Distribute your Modified Version as Source (either gratis or for a Distributor Fee, and with or without a Compiled form of the Modified Version) provided that you clearly document how it differs from the Standard Version, including, but not limited to, documenting any non-standard features, executables, or modules, and provided that you do at least ONE of the following:

(a) make the Modified Version available to the Copyright Holder of the Standard Version, under the Original License, so that the Copyright Holder may include your modifications in the Standard Version.

(b) ensure that installation of your Modified Version does not prevent the user installing or running the Standard Version. In addition, the Modified Version must bear a name that is different from the name of the Standard Version.

(c) allow anyone who receives a copy of the Modified Version to make the Source form of the Modified Version available to others under

(i) the Original License or

(ii) a license that permits the licensee to freely copy, modify and redistribute the Modified Version using the same licensing terms that apply to the copy that the licensee received, and requires that the Source form of the Modified Version, and of any works derived from it, be made freely available in that license fees are prohibited but Distributor Fees are allowed.

Distribution of Compiled Forms of the Standard Version or Modified Versions without the Source

(5) You may Distribute Compiled forms of the Standard Version without the Source, provided that you include complete instructions on how to get the Source of the Standard Version. Such instructions must be valid at the time of your distribution. If these instructions, at any time while you are carrying out such distribution, become invalid, you must provide new instructions on demand or cease further distribution.

If you provide valid instructions or cease distribution within thirty days after you become aware that the instructions are invalid, then you do not forfeit any of your rights under this license.

(6) You may Distribute a Modified Version in Compiled form without the Source, provided that you comply with Section 4 with respect to the Source of the Modified Version.

Aggregating or Linking the Package

(7) You may aggregate the Package (either the Standard Version or Modified Version) with other packages and Distribute the resulting aggregation provided that you do not charge a licensing fee for the Package. Distributor Fees are permitted, and licensing fees for other components in the aggregation are permitted. The terms of this license apply to the use and Distribution of the Standard or Modified Versions as included in the aggregation.

(8) You are permitted to link Modified and Standard Versions with other works, to embed the Package in a larger work of your own, or to build stand-alone binary or bytecode versions of applications that include the Package, and Distribute the result without restriction, provided the result does not expose a direct interface to the Package.

Items That are Not Considered Part of a Modified Version

(9) Works (including, but not limited to, modules and scripts) that merely extend or make use of the Package, do not, by themselves, cause the Package to be a Modified Version. In addition, such works are not considered parts of the Package itself, and are not subject to the terms of this license.

General Provisions

(10) Any use, modification, and distribution of the Standard or Modified Versions is governed by this Artistic License. By using, modifying or distributing the Package, you accept this license. Do not use, modify, or distribute the Package, if you do not accept this license.

(11) If your Modified Version has been derived from a Modified Version made by someone other than you, you are nevertheless required to ensure that your Modified Version complies with the requirements of this license.

(12) This license does not grant you the right to use any trademark, service mark, tradename, or logo of the Copyright Holder.

(13) This license includes the non-exclusive, worldwide, free-of-charge patent license to make, have made, use, offer to sell, sell, import and otherwise transfer the Package with respect to any patent claims licensable by the Copyright Holder that are necessarily infringed by the Package. If you institute patent litigation (including a cross-claim or counterclaim) against any party alleging that the Package constitutes direct or contributory patent infringement, then this Artistic License to you shall terminate on the date that such litigation is filed.

(14) Disclaimer of Warranty:

THE PACKAGE IS PROVIDED BY THE COPYRIGHT HOLDER AND CONTRIBUTORS "AS IS' AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES. THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT ARE DISCLAIMED TO THE EXTENT PERMITTED BY YOUR LOCAL LAW. UNLESS REQUIRED BY LAW, NO COPYRIGHT HOLDER OR CONTRIBUTOR WILL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING IN ANY WAY OUT OF THE USE OF THE PACKAGE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Table of Contents

List of Figures

List of Tables

1 Document Introduction

1.1 Document Purpose

This document, **SAI-Overview-B.05.03**, provides an overview of the Service Availability™ Forum (SA Forum) Interface specifications.

The main purpose of this document is to provide an introduction to the objectives of the SA Forum and its specifications, a description of the architecture of the interfaces, and an overview of the functionality covered by the documents published by the SA Forum in Release 6.1 of the specifications.

1.2 Organization of the SA Forum Interface Specification Documents

This section describes the organization of the Service Availability Interface specifications into their constituent documents. To simplify references to the set of documents published together, the latest specifications documents are bundled together into a so-called **release**. The set of documents in a release are intended to be compatible with one another. The version numbers of each document evolve independently of release numbers, thus a given release may consist of documents with different version codes. An overview of the document naming convention is given in Section 3.1. A new release generally contains a mix of new, updated, and unchanged specification documents.

The list of documents comprising Release 6.1 of the Service Availability Interface specifications is given in Section 1.2.1 through Section 1.2.3 below. In addition to the specifications, the list also contains C header files and material for modeling.

1.2.1 C Programming Model

The **SAI-CPROG-B.05.02** document describes the C programming model of the Service Availability™ Forum Application Interface Specification (AIS) specifications. It also provides all type definitions that are common to AIS documents.

1.2.2 Hardware Platform Interface Specification (HPI) Documents and Files

The Hardware Platform Interface Specification is organized into the following documents:

- **SAI-HPI-B.03.02** describes the HPI.
- **SAIM-HPI-B.03.02-xTCA** describes the mapping on AdvancedTCA® and MicroTCA® platforms.
- **SAI-HPI-CH-B.03.02.zip** contains the HPI C Header File.

- **SAIM-HPI-CH-B.03.02-xTCA.zip** contains the C Header Files for the HPI to AdvancedTCA® and MicroTCA® Mapping.

1.2.3 Application Interface Specification (AIS) Documents and Mappings

1.2.3.1 AIS Documents

The Application Interface Specification is organized into the following documents:

- **SAI-AIS-AMF-B.04.01** describes the Availability Management Framework.
- **SAI-AIS-CKPT-B.02.02** describes the Checkpoint Service.
- **SAI-AIS-CLM-B.04.01** describes the Cluster Membership Service.
- **SAI-AIS-EVT-B.03.01** describes the Event Service.
- **SAI-AIS-IMM-A.03.01** describes the Information Model Management Service.
- **SAI-AIS-LCK-B.03.01** describes the Lock Service.
- **SAI-AIS-LOG-A.02.01** describes the Log Service.
- **SAI-AIS-MSG-B.03.01** describes the Message Service.
- **SAI-AIS-NAM-A.01.01** describes the Naming Service.
- **SAI-AIS-NTF-A.03.01** describes the Notification Service.
- **SAI-AIS-PLM-A.01.02** describes the Platform Management Service.
- **SAI-AIS-SEC-A.01.01** describes the Security Service.
- **SAI-AIS-SMF-A.01.02** describes the Software Management Framework.
- **SAI-AIS-TMR-A.01.01** describes the Timer Service.

1.2.3.2 AIS Files

- **SAI-AIS-R6-CH-A.01.02.zip** contains the C Header files for all AIS specifications.
- **SAI-IM-XMI-A.04.02.xml.zip** contains the SA Forum Information Model in XML Metadata Interchange (XMI) v2.1 format produced using *MagicDraw* version 14.0, Patch 1 (see [31]).
- **SAI-AIS-IMM-XSD-A.01.01.xsd** describes the XML schema of the IMM Service initial file.
- **SAI-AIS-SMF-XSD-A.01.02.zip** contains the XML schemas and documentation for the entity types file and the upgrade campaign specification defined by the Software Management Framework.

1.2.3.3 AIS Java Mapping Files

- **SAIM-AIS-R6-JD-A.01.01.zip** contains the Javadoc of the Java Mapping.
- **SAIM-AIS-R6-JC-A.01.01.zip** contains the Java Code of the Java Mapping.

1.3 History

This section presents the main changes in the current version of this document, **SAI-Overview-B.05.03**, with respect to the version **SAI-Overview-B.05.02** as well as the main changes between the following versions:

- **SAI-Overview-B.05.01** and **SAI-Overview-B.05.02** and
- **SAI-Overview-B.04.02** and **SAI-Overview-B.05.01**.

Additionally, the main changes in other documents between these versions are also presented.

In the following subsections, editorial changes that do not change semantics or syntax of the described interfaces are not mentioned.

1.3.1 Changes from SAI-Overview-B.05.02 to SAI-Overview-B.05.03

The **SAI-Overview-B.05.03** document now covers the Mapping Release 6.1.

1.3.1.1 New Topics

- Chapter 4 has been extended to describe the HPI Mappings to AdvancedTCA and MicroTCA.
- Section 3.5 describes the new Java Mappings.

1.3.2 Main Changes in Other Documents to Which SAI-Overview-B.05.03 Refers

This section and its subsections describe the main changes in other specifications referred to in **SAI-Overview-B.05.03** with respect to the contents of these specifications as they are referred to in **SAI-Overview-B.05.02**.

1.3.2.1 Updated Specifications

Hardware Platform Interface Mapping to xTCA®

⇒ Update to base the mapping on the HPI B.03.02 version

⇒ Addition and adaption of mapping for areas:

- Annunciators
- Version control
- E-Keying

- FUMI
- MicroTCA
- Implementation of change requests

⇒ Cleanup and additions for Entity

⇒ Approach of "generic" xTCA mapping, with MicroTCA and ATCA specials, whenever possible

Java Mappings

⇒ The mappings for IMM and EVT have been added.

⇒ The mappings for new Release 6 interfaces have been updated.

1.3.3 Changes from SAI-Overview-B.05.01 to SAI-Overview-B.05.02

SAI-Overview-B.05.02 describes some corrections for Release 6. Only the references to the new versions of other modified specifications of this release have been updated.

1.3.4 Main Changes in Other Documents to Which SAI-Overview-B.05.02 Refers

This section and its subsections describe the main changes in other specifications referred to in **SAI-Overview-B.05.02** with respect to the contents of these specifications as they are referred to in **SAI-Overview-B.05.01**.

1.3.4.1 Updated Specifications

C Programming Model

⇒ A clarification has been added to the `SaStringT` type.

⇒ The section on the `SaNameT` type and its subsections have been thoroughly revised. The usage and formats of DNs and RDNs have been clarified.

Hardware Platform Interface

⇒ Correction of the Interface Version string

⇒ Update to support new mapping specification

- New Entity Types
- New Sensor Types
- New Control Output types
- Introduction of the `OUTNN` (OUT Not Null) parameter type to mark parameters that are `Out` only, unless set to NULL

⇒ Cleanup of some error codes, wording for parameter description, Max Valid parameter for list enumeration

Platform Management Service

⇒ A new value has been added to the `SaPlmTrackCauseT` enumeration.

⇒ The format of some distinguished names of PLM objects has been updated.

⇒ The format of security alarms has been corrected.

Software Management Framework

The version A.01.02 of the Software Management Framework specification (see [16]) has been released as a correction for Release 5.1, and it is also now part of Release 6. The main changes of this version of this specification with respect to its previous version are:

⇒ An optional optimization for the rolling upgrade procedure has been added.

⇒ Support of templates and IMM modify operations have been added to the single step upgrade.

⇒ The format of some distinguished names of SMF objects has been updated.

⇒ The prerequisites to start an upgrade campaign have been clarified, namely, the applicability of the campaign and the necessity of administrative ownership of IMM objects.

⇒ Missing enumerations for offline commands and missing configuration attributes for software bundle operations have been added.

⇒ The entity types file and the upgrade campaign specification XML schemas have been corrected.

⇒ The term ’prototype’ has been introduced to distinguish partially specified versioned entity types.

1.3.5 Changes from SAI-Overview-B.04.02 to SAI-Overview-B.05.01

This section and its subsections present the changes of the Overview document, **SAI-Overview-B.05.01**, which describes Release 6, with respect to the **SAI-Overview-B.04.02** document.

The **SAI-Overview-B.04.02** document has been split up into two documents for Release 6, the Overview document (**SAI-Overview-B.05.01)** and the AIS C Programming Model document (**SAI-AIS-CPROG-B.05.01**).

1.3.5.1 New Topics

⇒ Section 1.2 introduces the notion of an SA Forum “release”.

⇒ Section 3.1 introduces a convention for naming SA Forum documents.

⇒ Section 3.2 describes the SA Forum Architecture.
It contains among other things,

- a new classification of the AIS interfaces into Platform Services, Management Services, Frameworks, and Utility Services (see Section 3.2.2), and
- a section on service dependencies, which extends the corresponding section of the **SAI-Overview-B.04.01** document.

⇒ Section 3.3 describes the information modeling, Section 3.5 the Java Mappings, and Section 3.6 the virtualization.

⇒ Section 5.1 gives an overview of the new AIS Platform Service, the Platform Management Service, PLM.

⇒ A glossary lists the main terms used in the SA Forum specifications.

1.3.5.2 Deleted Topics

As a glossary comes with this document, the index of definitions (IOD) is no longer provided.

1.3.5.3 Other Changes

In Chapter 10, IMM replaces SNMP as the primary management interface.

1.3.6 Main Changes in Other Documents to Which SAI-Overview-B.05.01 Refers

This section and its subsections describe the main changes in other specifications referred to in **SAI-Overview-B.05.01** with respect to the contents of these specifications as they are referred to in **SAI-Overview-B.04.02**.

1.3.6.1 New Specifications

- Platform Management Service (PLM) for monitoring and controlling the hardware and low level system software

1.3.6.2 Updated Specifications

Details of the changes in the updated specifications are found in the *History* section of their respective specification documents. Only the main changes are mentioned here.

Availability Management Framework

- Introduction of the HA readiness state of service units for their service instances and of the HA Readiness State of components for their component service instances.
- Support for node capacity limits
- Support for correlation ids
- Alignment with PLM

Cluster Membership Service

- Fine-grained track API
- Support for correlation ids
- Alignment with PLM

Information Model Management Service

- Introduction of the notions of validator, applier, and runtime owner roles for object implementers
- Support of notifications and correlation ids

Notification Service

- Introduction of the miscellaneous notification type
- Extended handling of correlation ids

Hardware Platform Interface

- The Firmware Upgrade Management Instruments (FUMI) API has been refined and extended.
- The description of hot swap has been refined and extended.
- The text string representation of entity paths has been added.
- New resource events that allow dynamic updating of resources in a running system have been added.
- The functions `saHpiInitialize()` and `saHpiFinalize()` have been introduced to initialize and finalize the HPI library.
- The `saHpiMyEntityPathGet()` function has been added to return the caller's entity path.

1.4 References

The following documents contain information that is relevant to the specification:

[1] Service Availability™ Forum, Service Availability Interface, C Programming Model, SAI-AIS-CPROG-B.05.02

[2] Service Availability™ Forum, Application Interface Specification, Notification Service, SAI-AIS-NTF-A.03.01

[3] Service Availability™ Forum, Application Interface Specification, Information Model Management Service, SAI-AIS-IMM-A.03.01

[4] Service Availability™ Forum, Application Interface Specification, Availability Management Framework, SAI-AIS-AMF-B.04.01

[5] Service Availability™ Forum, Application Interface Specification, Cluster Membership Service, SAI-AIS-CLM-B.04.01

[6] Service Availability™ Forum, Application Interface Specification, Checkpoint Service, SAI-AIS-CKPT-B.02.02

[7] Service Availability™ Forum, Application Interface Specification, Event Service, SAI-AIS-EVT-B.03.01

[8] Service Availability™ Forum, Application Interface Specification, Lock Service, SAI-AIS-LCK-B.03.01

[9] Service Availability™ Forum, Application Interface Specification, Log Service, SAI-AIS-LOG-A.02.01

[10] Service Availability™ Forum, Application Interface Specification, Message Service, SAI-AIS-MSG-B.03.01

[11] Service Availability™ Forum, Application Interface Specification, Naming Service, SAI-AIS-NAM-A.01.01

[12] Service Availability™ Forum, Information Model in XML Metadata Interchange (XMI) v2.1 format, SAI-IM-XMI-A.04.02.xml.zip

[13] Service Availability™ Forum, IMM XML Schema Definition, SAI-AIS-IMM-XSD.A.01.01.xsd

[14] Service Availability™ Forum, Application Interface Specification, Software Management Framework, SAI-AIS-SMF-A.01.02

[15] Service Availability™ Forum, SMF Entity Types File XML Schema Definition, SAI-AIS-SMF-ETF-A.01.02.xsd [1]

1. Files SAI-AIS-SMF-ETF-A.01.02.xsd and SAI-AIS-SMF-UCS-A.01.02.xsd are packaged together; they are contained in a zip archive named SAI-AIS-SMF-XSD-A.01.02.zip.

[16] Service Availability™ Forum, SMF Upgrade Campaign Specification XML Schema Definition, SAI-AIS-SMF-UCS-A.01.02.xsd[1]

[17] Service Availability™ Forum, Application Interface Specification, Security Service, SAI-AIS-SEC-A.01.01

[18] Service Availability™ Forum, Application Interface Specification, Timer Service, SAI-AIS-TMR-A.01.01

[19] Service Availability™ Forum, Application Interface Specification, Platform Management Service, SAI-AIS-PLM-A.01.02

[20] Service Availability™ Forum, AIS C Header Files, SAI-AIS-R6-CH-A.01.02.zip

[21] Service Availability™ Forum, Mapping for Java, Documentation, SAIM-AIS-R6-JD-A.01.01.zip

[22] Service Availability™ Forum, Mapping for Java, Java Code, SAIM-AIS-R6-JC-A.01.01.zip

[23] Service Availability™ Forum, Hardware Platform Interface Specification, SAI-HPI-B.03.02

[24] Service Availability™ Forum, HPI to AdvancedTCA® and MicroTCA® Mapping Specification, SAIM-HPI-B.03.02-xTCA

[25] Service Availability™ Forum, HPI C Header File, SAI-HPI-CH-B.03.02.zip

[26] Service Availability™ Forum, C Header Files for the HPI to the AdvancedTCA® and MicroTCA® Mapping, SAIM-HPI-CH-B.03.02-xTCA.zip

[27] CCITT Recommendation X.730 | ISO/IEC 10164-1, Object Management Function

[28] CCITT Recommendation X.731 | ISO/IEC 10164-2, State Management Function

[29] CCITT Recommendation X.733 | ISO/IEC 10164-4, Alarm Reporting Function

[30] CCITT Recommendation X.736 | ISO/IEC 10164-7, Security Alarm Reporting Function

[31] `MagicDraw` Home Page (http://www.magicdraw.com)

[32] http://java.sun.com/j2se/javadoc/

References to these documents are made by placing the number of the document in square brackets.

1.5 How to Provide Feedback on the Specification

If you have a question or comment about this Specification, you may submit feedback online by following the links provided for this purpose on the Service Availability™ Forum Web site (http://www.saforum.org).

You can also sign up to receive information updates on the Forum or the Specification.

1.6 How to Join the Service Availability™ Forum

The Promoter Members of the Forum require that all organizations wishing to participate in the Forum complete a membership application. Once completed, a representative of the Service Availability™ Forum will contact you to discuss your membership in the Forum. The Service Availability™ Forum Membership Application can be completed online by following the pertinent links provided on the SA Forum Web site (http://www.saforum.org).

You can also submit information requests online. Information requests are generally responded to within three business days.

1.7 Additional Information

1.7.1 Member Companies

A list of the Service Availability™ Forum member companies can be viewed online by using the links provided on the SA Forum Web site (http://www.saforum.org).

1.7.2 Press Materials

The Service Availability™ Forum has available a variety of downloadable resource materials, including the Forum Press Kit, graphics, and press contact information. Visit this area often for the latest press releases from the Service Availability™ Forum and its member companies by following the pertinent links provided on the SA Forum Web site (http://www.saforum.org).

2 Purpose of the SA Forum and Its Specifications

New and emerging technologies bring ample challenges to the information communications industries. In order to fulfill user expectations, service providers must continue to maintain high levels of availability and dependability during the implementation of new technologies. Service providers need to rapidly deploy new services and vouch for their reliability to compete successfully for users. This means that their information and communications technology (ICT) infrastructure equipment must incorporate the highest levels of availability and dependability while balancing the constraints of short development cycles and increasing pressure to reduce development costs. The ICT industry recognizes that an effective solution is the broad adoption of open specifications.

The SA Forum addresses this situation by specifying standard interfaces that enable the delivery of highly available carrier-grade systems with off-the-shelf hardware platforms, middleware, and service applications. Interface standardization eliminates the need for ICT equipment manufacturers to develop applications from scratch and port them to each new hardware platform. Service continuity is achievable only with the cooperation of all elements in the stack—hardware, middleware, and applications software.

Industry standards for high availability offer many benefits. The implementation of such standards reduces the cost and time of development by eliminating the need for custom high availability code mingled with the application code within application programs. Industry standards benefit application programmers by allowing them to focus on the application logic and the services it provides and minimizing the need for special high availability programming skills. They also allow new hardware and software components to be integrated together into computer and communications systems with greater ease.

The SA Forum has developed a set of software interface specifications for carrier grade platform and middleware applications. The Application Interface Specification (AIS) defines standard interfaces that allow application developers to write highly available software that is portable across multiple platforms. The Hardware Platform Interface Specification (HPI) enables ISVs to develop COTS components that provide hardware platform management capabilities across multiple heterogeneous platforms.

3 Overview of the Service Availability Interface Specifications

3.1 Introduction

The Service Availability™ Forum has defined two interface specifications and their information model:

⇒ The Hardware Interface Specification called the Hardware Platform Interface (HPI). An overview of the HPI is provided in Chapter 4.

⇒ The Application Interface Specification (AIS). AIS is comprised of:

- AIS Platform Services (Chapter 5),
- AIS Management Services (Chapter 6),
- AIS Frameworks (Chapter 7), and
- AIS Utility Services (Chapter 8).

The details of each AIS Service interface are published in a separate document.

⇒ The complete SA Forum Information Model is published in UML using the XML Metadata Interchange (XMI) format.

Whenever the Service Availability™ Forum publishes new or amended interfaces in a given release, the references to the current documents are found in Section 1.4 of this Overview document.

Each SA Forum interface specification document follows a standard naming convention:

SAI[-<*interface*>][-<*topic*>]-<*relCode*>.<*majVersion*>.<*minVersion*>

where:

- **SAI** indicates a Service Availability Interface document.
- ***<interface>*** is optional, it is either HPI, AIS, or XMI.
- ***<topic>*** is optional and refers to the topic covered by the document, for instance, “Overview” for this document or “AMF” for the Availability Management Framework.
- ***<relCode>*** is a single capital ASCII letter referring to the release code version. Note: this is different from a release number, which refers to a set of documents released together.
- ***<majVersion>*** is the major version number.

- ***<minVersion>*** is the minor version number.

Examples:

- **SAI-Overview-B.05.01**
- **SAI-HPI-B.02.01**
- **SAI-AIS-AMF-B.04.01**

Details on the version numbering convention can be found in [1].

The interface specification documents provide the type and the function declarations defined in the 'C' programming language syntax. Details on the programming model and naming conventions for the AIS are given in [1]. Additionally, the AIS Services include a UML description of their information model, a service administration API, and a description of the service alarms and notifications.

The AIS Frameworks include a description of their system models.

Java Mappings of AIS Services are published in a collection of files generated with the Javadoc tool (see [32]), and which can be read with a standard browser. Also included in the Javadoc files is an overview of the Java mapping covering the design of the Java APIs, the Java programming model, and a comparison with the C APIs. To facilitate implementations of the Java API's the corresponding Java source code is also supplied.

The naming convention for files published as part of an SA Forum release is as follows:

<documentType>-<interface>-{<area> | R<releaseNumber>}-<fileContents>-<relCode>.<majVersion>.<minVersion>.<ext>

where:

- ***<documentType>*** is either **SAI** for an interface or **SAIM** for an interface mapping.
- ***<interface>*** is one of
  - **HPI** for the Hardware Platform Interface,
  - **AIS** for the Application Interface Specification, or
  - **IM** for the Information Model.
- ***<area>*** refers to the service or framework area that the file is part of.

- ***<releaseNumber>*** is the release number of the complete set of documents and files that are published together.
- ***<fileContents>*** refers to the content of the file, which is one of
  - **CH** for C header files,
  - **JC** for Java code,
  - **JD** for Java documentation,
  - **XMI** for XML Metadata Interchange information, or
  - **XSD** for XML Schema Definition information.
- ***<relCode>*** is a single capital ASCII letter referring to the release code version of the file.
- ***<majVersion>*** is the major version number.
- ***<minVersion>*** is the minor version number.
- ***<ext>*** is the file extension, which is one of
  - **zip** for a compressed archive file,
  - **xsd** for an XML Schema Definition file,
  - **xml** for an XML file, or
  - **xml.zip** for a compressed archive of an XML file.

The ***<area>*** and ***<releaseNumber>*** fields are mutually exclusive. When ***<area>*** is used, the version code comprising ***<relCode>.<majVersion>.<minVersion>*** refers to the version of the area independently from the release number. When ***<releaseNumber>*** is used, the version code refers to the version of the file within the ***<releaseNumber>***. When neither are used, the version code refers to the version of the interface.

Examples:

- **SAI-AIS-R6-CH-A.01.01.zip** AIS C Header files.
- **SAI-IM-XMI-A.04.01.xml.zip** SA Forum Information Model.
- **SAI-AIS-IMM-XSD-A.01.01.xsd** describes the XML schema of the IMM initial file.
- **SAIM-AIS-R5-JD-A.01.01.zip** contains the Javadoc of the Java Mapping.
- **SAIM-AIS-R5-JC-A.01.01.zip** contains Java code of the Java Mapping.

3.2 Service Availability Interface Specifications Architecture

The Service Availability™ Interface specifications were developed with a number of objectives in mind:

- to separate hardware from software concepts to allow for the independent evolution and management of hardware, middleware, and applications,
- to provide common abstractions for hardware and software resources for the purposes of high availability,
- to enable the open integration of commercial off-the-shelf (COTS) components into a highly available system by factoring out common availability functions into a modular set of frameworks and services,
- to focus on interfaces rather than on protocols to ensure portability and common APIs while allowing for innovation and differentiation in the implementation of the middleware, and
- to provide a simple programming model that covers a broad spectrum of systems and applications.

3.2.1 Basic Architecture

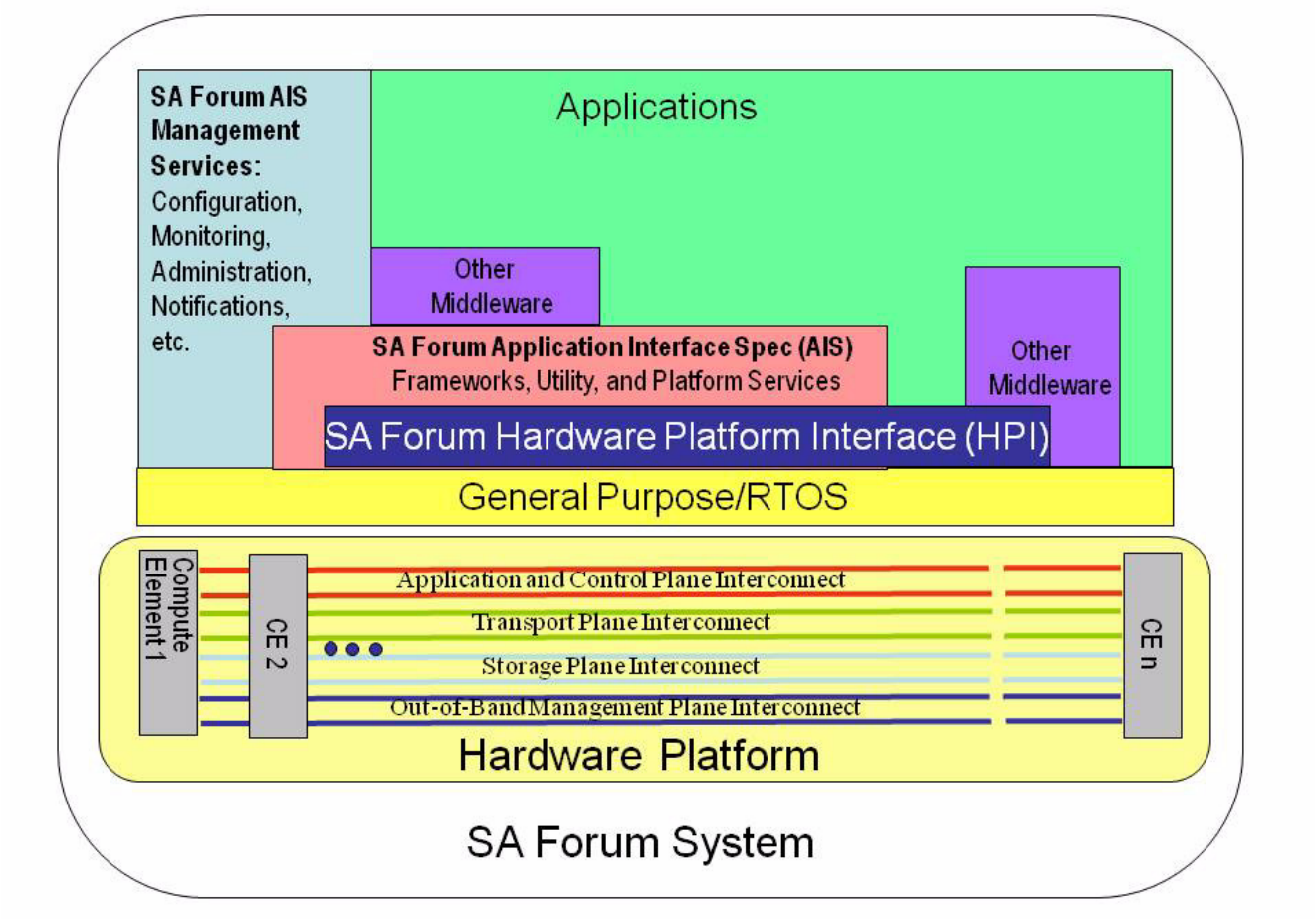
The Service Availability™ interfaces are conceived to work together within the scope of a system, which, in turn, is comprised of a set of hardware and software resources. Within the scope of a single system, the hardware resources can be seen as a set of independent interconnected computing elements. Among these are a set of computing elements capable of hosting one or more instances of a general purpose or real-time operating system on which the various software elements reside. The hardware platform together with the operating systems and middleware constitutes a distributed system.

Each independent computing element in the system may host one or more of the following software functions:

- middleware providing the Hardware Platform Interface
- middleware providing the AIS Framework, Platform, and Utility Services
- middleware providing the AIS Management Services
- other middlewares
- applications

The basic architecture of a SA Forum system with its constituent elements is depicted in FIGURE 1.

FIGURE 1 Basic Architecture of an SA Forum System



The degree of connectivity within each of the interconnect planes may vary from system to system. The set of operating system instances in the system that are configured to run service availability middleware and applications are referred to as the set of configured cluster nodes.

A cluster is configured on an interconnected hardware platform comprising one or more racks within a single site. Geographically distributed clusters are for further study. In general, it is assumed that the individual software components of the AIS Services on each node communicate with entities on other nodes via the control plane interconnect. In this case, for a correctly installed and configured system, there should be complete (and redundant) control plane connectivity between all configured cluster nodes.

3.2.2 Introduction to the Interfaces

3.2.2.1 Hardware Platform Interface

The HPI specification is comprised of a set of fine-grained APIs that provide low-level access to the hardware resources of the platform. Among the capabilities defined are

- discovery and inventory of HPI accessible hardware,
- configuration monitoring and control, and
- sending hardware related events and maintaining event logs and alarm tables.

3.2.2.2 Application Interface Specification

The Application Interface Specification is comprised of 12 services and two frameworks. The services are classified into 3 functional groups: platform related services, basic management infrastructure services, and general utility services. The frameworks comprise the fourth functional group.

Note: When the term 'AIS Services' is used without a further qualifier, it refers to all the services and frameworks that comprise the Application Interface Specification"

3.2.2.2.1 AIS Platform Services

These services provide a higher-level abstraction of the hardware platform and operating systems to the other AIS Services and applications:

- **Platform Management Service**— It provides a convenient abstraction for controlling configured hardware and low level software (HW/OS) resources. It defines APIs for monitoring state changes in application-defined groups of configured resources.
- **Cluster Membership Service**— It provides a consistent view of the healthy nodes in a cluster.

3.2.2.2.2 AIS Management Services

These services provide the basic standard management interfaces that should be used by the implementation of all AIS Services and applications:

- **Information Model Management Service**— It provides APIs for defining, obtaining, manipulating, and exposing configuration and runtime management information as well as for invoking administrative commands on managed objects.
- **Notification Service**— It provides data structures and APIs for notifying alarms, state changes, object life cycle changes, attribute value changes, security alarms, and other particular events (notifications of miscellaneous type).

- **Log Service**— It allows for cluster-wide logging of alarms, notifications, system messages, and application-defined log streams.
- **Security Service**— It provides access security functions for the AIS Services and HPI.

3.2.2.2.3 AIS Frameworks

- **Availability Management Framework**— It provides functions for availability management of applications and middleware.
- **Software Management Framework**— It is used for managing middleware and application software during hardware, operating system, middleware, and application upgrades while taking service availability into account.

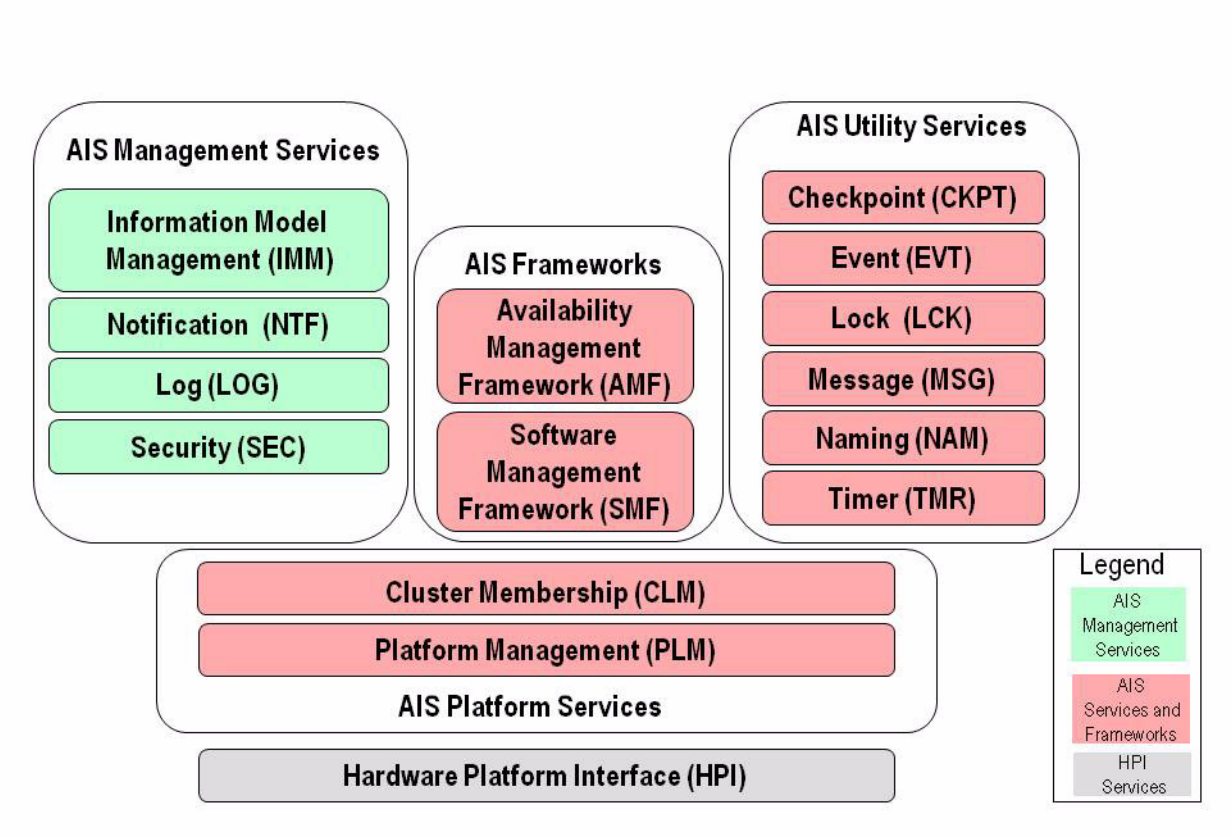
3.2.2.2.4 AIS Utility Services

These services provide some of the common interfaces required in highly available distributed systems, including:

- checkpointing,
- event distribution,
- distributed locks,
- message passing,
- name lookup, and
- local timer service.

A high-level view of HPI and AIS Services is depicted in FIGURE 2.

FIGURE 2 Overview of HPI and AIS Services



3.2.3 Service Dependencies

The different services and frameworks of the interface specifications have been designed to be modular and, to a certain degree, independent of one another. Thus, it is conceivable to have a system providing only HPI and no AIS and vice versa. The only hard and fast architectural dependency is the dependence on the Cluster Membership Service (CLM). All AIS Services with the exception of the Platform Management Service (PLM) and the Timer Service (TMR) depend on CLM. In general, all AIS Services should use the AIS Management Services for exposing their administrative interfaces, configuration, and runtime management information. FIGURE 3 shows some typical dependencies between the different services.

PLM is designed to work with HPI in that the objects in the PLM configuration that point to HPI entities are used to verify which configured resources have been discovered by HPI and are thus present.

CLM is designed to work with PLM and AMF to provide higher levels of service availability. For example, by propagating pending node removal events from PLM to AMF, the AMF can perform a graceful switch-over before the removal is performed.

The SA Forum does not impose any requirements on implementations of the interface specifications. In a typical installation, HPI will generally be used to discover the hardware resources present in the system. There may be one or more implementations or instances of the HPI middleware that provide the hardware platform interface for the system.

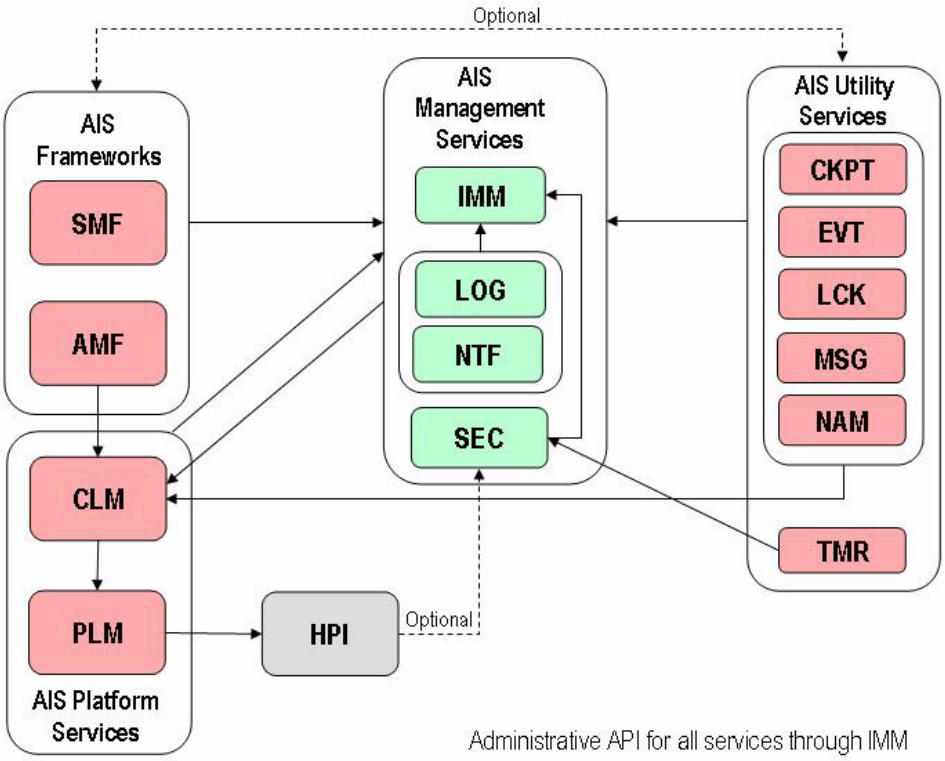
PLM matches its hardware elements configured in the information model with the discovered hardware resources exposed by HPI. It is assumed that there is a single instance of the platform management middleware providing the PLM Service running in the system. The cluster configuration of CLM in the information model consists of the information specific to each of the configured member nodes. The configuration of each CLM member node refers to the execution environment (in the PLM configuration) in which the node will be hosted.

The configuration information of the execution environment directly or indirectly refers to the hardware element hosting it (see FIGURE 6). The CLM implementation will generally use PLM to help determine the presence and state of health of its configured member nodes.

For Release 6, it is assumed in the specifications that there is only one instance of the IMM and CLM Services in the system.

In some cases, dependencies are mutual, and it is up to the implementations to resolve such issues. For example, CLM depends on IMM to obtain its configuration information while IMM depends on CLM to know what level of service to provide on a particular node depending on whether it is a member node or not. A more extreme example is if the Checkpoint Service is modeled as an AMF application (see Section 3.3), and it is therefore dependent on the AMF, but the AMF implementer may also wish to use the Checkpoint Service to replicate its state in the cluster. The service dependencies are depicted in FIGURE 3.

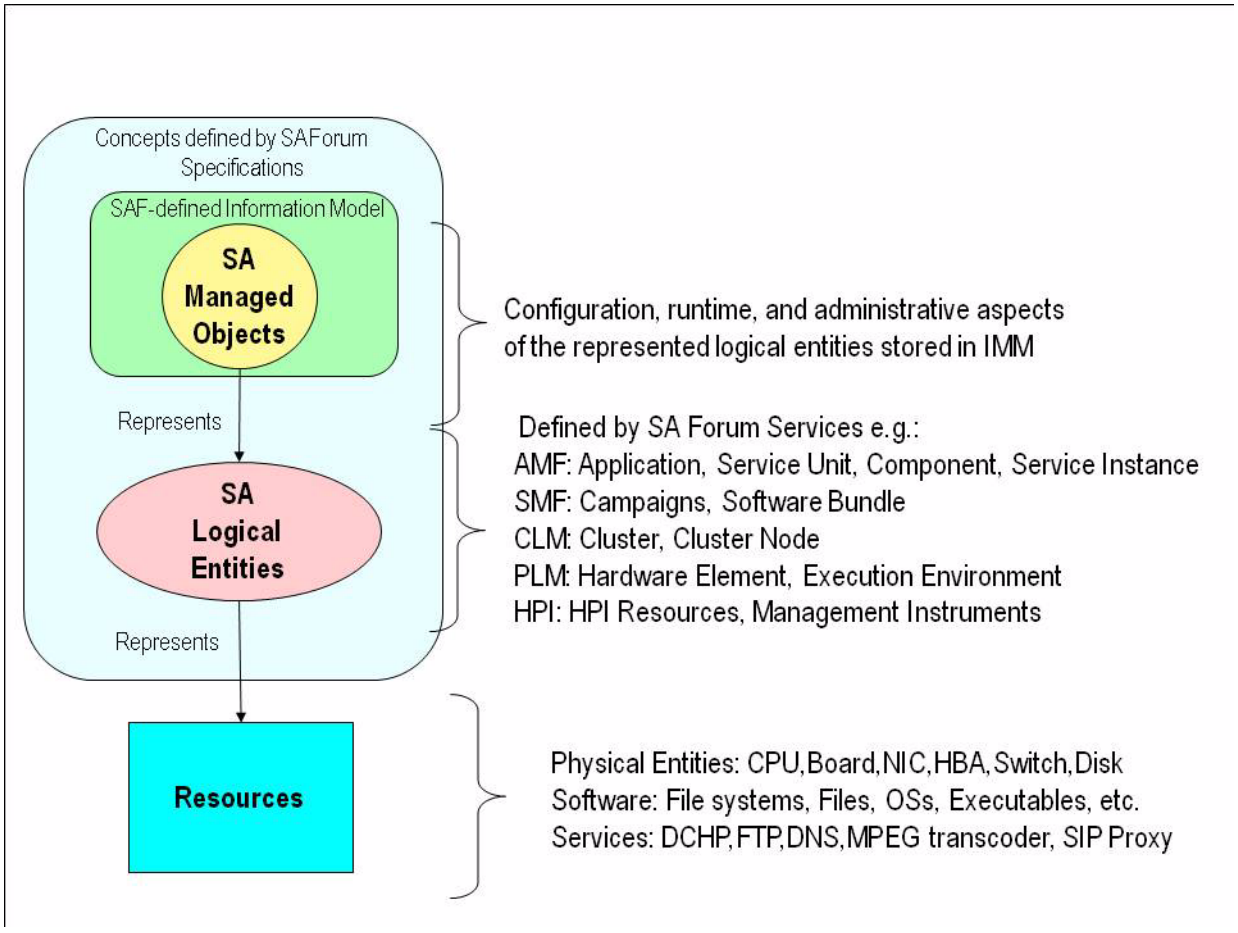
FIGURE 3 Service Dependencies



3.3 Information Modeling

Some of the resources in an SA Forum system are represented by logical entities defined by the various services and frameworks. These logical entities are the software abstractions that are directly implemented by the respective services and frameworks. In turn, some of these logical entities are represented by managed objects in the SA Forum-defined Information Model. This information model consists of the configuration and runtime managed objects defined by each service or framework together with their naming hierarchy (see Chapter 9) and administrative operations. The representation of resources and logical entities is illustrated in FIGURE 4.

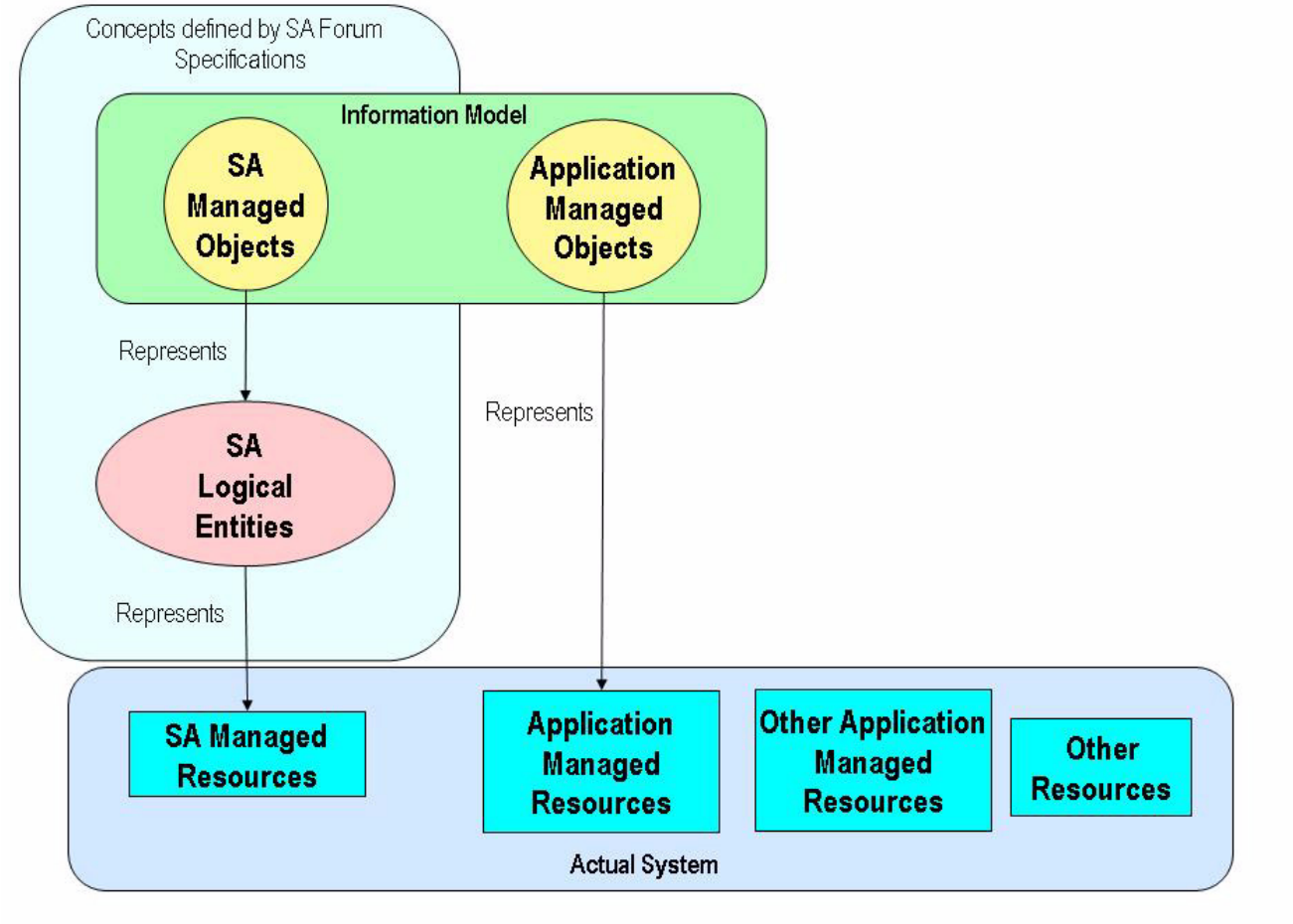
FIGURE 4 Information Model Basic Concepts



The information model exposed by the IMM Service in a given system may also contain application-defined managed objects that represent application-managed resources. There may also be other application-managed resources in the system that are not represented in the model exposed by the IMM Service.

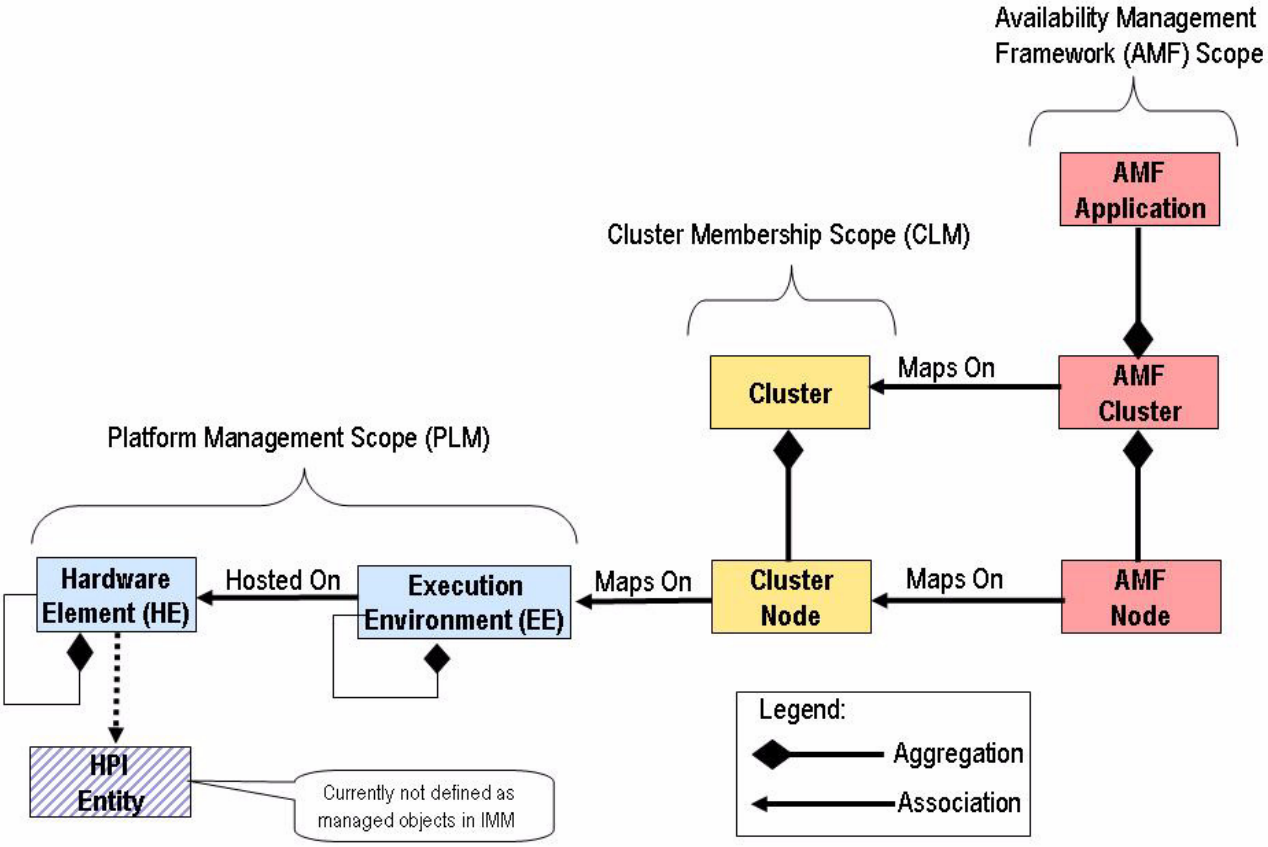
FIGURE 5 shows the various ways in which the resources of an actual system may be represented in the model.

FIGURE 5 Information Model Concepts in a System Context



A high-level subset of the SA Forum Information Model showing the relationships between some of the managed objects defined by PLM, CLM, and AMF is shown in FIGURE 6. Further details can be found in Chapter 9.

FIGURE 6 SA Forum Information Model Cluster View



3.4 Modeling of AIS Services

The SA Forum AIS does not specify any particular implementation of the various AIS Services. However, the SA Forum strongly recommends using the system modeling abstractions and logical entities that are made available by the Availability Management Framework specification when implementing such services. This promotes a single and unified AIS modeling scheme (based on Availability Management Framework logical entities) for AIS Services and SA Forum applications, that is, AIS Services are modeled, managed, and upgraded in the same way as any other SA Forum application would be modeled, managed, and upgraded.

3.5 Java Mappings

In addition to the 'C' syntax interface specifications, the Service Availability™ Forum provides a mapping of the specified interfaces to the Java language. Thus, a set of packages define a Java API which provides access to high availability middleware implementations that are compliant with the AIS. This release comprises mappings for the following AIS Services:

- Availability Management Framework (AMF)
- Cluster Membership Service (CLM)
- Information Model Management Service (IMM)
- Event Service (EVT)
- Notification Service (NTF)

The AIS Service API definitions in Java correspond very closely to the C language APIs for AIS. These definitions are designed such that:

- the implementation language of the server code for the AIS Services is independent of the client library language implementations;
- it should be possible to mix-and-match C and Java components in the same SA Forum cluster;
- there is no more and no less functionality in the Java API compared to the C API;
- the semantics of the Java interfaces are generally identical to the corresponding C interfaces;
- no particular Java application model is assumed; the API can be utilized by any Java SE or Java EE application environments;
- the Java APIs requires Java 5 or later versions.

For details, see the overview section in [21].

As supportive material, the Java code is provided in [22].

3.6 Virtualization

Virtualization is becoming necessary to ensure that systems can scale from both economic and technology perspectives in today's computer and clustered computing environments when exploiting multi-core processors. This is mainly due to the fact that the single-thread performance of modern microprocessors is not increasing while their throughput is being improved by increasing the number of processing cores they contain. Though the use of multi-core processors and virtualization creates a very appealing picture showing great flexibility in the configuration and re-configuration of complex systems, it is not necessarily always appropriate for HA systems.

High availability is primarily achieved by building systems that have:

- no single point of failure, that is, they have redundancy in all parts,
- standby functionality (active or inactive) that can take over (2N, N+1, N+M, ...),
- multiple (isolated) instances of the required functionality such that requests can be re-routed, and
- relatively small units of failure.

With multi-core and/or virtualization, it is possible to host multiple execution environments on a single hardware element, thereby affecting the formulas used to calculate and ensure availability in a complete system due to the relatively large failure unit represented by the multi-core processors on a single Field Replaceable Unit (FRU).

Virtualization technology by itself does not solve the problem of high availability.

Initially, the SA Forum specifications did not contain any explicit support for virtualization, even though a basis did exist such as the possibility to define nested FRUs at the HPI level. The change in this release that a node in the Cluster Membership Service is no longer mapped onto a physical node but onto an execution environment defined in the Platform Management Service has made it possible to use multi-core CPUs as well as multi-socket boards or DSP server farms and still have some knowledge of how these resources are mapped to one another.

The new Platform Management Service specification, by defining the object containment relationships directly in the information model, ensures that there is a clearer and direct separation of node and physical hardware while it still provides an easy way for the management functionality to understand which nodes are hosted on the same physical hardware element. This needs to be handled as a specific case to ensure that redundant services are not allocated to the same failure unit / hardware element.

However, the current specifications do not (yet) address the full exploitation of this dependency information in the Availability Management Framework. Nevertheless, it is still possible to make use of it in an implementation. It can also be made use of when constructing configuration files by making sure that redundant service units, checkpoint replicas, and all N-way active redundant components are not instantiated on the same physical FRU or hardware element.

4 Overview of the Hardware Platform Interface (HPI)

The Hardware Platform Interface (HPI) allows applications and middleware ("**HPI User**") to access and manage hardware components via a standardized interface. Its primary goal is to allow for portability of HPI User code across a variety of hardware platforms. The ability to monitor and control these systems is provided through a consistent, platform-independent set of programmatic interfaces. The HPI Specification provides data structures and functional definitions that can be used to interact with manageable subsets of a platform or system.

The HPI model includes the following basic concepts -- Entities, Management Instruments, Resources, Domains, and Sessions.

HPI Entities represent the physical components of the system. Each entity has a unique identifier, called an entity path, which is defined by the physical component's location in the physical containment hierarchy of the system.

Management Instruments model an entity's management capabilities. They are contained in one or more resources. These management instruments and management capabilities are the mechanisms by which HPI Users can control and receive information about the state of the system. Examples are Sensors, Controls, Inventory Data Repositories, Timers, Annunciators, Diagnostics Initiators, and Firmware Upgrade Management Instruments.

HPI Resources provide management access to the entities within the system. Frequently, HPI resources represent local control processors used for management of the entity's hardware. Each HPI resource is responsible for presenting a set of management instruments and management capabilities to the HPI User. HPI resources may be dynamically added and removed in a system, such as when hot-swappable Field Replaceable Units (FRUs) which include management capabilities are physically added and removed.

Domains provide access to sets of resources. Each domain also provides information about the resources that are accessible through that domain. Many systems may have only a single domain, whereas systems that have areas dedicated to separate tasks, for example, may manage these areas through separate domains.

Sessions provide all access to an HPI implementation by HPI Users. An HPI session is opened on a single domain; one HPI User may have multiple sessions open at a time, and there may be multiple sessions open on any given domain at a time.

HPI is a generic interface allowing the representation of the underlying hardware in a way chosen by the vendor. The HPI User application needs to discover the structure of the hardware before it starts managing it. However, it is advantageous to create a common model describing the hardware for a class of similar systems. This allows to implement the HPI User application such that it 'knows' in advance what the hardware is capable of doing. Such models are called mapping of a hardware structure to its HPI representation. Currently, a mapping specification is available for PICMG® AdvancedTCA and MicroTCA systems (xTCA Mapping Specification), see [24].

The xTCA Mapping Specification defines how to represent in HPI hardware elements mandated in the PICMG® specification. The examples are Management Instruments (Sensors, Controls, Firmware Upgrade Management, etc.), representation of the xTCA hot-swap state machine using the HPI hot-swap state machine, or definition of how to use some specific HPI elements like Resources and Domains in the context of xTCA. The mapping specification does not restrict the management capabilities to those mandatory in all xTCA systems, but it leaves space for additional functionality as defined by vendors. In sum, representing xTCA systems according to the mapping specification allows for even better portability of HPI User applications with rich functionality between different vendors' systems.

5 Overview of the AIS Platform Services

The AIS Platform Services are comprised of the Platform Management Service (PLM) and the Cluster Membership Service (CLM).

PLM provides a simple and convenient view of the hardware platform to CLM and other application components. PLM can be used by CLM to monitor the state of all hardware and software elements required for the proper functioning of its configured cluster nodes. Dependencies outside of the direct containment relationship exposed by PLM must be managed by the CLM implementation by creating and monitoring the necessary configuration-dependent PLM entity groups.

CLM is used by all cluster-aware AIS Services (except PLM and the Timer Service) and by applications to monitor and obtain the current cluster membership, which is always a subset of the configured cluster nodes.

Availability Management Framework implementations in particular can use the CLM for availability management of services running on cluster nodes.

5.1 Platform Management Service

The Platform Management Service (PLM) provides a logical view of the hardware and low-level software of the system. Low-level software in this sense comprises the operating system and virtualization layers that provide execution environments for all kinds of software.
This logical view is presented in the Service Availability™ Forum Information Model by a set of objects that

- allow for the management of hardware entities and execution environments,
- allow other software to track status changes of hardware and execution environments, and
- allow the mapping of the HPI (see [23]) view onto objects represented in the information model.

The PLM Service typically uses HPI to derive all necessary information from the hardware.

The PLM Service not only provides the hardware information in the Service Availability™ Forum Information Model through the IMM Service (see [3]), but it also provides objects that are administratively configurable. Additionally, the PLM Service is responsible for matching the configuration with the discovered hardware.

The main logical entities implemented by the PLM Service are:

Execution Environment (EE)

An execution environment is a logical entity that represents an environment capable of running some software programs. An execution environment may or may not host a CLM cluster node. In most cases, a CPU blade or an SMP machine runs a single operating system instance modeled as a single execution environment. If a hypervisor provides hardware virtualization, the hypervisor itself and each operating system running under its control are modeled as separate execution environments.

Hardware Element (HE)

A hardware element is a logical entity that represents any kind of hardware entity, which can be, for instance, a chassis, a CPU blade, or an I/O device. Typically, all FRUs (Field Replaceable Units) are modeled as hardware elements. If necessary, the system architect may model in the PLM Information Model additional entities which are part of a FRU as hardware elements, for example, I/O ports, CPU cores, and so on.

The PLM Service maintains the state information of hardware element entities. For this purpose, it retrieves as necessary any information about the health of the hardware. The PLM Service may also map HPI events to notifications distributed by the Service Availability™ Forum Notification Service (see [2]) and generate these notifications.

Similarly, the PLM Service retrieves all necessary information about the health of the operating system and any available virtualization layer to maintain states of execution environment entities and generate necessary notifications about events of the execution environment entities.

The PLM Service allows application processes to register a callback function to receive notifications when PLM Service entities start or stop to provide service. This mechanism also allows application processes to gracefully shut down their own services when a PLM Service entity is about to terminate, for instance, when the extraction of an HE is pending or when an EE is being shut down due to an administrative command.

5.2 Cluster Membership Service

The Cluster Membership Service provides applications with membership information about the nodes that have been administratively configured in the cluster configuration (these nodes are also called cluster nodes or configured nodes) and is core to any clustered system. A cluster consists of this set of configured nodes, each with a unique node name.

The two logical entities implemented by the Cluster Membership Service are:

Cluster—It contains, amongst other items, the name of the cluster and is the parent object of the cluster node objects.

Cluster Node—It contains, amongst other item, the name of the cluster node, its node ID, the object reference of the hosting PLM execution environment, its membership state, administrative state, and communication addresses.

A member node is a configured node that the Cluster Membership Service has recognized to be healthy and well-connected to the rest of the cluster in order to host HA applications and services. The set of member nodes at a point in time is referred to as the cluster membership or simply as membership. The Cluster Membership Service is the authority that decides whether a configured node is transitioned to be a member node of the cluster. The Cluster Membership Service must ensure that only a single cluster is formed from the set of configured nodes. It is implementation-specific whether the Cluster Membership Service can still satisfy this requirement in the presence of interconnect failures.

The Cluster Membership Service provides APIs to coherently obtain the membership in blocking and non-blocking modes. Information on individual configured nodes can also be obtained through the AIS Management Services, but these services do not necessarily provide a coherent view of the membership at a given point in time.

The Cluster Membership Service also allows application processes to register a callback function to receive membership change notifications as those changes occur. Tracking can be fine grained or coarse grained. The callback function provides correlation identifiers allowing the change to be associated with a root cause.

6 Overview of the AIS Management Services

The AIS Management Services are comprised of the Information Model Management (IMM) Service, the Notification (NTF) Service, the Log (LOG) Service, and the Security (SEC) Service. The IMM, NTF, and LOG Services together provide essential management services to both the AIS middleware as well as to applications. They also provide the necessary APIs to manage all resources represented by the managed objects in the IMM Information Model. Applications and AIS Services expose in the information model their configuration and runtime management information as well as their administrative operations. Their notifications are logged using the LOG Service, which can also be used separately.

The SEC Service is currently only defined for authorizing internal access to the AIS and HPI Services.

6.1 Information Model Management Service (IMM)

The different logical entities of an SA Forum system, such as HEs and EEs managed by the PLM Service, components managed by the Availability Management Framework, checkpoints provided by the Checkpoint Service, or message queues provided by the Message Service are represented by the various managed objects of the SA Forum Information Model.

The SA Forum Information Model (IM) is specified in UML and managed by the Information Model Management (IMM) Service.

The managed objects in the SA Forum Information Model are defined by their class, which determines the number and type of their attributes as well as the administrative operations that can be performed on them. Each class has a unique attribute used for naming the object in the object hierarchy. There are two types of managed objects:

- configuration objects, which contain configuration attributes and optionally runtime attributes and
- runtime objects, which contain only runtime attributes.

Administrative operations can be defined for both types of managed objects.

The IMM Service exposes two sets of APIs:

(1) An Object Management API (OM-API), exposed typically to system management applications (for example, management agents).

(2) An Object Implementer API (OI-API) restricted to Object Implementers, which are typically part of the implementation of the managed services or resources.

The Object Management API includes functions to

- manage the life cycle of object class definitions,
- manage the life cycle of configuration objects,
- search, access, and manage objects, and to
- group a set of configuration requests together into a Configuration Change Bundle (CCB), so that either all or none of the requests in the bundle will be executed.

The Object Implementer API includes functions for an object implementer to

- associate itself with an object class or subtree of the information model,
- manage the life cycle of runtime objects,
- receive callbacks on the creation, modification, and deletion of its objects,
- receive callbacks when defined administrative operations are invoked on its objects and provide the result of the operation to the invoker, and to
- participate in the execution of configuration change bundles either as the sole implementer or as a validator and/or as an applier of CCB requests for its objects.

The IMM Service also provides an administrative command to export the contents of its information model in a file conforming to the IMM XML Schema Definition (see [13]). Upon initial startup, the IMM Service can be configured to either initialize from its most recent usable persistent store or from a file containing an application-generated or IMM Service-exported configuration file.

6.2 Log Service (LOG)

The SA Forum distinguishes between a Log Service and a Trace Service (the latter service is currently not yet specified). The former service is for cluster-wide, function-based information suited for system administrators or automated tools, whereas the latter is low-level implementation-specific information suited for developers or field engineers.

The Log Service enables applications to express and forward log records through well-known log streams that lead to particular output destinations such as a named file. Once at the output destination, a log record is subject to configurable and public-output formatting rules. Since the output format is public, third party tools can read these log files.

The Log Service defines four types of log streams:

- the alarm log stream for ITU X.733 [29] and ITU X.736 [30] based log records,
- the notification stream for ITU X.730 [27] and ITU X.731 [28] based log records,
- the system stream is for system-relevant log records, and
- the application stream is for application-specific log records.

For each of the alarm, notification, and system log stream types, there is exactly one log stream in an SA Forum cluster. However, any number of application log streams can coexist, each with a unique name that can come and go as needed by running applications.

6.3 Notification Service (NTF)

The Notification Service is—to a great degree—based on the ITU-T Fault Management model (as found in the X.700 series of documents) as well as on many other supportive recommendations.

The Notification Service is centered around the concept of a notification, which explains an incident or change in status. The term ‘notification’ is used instead of ‘event’ to clearly distinguish it from ‘event’ as defined by the AIS Event Service.

The Notification Service defines six notification types with distinct parameters. They are:

- Alarm
- Security Alarm
- Object Creation/Deletion
- State Change
- Attribute Value Change
- Miscellaneous

The Notification Service is based on a publish/subscribe paradigm.

Any number of notification producers can publish notifications.

Notification consumers can be of two types:

- Notification subscribers, which receive selected notifications as they occur.
- Notification readers, which retrieve historical notification entries from the persistent notification log.

Any number of notification consumers of each type can exist at a point in time.

AIS Services that generate notifications have a section in their specification that describes these notifications. They are expressed using the notification producer API syntax and semantics as specified in the Notification Service. The expectation is that an alarm correlator, an element manager, or a management subagent in the cluster would subscribe for notifications (using the notification subscriber API) in which they are interested.

6.4 Security Service (SEC)

The SA Forum Security Service provides mechanisms that can be used by AIS Services to authorize AIS Service client processes within the cluster to perform particular activities. These mechanisms can be used to preserve the integrity of the high availability infrastructure and of SA Forum applications, including their data, by protecting against unauthorized access.

The enforcement of security is delegated to the implementation of the AIS Services. Each security-enabled AIS Service must request authorization from the Security Service on behalf of the client processes of the AIS Service when these clients access the service or invoke the various capabilities the AIS Service provides. The Security Service responds to these authorization requests with a granted or denied indication, and it is then up to the AIS Service to allow or disallow access or capability invocation accordingly.

AIS Services must be prepared to adapt to changes in the security policy. Upon changes in the security policy, the Security Service informs its subscribers about those changes using appropriate callbacks, which are registered at Security Service initialization.

Note: *As it was not possible to update all AIS Service specifications on time for the SA Forum Security Service release, the current release of SA Forum Security Service only supports the UID authentication mechanism for AIS Service client processes' authentication. Further updates of the various AIS Service specifications will enable support for other authentication mechanisms like Private Key Infrastructure (PKI) and shared secrets.*

7 Overview of the AIS Frameworks

The AIS Frameworks are comprised of the Availability Management Framework (AMF) and the Software Management Framework (SMF).

The Availability Management Framework provides a comprehensive and generic system model for structuring highly available services and the applications that implement these services, including most middleware. It provides APIs for carrier-grade availability-aware applications that interact closely with the framework as well as command line interfaces for managing nonavailability-aware applications through configuration alone.

A key concept of the Availability Management Framework system model is the separation of the instance of a software component executing on a node from the role it plays in providing its service. A given instance of a software component providing its service in the standby role can be called upon by the Availability Management Framework at some later time to provide its service in the active role while calling upon another (redundant) component to take on the role of providing the same service in the standby role.

The Software Management Framework complements the Availability Management Framework by providing a consistent and reliable framework for delivering and upgrading software (and hardware) in a SA Forum system. The Software Management Framework system model is comprised of software bundles describing the contents and dependencies of the software to be deployed or upgraded and upgrade campaigns that structure the upgrade procedures.

7.1 Availability Management Framework (AMF)

The Availability Management Framework is the software entity that enables service availability by coordinating other software entities within a cluster.

The Availability Management Framework provides a view of one **logical cluster**, which consists of a number of cluster nodes. These nodes host various resources in a distributed computing environment.

The Availability Management Framework provides a set of APIs to enable highly available applications. In addition to component registration and life cycle management, it includes functions for error reporting and health monitoring. The Availability Management Framework also assigns active or standby workloads to the components of an application as a function of component state and system configuration. The Availability Management Framework configuration allows prioritization of resources and provides for a variety of redundancy models. The Availability Manage-

ment Framework also provides APIs for components to track the assignment of work or so-called component service instances among the set of components protecting the same component service instance.

7.2 Software Management Framework (SMF)

An SA Forum system can be characterized by the deployment configuration, which consists of the software deployed in the system along with all configured software entities. The deployment configuration constitutes an essential part of the information model managed by IMM ([3]).

The Software Management Framework maintains the information model that describes the availability and deployment of software in an SA Forum cluster and allows for the evolution of a live system by orchestrating the migration from one deployment configuration to another. This migration process is often referred to as an upgrade.

The Software Management Framework migrates the system from one deployment configuration to a new desired one based on the campaign specification provided in the form of an XML file. During this migration, the Software Management Framework (a) maintains the campaign state model, (b) monitors for potential error situations caused by the migration, and (c) deploys error recovery procedures as required. To accomplish all these tasks, the Software Management Framework interacts with the Availability Management Framework in order to maintain availability and with other AIS Services and the SA Forum HPI as necessary.

The Software Management Framework also provides an API for client processes to register their interest in receiving callbacks when a relevant upgrade is initiated in the cluster and as the upgrade progresses through significant milestones.

8 Overview of the AIS Utility Services

The following AIS Utility Services provide functionality of the cluster on which the AIS Frameworks and the highly available applications can be implemented:

- Checkpoint Service
- Event Service
- Lock Service
- Message Service
- Naming Service
- Timer Service

Each of these services is briefly described in the following subsections.

8.1 Checkpoint Service

The Checkpoint Service provides a facility for processes to record checkpoint data incrementally, which can be used to protect an application against failures. When processes recover from a failure (with a restart or a fail-over procedure), the Checkpoint Service can be used to retrieve the previous checkpoint data and resume execution from the state recorded before the failure, thus minimizing the impact of the failure.

Checkpoints are cluster-wide entities. A copy of the data stored in a checkpoint is called a checkpoint replica, which is typically stored in main memory rather than on disk for performance reasons. A checkpoint may have several checkpoint replicas stored on different nodes in the cluster to protect it against node failures.

8.2 Event Service

The Event Service is a publish/subscribe multipoint-to-multipoint communication mechanism that is based on the concept of event channels: One or more publishers communicate asynchronously with one or more subscribers by using events over an event channel. Event channels are cluster-wide named entities.
Publishers can also be subscribers on the same event channel.

Events consist of a standard header and zero or more bytes of published event data. The Event Service API does not impose a specific layout for the published event data.

8.3 Lock Service

The Lock Service is a distributed lock service, which is intended for use in a cluster where processes in different nodes might compete with each other for access to a shared resource.

The Lock Service provides entities called lock resources, which application processes use to coordinate access to shared resources.

The Lock Service provides a simple lock model supporting one locking mode for exclusive access and another one for shared access.

8.4 Message Service

The Message Service specifies APIs for a cluster-wide interprocess communication system based on the concept of message queues. Processes communicate by sending messages to a given message queue and by retrieving messages from it. A single message queue can have multiple sending processes but at most one receiving processes at any given time. The single message queue thus supports point-to-point or multi-point-to-point communication patterns. Message queues are persistent or non-persistent. The Message Service must preserve messages that have not yet been consumed when the message queue is closed.

Processes sending messages to a message queue are unaware that the process that was originally receiving these messages may have been replaced by another process during a fail-over or switch-over.

Message queues can be grouped together to form message queue groups. Message queue groups permit multipoint-to-multipoint communication. They are identified by logical names, so that a process is unaware of the number of message queues and of the physical location of the message queues to which it is communicating. The sender process addresses message queue groups by using the same mechanisms that it uses to address single message queues. The message queue groups can be used to distribute messages among message queues pertaining to the message queue group. Regardless of the number of message queues to which messages are distributed, the message queue group remains accessible under the same name.

Message queue groups can be used to maintain transparency of the sender process to faults in the receiver processes, represented by the message queues in the message queue groups. The sender process communicates with the message queue group. If a receiver process fails, the sender process continues to communicate with the message queue group and is unaware of the fault, because it continues to obtain service from the other receiver processes.

8.5 Naming Service

The Naming Service provides a mechanism by which human-friendly names are associated with (‘bound to’) objects, so that these objects can be looked up given their names. The objects typically represent service access points, communication end-points and other resources that provide some sort of service.

The Naming Service imposes neither a specific layout nor a convention on either the names (UTF-8 encoding assumed) or the objects to which they are bound. It allows the users of the service to select and use their own naming schema without assuming any specific hardware or logical software configuration. The clients of the Naming Service are expected to understand the structure, layout, and semantics of the object-bindings they intend to store inside and retrieve from the service.

8.6 Timer Service

The Timer Service provides a mechanism by which client processes can set timers and get notified when a timer expires. A timer is a logical object that is dynamically created and represents either absolute time or a duration.

The Timer Service provides two types of timers: single event timers and periodic timers. Single event timers will expire once and are deleted after notification. Periodic timers will expire each time a specified duration is reached, and the process is notified about the expirations. Periodic timers have to be explicitly deleted by invoking a timer deletion function.

9 SA Forum Information Model

The **SA Forum Information Model** is described in UML using the XML Metadata Interchange (XMI) v2.1 format (see [12]). The SA Forum Information Model has been organized as UML class diagrams.

Some UML diagrams provide an overview showing how various object classes relate to each other. Other diagrams show the attributes and administrative operations of individual object classes.

This section presents the following overview diagrams. Their names in [12] are:

- 1- Cluster View (see Section 9.3.1)
- 3.2- AMF Instances View (see Section 9.3.2)

These two preceding UML diagrams, the other UML diagrams, and the formats of the Distinguished Names (DNs) of the objects of the information model of all AIS Services are presented in the respective specification documents.

Note that the attributes in the XMI format that contain RDNs (see [1]) contain only the RDN value. The name of the attribute is identical to the RDN type. The corresponding RDN is obtained by concatenating the RDN type and the RDN value using the “=” sign.

Example: The value of the attribute `safAmfCluster` in the `SaAmfCluster` object class is, for instance, `myAmfCluster`.
The corresponding RDN is “`safAmfCluster=myAmfCluster`”.

If an attribute (of type `SaNameT`) refers to the DN, it contains the entire DN (which is composed of RDNs). Thus, it is possible to say that a DN that contains a single RDN is equal to the RDN.

Example: The attribute `saAmfClusterClmCluster` in the `SaAmfCluster` object class contains the DN of the CLM cluster. A possible value of this attribute is "`safClmCluster=myClmCluster`".

Section 9.1 describes the set of rules that must be followed when implementing the classes of the UML model in the Information Model Management Service (IMMS).

Section 9.2 describes rules that were used to represent some relationships among object classes.

9.1 Rules for Converting UML Classes into IMM Classes and Objects

The SA Forum Information Model is implemented within the Information Model Management (IMM) Service. This section describes some general rules used to map UML classes to IMM Service classes and objects. For details on the IMM Service, refer to [3].

9.1.1 IMM Service Class Category

A UML class stereotype is used to indicate the IMM Service class category (`CONFIG`/`RUNTIME`).

9.1.2 Using UML Constraints to Represent IMM Service Attributes

Table 1 presents the IMM Service attribute definitions.

Table 1 Mapping Attribute Characteristics from UML to IMM Service

UML	IMM Service
`CONFIG` constraint	`SA_IMM_ATTR_CONFIG`
`RUNTIME` constraint	`SA_IMM_ATTR_RUNTIME`
`WRITABLE` constraint	`SA_IMM_ATTR_WRITABLE`
`CACHED` constraint	`SA_IMM_ATTR_CACHED`
`PERSISTENT` constraint	`SA_IMM_ATTR_PERSISTENT`
`RDN` constraint	`SA_IMM_ATTR_RDN`
`[1]` or `[1..*]` multiplicity	`SA_IMM_ATTR_INITIALIZED`
`[0..*]` or `[1..*]` multiplicity	`SA_IMM_ATTR_MULTI_VALUE`

9.1.2.1 Multiplicity

The multiplicity of associations is defined to describe valid configurations from IMM perspective as opposed to the operational perspective.

Though the multiplicity should be [1] or [1..*] for entities to provide service (for instance, a service group of the Availability Management Framework should have at least one service instance, an Availability Management Framework cluster should map to one Cluster Membership Service cluster, and so on), the multiplicity is as a rule shown in the form [0..1]] (instead of [1]) and [0..*] instead of [1..*]) to enable a preconfiguration of some objects using the IMM Service (as long as these objects are not used at runtime).

9.1.2.2 Initial Values

If an attribute multiplicity is [0..1] or [0..*], the UML 'Initial Value' provides the attribute default value.

The initial values (defaults) for a particular `CONFIG` attribute x of an object class are specified using one of the two following ways:

1. A specific default value, which is used by the IMM Service when the object class containing x is specified, and the attribute x is not defined or defined with no value.
2. Instead of a specific default value, the name of another attribute y in the same or in another object class can be specified as the default value of x. The value of the attribute y is used in case the object class containing x is specified, and the attribute x is not defined or defined with no value.

The only default notion handled by IMM is the first case. It is the responsibility of the Object Implementer to handle the second case by reading the value of the attribute y if the attribute x is undefined or defined with no value. See an example explaining the second case in Section 9.1.2.4.4.

9.1.2.3 Mapping UML Types to IMM Types

When the UML type of an attribute has no equivalent in the IMM Service, an attribute constraint is used to specify to which IMM Service type the attribute must be mapped (`SAUINT32T` or `SASTRINGT` attribute constraints). Some examples of this usage are presented in the examples of the next section.

9.1.2.4 Examples

9.1.2.4.1 Example 1

For example, the first attribute of the `SaAmfApplication` object class (see [4]) is described as:

`safApp`: `SaStringT` [1] {`RDN`, `CONFIG`}, where

- `safApp` is the attribute name,
- `SaStringT` is the attribute type,
- [1] is the attribute multiplicity and indicates in this case that the attribute has only one value and must be specified (it is mandatory).
- {`RDN`, `CONFIG`} is the list of constraints for this attribute and indicates in this case that the attribute is the object RDN and is a configuration attribute.

As the SaStringT is a type supported by the IMM Service, the attribute will be implemented as an SA_IMM_ATTR_SASTRINGT IMM attribute.

9.1.2.4.2 Example 2

In the same SaAmfApplication object class, the third attribute is described as:

saAmfApplicationAdminState: saAmfAdminStateT [1] {RUNTIME, CACHED, PERSISTENT, SAUINT32T}, where

- saAmfApplicationAdminState is the attribute name,
- saAmfAdminStateT is the typedef defined in the Availability Management Framework specification. The valid values for this attribute are the values of the typedef definition,
- [1] is the attribute multiplicity. This attribute is always present.
- {RUNTIME, CACHED, PERSISTENT, SAUINT32T} is the list of constraints for this attribute and indicates in this case that the attribute is a runtime attribute that is both cached by the IMM Service and persistent. SAUINT32T indicates that this attribute must be implemented as an SA_IMM_ATTR_SAUINT32T IMM attribute.

9.1.2.4.3 Example 3

The third attribute of the SaAmfSG object class is described as:

saAmfSGAutoAdjust: SaBoolT[0..1] = 0 (SA_FALSE) {CONFIG, WRITABLE,SAUINT32T}, where

- saAmfSGAutoAdjust is the attribute name,
- SaBoolT is the typedef defined in [1]. The valid values for this attribute are the values of the typedef definition,
- [0..1] = 0 (SA_FALSE) indicates that this attribute can only take a single value, but that the value is optional. If the value is not defined for this attribute, a default value of 0 must be taken;
- {CONFIG, WRITABLE,SAUINT32T} is the list of constraints for this attribute and indicates in this case that the attribute is a configuration attribute and that its value can be updated dynamically after the object has been created. SAUINT32T indicates that this attribute must be implemented as an SA_IMM_ATTR_SAUINT32T IMM attribute.

9.1.2.4.4 Example 4

The `saAmfCompDisableRestart` attribute in the `SaAmfComp` object class is described as:

`saAmfCompDisableRestart`: `SaBoolT` [`0..1`] = `saAmfCtDefDisableRestart`{`CONFIG`, `WRITABLE`, `SAUINT32T`}, where:

- `saAmfCompDisableRestart` is the attribute name,
- `SaBoolT` is the typedef defined in [1]. The valid values for this attribute are the values of the typedef definition,
- `[0..1] = saAmfCtDefDisableRestart` indicates that this attribute can only take a single value, but that the value is optional. The `saAmfCtDefDisableRestart` represents the default value of the `saAmfCompDisableRestart` attribute as described in the second case of Section 9.1.2.2. `saAmfCtDefDisableRestart` is an attribute of the `SaAmfCompType` object class, and it is defined as: `saAmfCtDefDisableRestart`: `SaBoolT` [`0..1`] = 0 (`SA_FALSE`){`CONFIG`, `WRITABLE`, `SAUINT32T`}.

9.2 Rules Used for Representing some Relationships Among Classes

The following rules were used for representing some relationships among object classes.

1. The naming hierarchy (established by the DN names) is represented by a composition.
2. If the relation is an association between object classes x and y, and this relation does not reflect the naming hierarchy, in most cases, one of these classes has an attribute (usually a multi-valued attribute) referring to the other. An arrow is used to show the navigability. The arrow points to x if y contains an attribute referring to one or more objects of x (meaning that one can navigate from y to x). If neither of the two classes has an attribute referring to the other, an ***X*** is shown in both ends of the association to indicate that it is not possible to navigate from x to y or from y to x. No case exists for which both x and y have an attribute referring to each other.
3. If the relation is an association class z between classes x and y, z contains an attribute referring to either x or y. If this attribute refers to x, an arrow points from y to x to show the navigability from y to x.

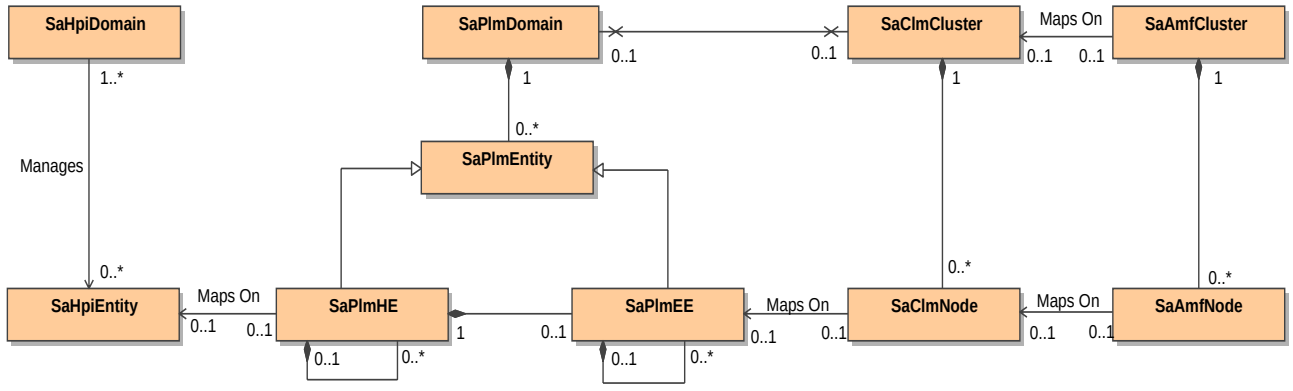
9.3 Some Global Views on AIS Classes

9.3.1 AIS Cluster View

The overview diagram FIGURE 7 shows the relationships among some object classes of HPI, the Platform Management Service, the Cluster Membership Service, and the Availability Management Framework classes.

Attributes and operations of the Availability Management Framework classes `SaAmfCluster` and `SaAmfNode` are shown in [4]. Attributes and operations of the Cluster Membership Service classes `SaClmCluster` and `SaClmNode` are shown in [5]. Attributes and operations of the Platform Management Service classes are shown in [19], attributes and operations of HPI classes are shown in [23].

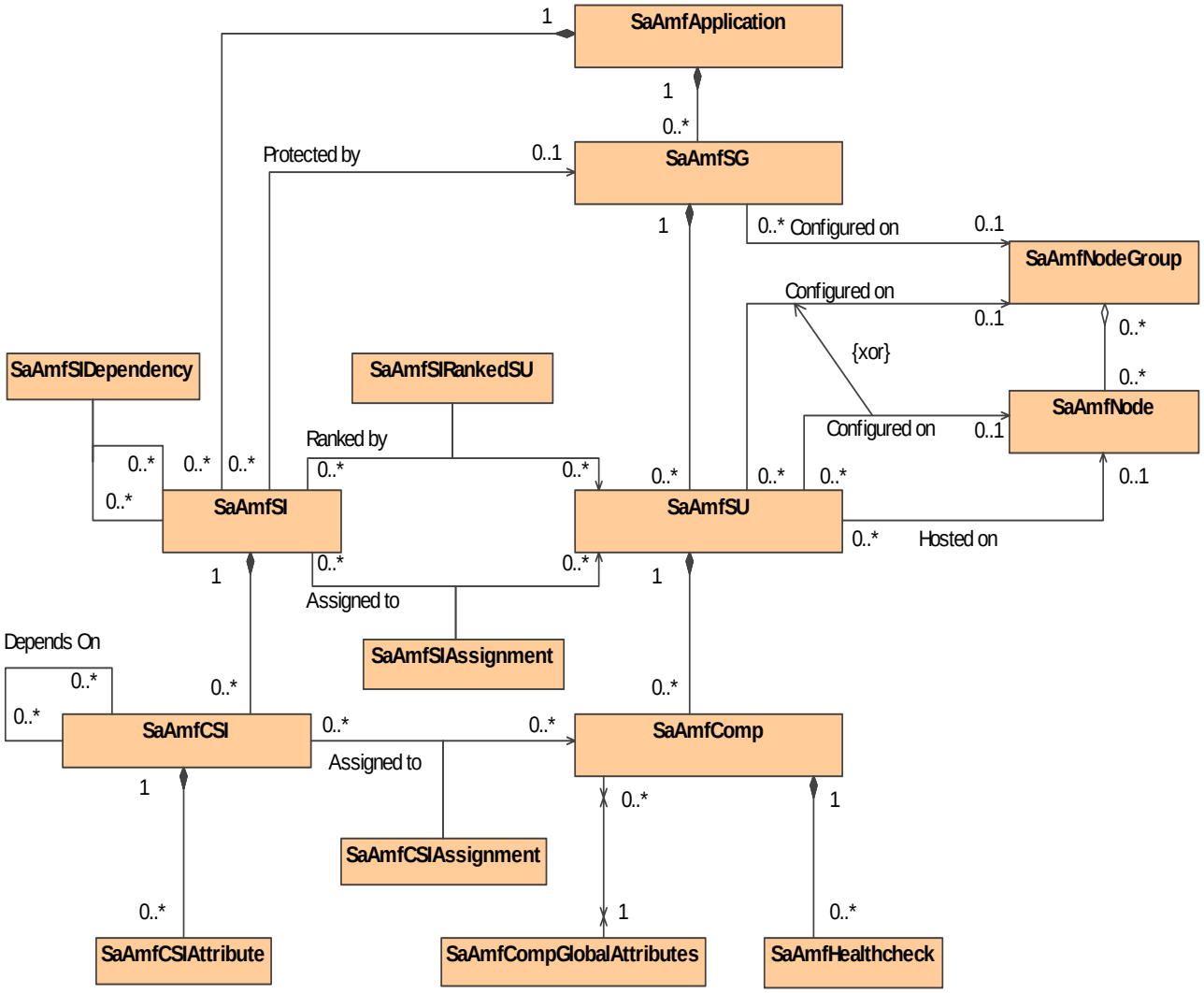
FIGURE 7 1- Cluster View



9.3.2 Availability Management Framework Instances View

The overview diagram FIGURE 8 shows in a simplified manner (by omitting all type object classes) how the various object classes of the Availability Management Framework relate to each other. Attributes and operations of all these classes are described in [4].

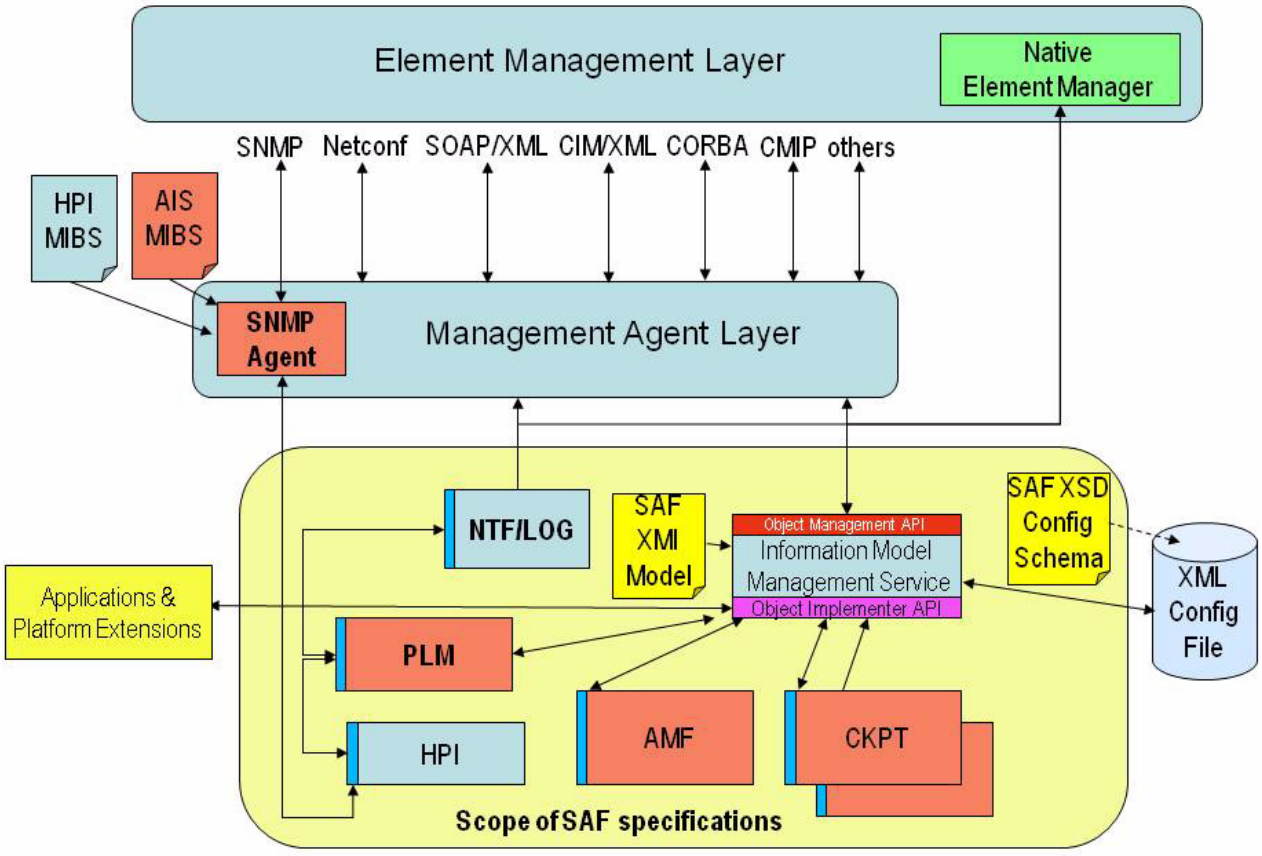
FIGURE 8 3.2- AMF Instances View



10 SA Forum Operation and Administration

Management applications can either access the AIS Management Services directly or interface with them through an appropriate management agent. In the latter case, the mapping of the SA Forum Information Model to the structure required by the application is out of scope for the SA Forum. The model provided in the XMI format and the configuration schema provided in the XSD format (see [12] and [13] respectively) can be used to define mappings. Information model extensions defined for applications and platform extensions can also be accommodated in the same way. A pictorial overview of the SA Forum management environment is provided in FIGURE 9.

FIGURE 9 SA Forum Management Environment



11 Glossary

2N — In a 2N redundancy model, every active element has its own dedicated standby.

active — An element that is currently in use and providing a service is said to be in the active role.

AIS — Application Interface Specification

AIS Service — A service provided by the Application Interface Specification. When the term AIS Services is used without a further qualifier, it refers to all the services and frameworks that comprise the Application Interface Specification.

AIS Framework — A framework provided by the Application Interface Specification.

AMF — Availability Management Framework part of AIS

area — The AIS is divided into a number of distinct services and frameworks. Each of these is referred to as a particular area in the CPROG document. The term area is used to refer to the functionality defined by the particular AIS Service or AIS Framework.

application — A set of services designed to fulfill specific needs of a user that are managed and deployed on a platform. Application software is distinct from operating systems or middleware which together with the hardware constitute a platform.

area server — Is the term used to refer to the logical elements in a system that service the function calls of the interface of a given area.

ASCII — American Standard Code for Information Interchange

availability — Refers to the proportion of time that a system is available to perform its function correctly and can be very generally defined as uptime divided by (uptime + downtime).

carrier grade — A set of characteristics required from equipment or software used for providing service in a public network. These characteristics include high degrees of availability, maintainability, manageability, reliability, responsiveness, and scalability.

CKPT — Checkpoint Service part of AIS

CLM — Cluster Membership Service part of AIS

cluster — A collection of computers or nodes which may be considered a unified computing resource, that is, an environment within which availability and scaling of services may be extended beyond that of which a single node is capable.

cluster member — Refers to a node in the cluster membership.

cluster membership — See membership.

component — Part of a system. In the context of the AIS it refers to the smallest software unit managed by the AIS frameworks. A component is typically instantiated as one or more processes executing on a single node.

COTS — Commercial Off the Shelf

CPROG — Title of the document that describes the programming model of the AIS specifications for the C programming language.

DN — Distinguished Name. It is used in object naming hierarchies to refer to a fully qualified name within the hierarchy.

DSP — Digital Signal Processor

EE — Execution Environment, a logical entity defined in PLM to model an hypervisor or operating system instance

EVT — Event Service part of AIS

fail-over — Is the process by which services are moved from a failed active element to an operational standby or spare.

failure unit — A failure unit or unit of failure in a system is a set of elements that are designed to fail as a unit. Failure units can be composed. For example, a node is a failure unit in a cluster, a process or a component is a failure unit in a node.

FRU — Field Replaceable Unit

HE — Hardware Element, a logical entity defined in PLM to model any kind of hardware entity

high availability — High probability that when a service is requested, it will be provided

HPI — Hardware Platform Interface

ICT — Information and Communication Technologies

ITU — International Telecommunication Union (www.itu.int)

IM — The SA Forum Information Model

IMM — Information Model Management Service part of AIS

ISV — Independent Software Vendor

LCK — Lock Service part of AIS

LOG — Log Service part of AIS

member node — Refers to a node in the cluster membership.

membership — Refers to the set of nodes that are part of a cluster at a given time. Membership can be of two types: potential and actual. Potential membership is the set of nodes that have been configured to be members of the cluster. Actual membership is defined by the set of nodes that, at a given time, are deemed capable of providing service and of communicating with the other members. When not qualified, the term refers to actual membership.

middleware — Refers to any software that is not provided as part of the operating system but is layered on top of one or more instances of it and provides APIs and services upon which higher level applications are based. Often applications are not written directly to the operating system, but are rather written to APIs provided by some portability and robustness enhancing middleware layer. This layer of abstraction is often the platform for application or system services.

MSG — Message Service part of AIS

N+1 — In an N+1 redundancy model, a single standby element is shared by N active elements. If one of the active elements fails, the single standby element will replace it.

N+M — With an N+1 redundancy model, the system can recover from the loss of one element, but if a second element fails before the first one can be repaired, a loss of service may result. In an N+M redundancy model, there are N active elements and M standby elements.

N-way — In an N-way redundancy model, each element can have both active and standby roles at the same time. It has the advantage that all elements can be used to provide active service while still providing standby protection. The AMF allows only one element to have the active role for a service at a time.

N-Way active — Redundancy model that does not require failover in terms of element functionality but rather the redirection of requests to available resources.

NAM — Naming Service part of AIS

node — Element of a system which itself is a complete computer system, containing hardware, operating system, and communications capabilities.

notification — Information emitted by a managed object relating to an event that has occurred within or concerning a managed object.

Overview — Title of the document that introduces the objectives of the SA Forum and its specifications, describes the architecture of the interfaces, and provides an overview of the functionality of the interface specifications.

NTF — Notification Service part of AIS

PLM — Platform Management Service part of AIS

RDN — Relative Distinguished Name. It is used in object naming hierarchies.

redundancy — Redundancy is a strategy which may be used to improve availability by coordinating multiple instances of an element such that when there is a failure in one, the others ensure that the services that were provided by the failed element continue to be provided by the redundant elements.

redundancy model — It refers to the arrangement between a set of elements to provide a given degree of redundancy. In general, the arrangement determines the ideal number of elements in the active role and the number of elements in the standby role for the same service. Some of the redundancy models supported by the AMF are: 2N, N+1, N+M, N-way, and N-Way active.

release code — A single capital ASCII letter designating the designating the release code version of an SAI document.

release number — A decimal number used to refer to the complete set of specification documents published together by the SA Forum.

reliability — The probability that a product (system) performs its intended function for a specified time period when operating under normal (or stated) environmental conditions.

SAI — Service Availability Interface

SEC — Security Service part of AIS

service availability — Refers to the property that one can initiate a service at any given moment.

service continuity — A synonym for service reliability

service reliability — Refers to the property that an initiated service request will be fulfilled.

SMF — Software Management Framework part of AIS

SMP — Symmetric Multi-Processor

standby — An element in the standby role is coordinated with one or more active counterparts and is ready to take on the active role of these counterparts if one fails or a switch-over is initiated. An element in the standby role is not expected to actively process and respond to service requests.

switch-over — Refers to the coordinated movement of services from the current active elements to other elements. The latter would typically be have the role of standby or spare prior to the switch-over.

TMR — Timer Service part of AIS

UML — Unified Modelling Language

UTF-8 — 8-bit Unicode Transformation Format. A variable-length character encoding for Unicode (unicode.org).

XMI — XML Metadata Interchange

XML — eXtensible Mark-up Language

XSD — XML Schema Definition