

Evaluation of the SAF AIS implementations

Mika Karlstedt
University of Helsinki

Kimmo Raatikainen
University of Helsinki

July 31, 2007

Abstract

The SA Forum was established 5 years ago to create standards for the high availability market. From the beginning, the focus of it has been on carrier-grade properties required by telecom networks. The first implementations were partly based on proprietary APIs since the specifications were not final yet. Nowadays implementations have emerged with more complete coverage of the specifications.

This document looks at a number of commercial and open source SAF AIS implementations to evaluate which parts of the SAF AIS specifications have been implemented and how faithful these implementations are. Some of the implementations provide some of the functionality using components that are not SAF AIS compliant while some of the functionality is not implemented at all. We were interested only in the direct support they have for the AIS. We did find frameworks that supported the AIS, but we were disappointed about the speed at which the implementers have started to support the specifications. Most frameworks support parts of the B.01 specification which is over two years old. The newer B.02 specification, which was released over a year ago, has no support yet. We would have expected that providing support for the B.02 would have been faster when there already is support for the B.01. All is not lost yet since the AIS have received more support lately, and we will expect the next few years to decide whether the AIS will be success or not.

1 Introduction

Most high availability systems can be divided into two components, the HA framework and the core application. The core application implements the real functionality of the system, while the HA framework provides highly available infra structure that allows the system to be highly available. There are also two ways to implement the HA systems. You can either build the whole system yourself or you can use third-party HA framework and build only the core application yourself. There are many HA frameworks you can choose from with different properties and capabilities. One drawback with these frameworks is that they are not standardized nor compatible with each other. It is not possible to start a project using one framework and later migrate to another, if

the original framework does not meet all the requirements that arise during the development.

Most HA frameworks have similar components with similar functionality. It is to be expected that they resemble each others since they all try to solve the same problem. Some differences in the frameworks are inevitable because the frameworks have different targets, like stateless vs. stateful server clusters. But even when both frameworks have the same target changing the framework requires great deal of extra effort. The Service Availability Forum aims to change that. It was founded in December 2001 to create open standards for high availability frameworks. The goal is to create a set of standard APIs similar to POSIX that would apply to HA frameworks. It should be possible to create a fully functional HA application using only the standard APIs. It would then be possible to migrate from one HA framework to another with minimal effort. In other words the HA frameworks should become COTS technology just like operating systems have already become.

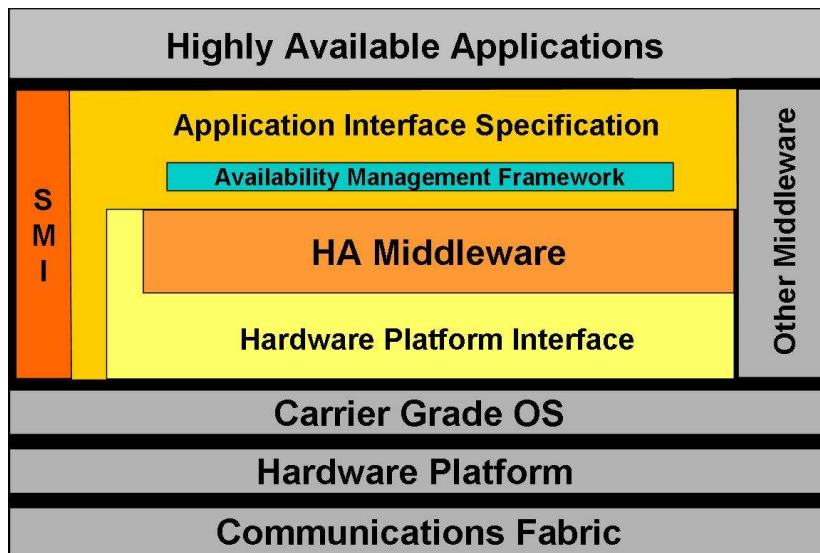


Figure 1: An Overview of the SAF model

The SA Forum specifications are divided into three parts as depicted in the picture 1 taken from their web site. The gray areas are outside the scope of the SA Forum leaving orange/yellow colored parts the focus of the SA Forum. The HA middleware is divided into three parts, System management Interface (SMI), Application Interface Specification (AIS) and Hardware Platform Interface (HPI). The Availability Management Framework (AMF) is part of the AIS and shown separately to emphasize it's importance.

Of the three parts System Management Interface has not yet been defined. Since the Forum has been around for over five years and there is still no version

of the SMI, it is unclear whether the SA Forum will ever manage to create the SMI specification. The other two parts are more successful, with numerous implementations. Especially the HPI has become a success story with most hardware vendors providing a version of it with their products. Because we wanted to evaluate how widely the SAF specifications are adopted we left the HPI out, since we were already convinced that the HPI is a success. We could then concentrate on the newer AIS specification.

The structure of this paper is following. First we introduce the SAF AIS specifications in the chapter 2, then we go through each of the implementations in the chapter 3. And finally the paper ends with a conclusions in the chapter 4, where we collect the data and see if we can get some ideas about the success of the specifications.

2 An introduction to SAF AIS

Specification	28.04.2003	19.11.2004	03.01.2006	27.02.2007
AMF	A.01	B.01	B.02	B.02
CKPT	A.01	B.01	B.02	B.03
CLM	A.01	B.01	B.02	B.03
EVT	A.01	B.01	B.02	B.03
IMM			A.01	A.02
LOCK	A.01	B.01	B.02	B.03
LOG			A.01	A.02
MSG	A.01	B.01	B.02	B.03
NAM				A.01
NTF			A.01	A.02
TIM				A.01

Table 1: AIS specification versions

The first version of AIS was released in the April 2003. It consisted of six different specifications. Later versions have added first three and then two new specifications. Meaning that the latest release has 11 specifications and at least one new specification is about to be released. All eleven specifications with their release dates are shown in the table 1. Minor releases with just editorial changes have been left out from the table.

Traditional way of creating a highly available systems is to use hardware redundancy and watchdog systems to monitor the hardware components. When the monitoring system detects a failure it orders a new component to take place of the failing component. Simply starting a new server is not always enough as all the processing done in the failing server will be lost. Redoing the work slows the response to the user which in real-time environments can be unacceptable. Similarly some operations cannot be repeated without undesired side-effects.

The situation changes if the application is stateful and we have a mechanism to store the state of the application somewhere. When the server fails we can retrieve the latest checkpoint data and continue processing from there greatly decreasing the slowdown.

Real world HA applications are inevitable distributed applications because there has to be a cluster of computers to ensure high availability even when a hardware failure makes one computer inoperable. The clusters require monitoring and configuration services so that they could provide the required services with required availability. The distributed nature of the applications on the other hand provide other difficulties for the application development. Therefore most HA frameworks provide also a set of services that make building HA applications easier to develop. The AIS specifications reflect this by defining a single monitoring and administration component (AMF) and a set of services targeted to make it easier to write HA applications. The set of HA services will likely increase in the future, though the next new specification (Software Upgrade Specification) is an administrative specification by nature.

The SA Forum specifications are often regarded as targeted towards telecom environments, but there is nothing in the specifications that is specific to telecom applications. The specifications are clearly targeted to stateful servers instead of the easier stateless servers. But there is for example no reference to real-time capabilities nor is there any real-time requirements. The specifications are also operation system agnostic, though there is a reference that the specifications are made with POSIX-like OS in mind. The documentation states that implementing the APIs in a non-POSIX-like operating systems is both possible and encouraged. It should be possible to adopt the specifications so that the implementation succeeds in a non-POSIX operating system.

From the previous, it should be obvious that the SA Forum is not Linux specific standards body, but all available SAF AIS compliant frameworks are implemented on top of Linux. Few frameworks provide implementation in other operating systems too, but they are exceptions.

As it is important to know the content of the AIS specifications before we can evaluate the frameworks that implement them, we have gathered a short description of all eleven specifications to the tables 2 and 3. The tables try to give some idea about the purpose of each specification. For better description we encourage any interested reader to check the specifications which are readily available from the SA Forum web site. Especially the overview document gives good, short and accurate description about all of the specifications. The table shows also the acronyms used to designate the specifications. We decided to refer to the specifications using the acronyms as the real names of the specifications are too cumbersome to use.

3 The implementations of the SAF specifications

Next we go through some of the different frameworks that have implemented parts of the SAF specifications. We evaluate only how much of the specifica-

Availability Management Framework (AMF)	AMF monitors the notifications and starts and stops applications and nodes as required, using the configuration data available. It also assigns workloads to different nodes. The requirement for the AMF to work, is that the nodes have some kind restarting mechanism that the AMF can trigger.
Checkpoint Service (CKPT)	CKPT maintains the checkpoints. The CKPT is designed to be stored in the main memory. Meaning that the checkpoints need to be copied to another nodes, so that the checkpoints would not be lost if the node holding the checkpoints fails. Every checkpoint has a retention time. If nobody references the checkpoint during that time, the checkpoint becomes obsolete and will be removed to make room for new checkpoints.
Cluster Membership Service (CLM)	CLM maintains the clusters membership data. It can only be used to check the current status of the cluster, as the real work of manipulating the cluster is the responsibility of the AMF.
Event Service (EVT)	EVT is based on the publish/subscribe-model. Subscribers are anonymous meaning that the publisher does not need to know anything about the subscribers and can publish events even when there is no subscriber. Basic concept is the channel which identifies the event type.
Information Model Management Service (IMM)	The IMM is based on the CIM (Common Information Model) specification from the DMTF (Distributed Management Task Force). It is designed to manipulate the attributes of the entities that make the cluster. There are two APIs, one for the management applications and one for the object implementers. This service can be used to query the status of various entities and to set the values for some attributes.
Lock Service (LCK)	LCK is a distributed lock service, which can be used to synchronize access to shared resources. There can be more than one instance of the component active at the same time and it is the task of the LCK to ensure the consistency of the locking data across the different instances.
Log Service (LOG)	LOG is a distributed cluster-wide logging service. The SAF divides log usage to two categories; logging usage data to the administrator and using the logs as a debug tool. The LOG is meant for the administrator usage, application developers can (and should use) the logging facility found from the OS. There are three default log streams; alarm, notification and system log streams. It is also possible to create other application specific log streams. Even the application specific streams are cluster-wide.
Message Service (MSG)	The MSG is based on the idea of message queues, it provides a store-and-forward type of messaging service. Any process can send messages to any queue, but only one process at a time can open the queue and receive messages from it. It is also possible to create message queue groups so that there can be more than one receiver at the same time. With message queue groups it is possible to choose between different delivery policies, and broadcast the same message to every queue or to only one queue in the group. It is also possible to set a retention time to the queue, in which case the queue will be deleted if it is not opened within the retention time. The messages are persistent until delivery, but the specification does not require that the messages survive a node crash or a cluster shutdown.

Table 2: The components of the SAF AIS

Naming Service (NAM)	NAM provides a simple naming service. It can be used to map any name to any data, the specification poses no requirements to either the names or the data to which it maps. It is expected that the names use UTF-8 encoding but it is not strictly required.
Notification Service (NTF)	NTF is very similar to Event Service and they both use the same publish/subscriber model. There are 5 different notification types Alarm, State change, Object Create/Delete, Attribute change and Security alarm of which the Alarm and Security alarm notification has to be logged. No other notification type is allowed. Some of the notifications are created by the hardware, some by the AMF and some by the applications, but anybody can subscribe to any of these and it is possible to use filters so that the process receives only the kind of notifications it really is interested in.
Timer Service (TIM)	It is very likely that a working HA application requires hundreds or thousands of timers. While it is possible to use OS for managing all these timers, the SA Forum expects that a specific component would improve the performance and reduce resource usage.

Table 3: The components of the SAF AIS continued

tions are implemented, not the quality of the implementation. We found few frameworks that have implemented only one of the AIS specifications and left them out of this evaluation. None of the frameworks implement all of the AIS. For some of the frameworks we found roadmaps that include at least some of the missing specifications. When available we try to include the roadmap data also, though it should be noted that we did not get access to roadmaps of every framework and the usage of roadmap information can skew the results. We do not think that to be a problem, because we are mainly trying to evaluate how seriously different frameworks take AIS. We expect that any framework that is serious about supporting AIS, will provide at least some amount of information whether they will include the missing specifications and when.

3.1 OpenAIS

OpenAIS is an open source project aiming to implement the AIS part of the SAF. Currently it has fully compliant implementations of the Cluster Membership Service, of the Checkpoint Service and of the Event Service. It has also beta versions of the Availability Management Framework and the Lock Service. It has also some code for the Messaging Service but it is still in alpha phase. We could not find any roadmap for the remaining specifications though they will most likely be included in time.

The OpenAIS is school book example of AIS implementation. It is quite evident that the project aims to implement AIS specification just as the SAF has designed them. Every specification is completely separated from others and provides a nice library against which the applications can be compiled. They

have also implemented a totem-protocol for totally ordered message delivery that can be used by applications if they need it. The totem implementation is so good that at least OpenClovis uses it also and Linux-HA has made some effort to ensure that totem integrates nicely to it.

3.2 OpenClovis

OpenClovis is another open source implementation of the SAF specification. It consists of many separate components some of which are SAF compliant. The framework documentation includes a document that specifies how compliant the framework is. Basically there is some kind of implementation for eight of the nine specification in the B.01 release, only the Lock Service is missing.

The components are divided into two categories based on whether they have AIS compliant API or not. There are AIS compliant APIs for the Availability Management Framework, Cluster Membership, Checkpoint and Event Services. The AMF of the version 2.2 does not support health checks and supports only the 2N and No redundancy models. The Event Service does not support event retention other than that the APIs are compliant. Like so many other frameworks, the OpenClovis is implemented against the openHPI implementation and can thus provide also platform management functionality, though there are no specifications in the AIS concerning that.

Most other frameworks have components that provide AIS functionality without having an AIS compliant API, but the OpenClovis is the only framework with a specific document for the purpose. In reality the AIS compliant components are implemented using simple wrappers that hide the fact that the functionality is implemented using a proprietary API of the OpenClovis. This fact shows in semantic differences between the implementation and the AIS specification. There are cases where for example the callbacks are invoked from the thread belonging to the OpenClovis context where they should be invoked in the application context. Another peculiarity of the framework is that it includes OpenAIS, but uses only the totem implementation of it. The rest of the implementation is probably included for license reasons.

The table 4 shows all the components that have something to do with any AIS specification. The second column specifies which AIS specification the component provides, if the API is not AIS compliant.

The structure of the OpenClovis is depicted in the picture 2. As can be seen from the picture the OpenClovis is built in a layered model with 4 layers. The lowest layer is Adaptation layer which is used to make OpenClovis OS agnostic. Next is the Communication Core which includes components that provide the communication services to upper layers. The third layer is Infrastructure core which provides the infrastructure services for upper layer. The highest layer is made of high availability sublayer and System management sublayer which are jointly located on the highest level, but which are logically two separate layers.

We were unable to find any information whether the OpenClovis plans to support the missing components or whether they plan to provide AIS compliant APIs for the components that still miss them.

Component	AIS specification	AIS compliant API
Inter Object Communication and Remote Method Dispatch	Messaging Service	No
Clovis Object Repository Service (COR)	Information Mode Management Service	No
OpenClovis Alarm Service	Notification Service	No
OpenClovis Log Service	Log Service	No
Availability Management Framework		Yes
CheckPoint Service		Yes
Cluster Membership Service		Yes
Event Service		Yes

Table 4: AIS components of the OpenClovis

3.3 GoAhead SelfReliant

The SelfReliant from GoAhead will support most of the AIS specifications with the release 5.0. There is an Early Access Release for the 5.0 version which at the moment (EAR2) supports directly Cluster Membership, Checkpoint, Event and Messaging Service. The Final version will support also Availability Management Framework and Log Service, and the version 5.1 includes the support for Lock Service. The structure of the SelfReliant is shown in picture 3. The picture depicts a future version of the SelfReliant as the present version does not have support for all of those AIS components, but evidently some 5-series version will have. The SelfReliant has also good support for all redundancy models specified in the AIS. The current release 4.2 does not support any of the AIS APIs, but does of course provide similar functionality to most of the AIS specifications. The 4.2 release comes with a two versions SR-BAM (SelfReliant Basic Availability Management) and SR-AS (SelfReliant Advanced Suite), where the SR-AS is as the name implies more advanced. Some features are available only in the SR-AS, most of the redundancy models belong to these features.

The GoAhead has also announced the release of the SAFfire 2.0 which will include Self-Reliant 5.0 and adds support for three other services specified in the B.02 release. It is expected to be available in Q3 of 2007.

3.4 Motorola NetPlane or OpenSAF

OpenSAF is the newest addition to the AIS frameworks being released in June 2007. It is based on the NetPlane framework from Motorola which they open sourced at that time. It is still too early to tell what the future of the OpenSAF will look like, but we do expect bright future for it, as most NEPs (Network Equipment Providers) have already expressed support for the project.

The picture 4 shows the NetPlane architecture. It is apparent from the picture that NetPlane supports directly AMF, CLM, CKPT, EVT, LCK and

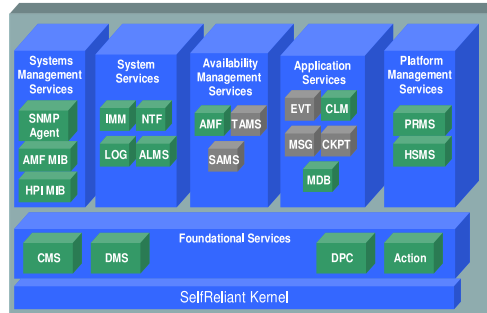
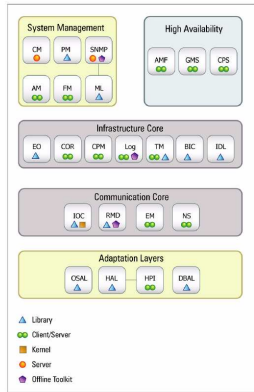


Figure 3: SelfReliant

Figure 2: OpenClovis

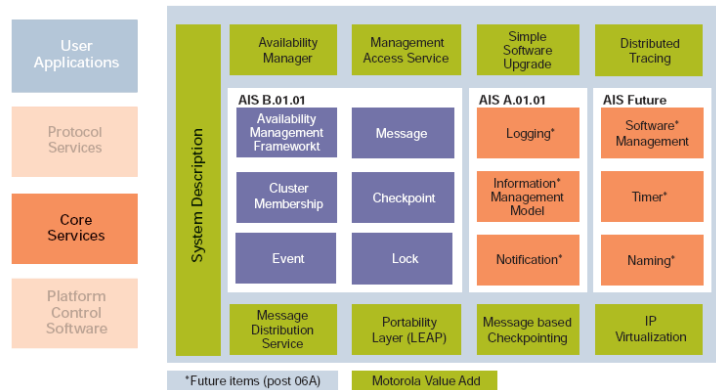


Figure 4: The structure of the NetPlane architecture

MSG. There is no support for IMM, LOG or NTF at the moment, but Motorola has stated that they want to include support for them too. They also plan to support other future AIS specifications. All that changed when they open sourced their product. The future of the OpenSAF will be decided once the OpenSAF foundation has been created.

3.5 Fujitsu-Siemens SAFE4CS

SAFE4CS is an implementation by the Fujitsu-Siemens that covers most of the SAF AIS. It supports AMF, CKPT, CLM, EVT, LOCK and MSG, in other words the B.01 release specified in the November 2003. It is also one of the oldest frameworks to provide reasonable support for the AIS.

The SAFE4CS consists of two major components, RTP4CS and SAFE4CS. RTP4CS (Resilient Telco Platform for Continuous Services) is the underlying clustering solution built on top of PRIMECLUSTER. It provides a high availability clustering solution that provides SAF AIS compliant Cluster Membership Service and Lock Service. SAFE4CS provides the other 4 components of the SAF AIS (AMF, CKPT, EVT, MSG). RTP4CS provides functionality similar to both LOG and NTF, but the APIs provided nor the semantics are not compatible with the SAF AIS counterparts. Unfortunately the future of the framework is unclear and we could find no roadmaps for the future.

3.6 ESO Technologies HAPx

The ESO Technologies is in the middle of developing the implementation called HAPx. Present version 1.2.0 includes Logging, Cluster Membership and Checkpoint Services. There is also limited implementation of the Availability Management Framework. The next release which is to be released in mid-2007 will include support for more services. The release at the end of 2007 should include support for all specifications of the B.01 release. The HAPx is the first framework to support Logging Service.

3.7 Other frameworks

We did evaluate some other frameworks also, but decided to leave them out of this evaluation as they did not have enough AIS APIs implemented. We collected the frameworks in this chapter so that we could specify more accurate reasons why these frameworks were left out of this study.

One of the most famous and at the same time oldest HA framework in Linux is Linux-HA. The framework has supported CKPT since 2004, but does not support any other APIs. We found some notes that at some time in the past there were plans to provide better support for AIS, but it seems that those plans have been forgotten.

Netra HAS is a HA suite from Sun. It is clustered solution with supports SAF Cluster Membership Service specification. Unfortunately that is all it does. There is no AMF nor any other SAF components available. And since SAF CLM API can be used only to read the status of the cluster not to modify it, the implementation is basically useless from the SAF point of view.

Element is a framework made by ENEA. At the moment the framework does not support any AIS specifications, but the company has plans to support the AIS in the future. They already have some kind of beta version available but

since we did not get more information about their plans, we could not include the Element in this study.

Trillium is a framework from the Continuous Computing, and while it definitely is aimed for the same market as the AIS, there is no reference in any of their documentation that they will ever support AIS.

4 Conclusions

We have collected the data in the table 5. We left the two newest specifications out of the table because we collected some of the data about the frameworks before the specifications were released in February 2007. The Y stands for Yes and N stands for No. The F means that we were able to find roadmaps or other plans to include the feature in the future.

Implementation	AMF	CKPT	CLM	EVT	LOCK	MSG	IMM	LOG	NTF
HAPx	F	F	F	F	F	F	F	F	F
OpenAIS	F	Y	Y	Y	F	F	N	N	N
OpenClovis	Y	Y	Y	Y	N	N	N	N	N
OpenSAF	Y	Y	Y	Y	Y	Y	F	F	F
Safe4Try	Y	Y	Y	Y	Y	Y	N	N	N
SelfReliant	Y	Y	Y	Y	F	Y	F	F	F

Table 5: Coverage of the AIS implementations

We have ordered the specifications in time order. The first six were released already in the first release while the last three were added later. From the table we can see that five of the first six specifications have a good support, the LCK being the exception in the first category of the AIS specifications. Similarly none of the three newest specifications have an implementation yet even though all of them are vital part of any successful telecom environment and indeed similar functionality is found from all implementations.

Implementation	2N	N+M	N-way	N-way Active	No redundancy
OpenClovis	Y	N	N	N	Y
OpenSAF	Y	Y	Y	Y	Y
Safe4CS	Y	N	N	N	Y
SelfReliant	Y	Y	Y	Y	Y

Table 6: Redundancy models

The AMF specifies five different redundancy models and we studied also how many of these are supported by the frameworks. The redundancy models are important because they specify how the workloads will be distributed to the nodes in the cluster. 2N-redundancy model is familiar to anyone familiar with HA systems. There is a primary and a backup server. The primary serves the

client and backup just keeps a backup of all the important data. If the primary fails for any reason, then the backup becomes primary and starts to serve the client. The problem with that model is that half of the servers are actively doing something and half are basically idle. In practice it is very unlikely that there would be many nodes down at the same time. It would therefore make sense to use more than half of the nodes serving clients and leave the rest as backups. This is exactly what the N+M redundancy model is all about. There is N active nodes serving clients and M backup nodes. If the cluster consists of 16 nodes, we have 8 primary and 8 backup nodes if we use 2N-redundancy model. With N+M redundancy model we could divide the nodes to 12 active and 4 backup nodes. Of course the backup nodes in the two models will behave differently and it is not always possible to use N+M redundancy model. The last two models N-way and N-way active are modifications of the N+M model and it is not possible to describe them without explaining the object model used in the AMF, which we do not have space in this paper. Fortunately they are not as important than the three other redundancy models.

The results are gathered in the table 6 but only from the implementations that support AMF. It is obvious from the table that 2N-redundancy is well supported as is No redundancy but the others are less widely supported. Unfortunately N+M is perhaps the most important of the redundancy models, so there is still work in this area.

The final conclusion of this evaluation is that there is good support for the AIS. The support is not quite so good as one would expect, as most frameworks support only B.01 specifications, which is over two years old. We would have expected that some of the frameworks would have already implemented B.02 specifications. We can understand that the three new A.01 specifications are not yet supported, but we expected to find more roadmaps to include these specifications. In the end the deciding factor is not the frameworks alone. The applications are as important as the frameworks. Without applications that require or use the APIs there is no need to support them. It is unfortunately very difficult to find out whether there are any applications built on top of the AIS APIs, but we are confident that AIS will have an impact on the HA frameworks, at least in the embedded (like telecom) environments.

References

- [SAF06] Service Availability Forum, <http://www.saforum.org/specification/>
- [oClovis] OpenClovis ASP, Alignment with the Service Availability Forum Standards V1.3- 1 Aug 2006