4th International Digital Curation Conference
December 2008

The eXtensible Access Method (XAM) Standard

Steve Todd,
EMC Corporation

November 2008

Abstract

Recent developments in the storage industry have resulted in the creation of an industry standard application programmer's interface (API) known as XAM, the eXtensible Access Method. The XAM API focuses on the creation and management of reference information (otherwise known as fixed content). Storage vendors supporting the XAM API will provide new benefits to applications that are creating and managing large amounts of fixed content. The benefits described by this paper merit consideration and research by developers creating applications for Digital Curators.

The 4th International Digital Curation Conference taking place in Edinburgh, Scotland over 1-3 December 2008 will address the theme *Radical Sharing: Transforming Science?*

Introducing the XAM Initiative

In 2004 a small group of mass storage device vendors gathered to discuss the growing importance of storing "fixed content". Fixed content is information that once written does not change (or is changed infrequently). This small group grew to include two more types of interested parties: application vendors and end users.

A very specific set of needs were generated:

- The needs of application vendors to annotate fixed content, indicate storage policies for fixed content, and manage enormous amounts of individual pieces of fixed content.
- The needs of end users to select from a variety of fixed content applications and storage vendors, perform fixed content migration between different application and storage vendors, and comply with local, national, and international regulations.
- The needs of storage vendors to comply with fixed content regulations, efficiently store large amounts fixed content (and associated metadata), and support the large range of applications that generate fixed content.

In December of 2005 the Storage Networking Industry Association (SNIA) created the XAM Initiative to address these needs. The members of SNIA are "dedicated to developing and promoting standards, technologies, and educational services to empower organizations in the management of information" (SNIA, 2008). The XAM initiative brought together a variety of application vendors, end users, members of academia, and storage vendors. As a result of this activity it has recently been announced (XAM Press Release, 2008) that a specification for a *standard* fixed content API is close to approval (XAM specification 1.0), and conformant software developer kits are nearing release. The XAM specification will also be submitted for consideration as an ANSI/ISO standard.

XAM is based upon the time-tested and proven methods of commercial solutions that are already available in the marketplace. These solutions are migrating from their own proprietary APIs towards the XAM industry standard.

XAM is also designed with long-term retention in mind. SNIA's 100 Year Archive Task Force believes that both the OAIS and XAM standards can assist with the "grand technical challenges" of long term digital information retention: logical and physical migration. (Year100, 2007).

The availability of a software development kit (SDK) for XAM means that application developers can write new applications specifically designed for fixed content. Digital Curation software falls into this category; this paper is an introduction and a call to research.

One of the goals of the XAM API is to become as ubiquitous (Todd, 2006) to application developers as the file system API.

XAM History and Motivation

In 2002 EMC Corporation introduced a storage system specifically designed for the storage and retrieval of fixed content. The Centera storage system differs from traditional block and file storage systems in several ways.

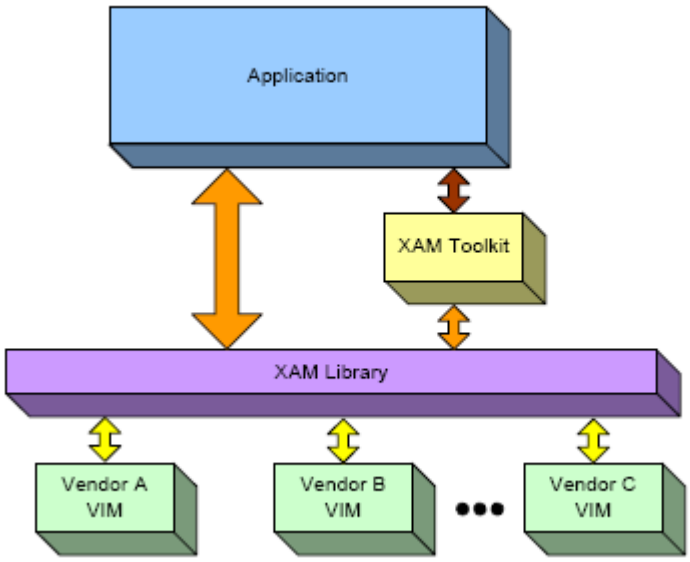
- An object interface is presented to applications as opposed to a block storage or file storage interface.
- Content cannot be stored to the system without accompanying metadata.
- Overwriting content is not permitted.
- All content and associated meta-data are both protected and retrieved using hash values known as content addresses. Successful retrieval of content implies that the content and meta-data are original and authentic.
- The system presents a flat address space, which obviates the need for configuring front end file systems or logical units (LUNs). Adding new capacity to store additional objects also does not require manual configuration; new disks are automatically assimilated into the general object pool.

In 2002 there were no widely supported standards for communicating with object addressable storage systems. Block storage systems, on the other hand, benefited from the SCSI protocol, while file storage systems benefited from the CIFS and NFS protocols. This lack of standards required the creation of a proprietary application programmer's interface and a software library known as the Centera SDK. Many application vendors (CenteraPartners, 2008) integrated the Centera API into their software offerings. These software offerings included fixed content products such as medical imaging and email archiving.

EMC reasoned that Centera would benefit from the creation of an object-based standard. In 2004 EMC joined forces with IBM and began working on a specification for a programmer's interface for fixed content storage devices. In 2005 several other vendors joined the effort, including Hewlett-Packard, Hitachi, and SUN. A joint proposal was made (XAM Version 1.1) to certain software vendors. In late 2005, XAM Version 1.2 was proposed to SNIA. The Fixed Content Aware Storage Technical Working Group (FCASTWG) was formed with the goal of creating and ratifying a formal XAM interface (XAMGenesis, 2007). In October of 2007 the first multi-vendor demonstration of XAM technology took place (XAM Plugfest, 2007), and in the summer of 2008 the first versions of shipping code became available (XAM v1.0, 2008).

The first version of the XAM API provides the basic methods for storing, retrieving, and searching fixed content. It also provides support for storage system policies, such as the retention of fixed content, and the electronic shredding of content. More specific details about XAM API implementations and functionality are described in subsequent sections.

In addition to providing API documentation for both the C and Java programming languages, SNIA has provided an architecture document which describes the software architecture for XAM. The following diagram has been extracted from the architecture document and inserted here in order to provide more detail about the XAM software architecture.



Applications are developed using the standard XAM function calls provided by the XAM library. XAM toolkits can also be built on top of this library. Underneath the XAM library are vendor implementation modules (VIMs). The purpose of a VIM is to translate the standard XAM API calls into the protocol of the underlying storage system. XAM, therefore, is not a protocol to an object-based storage system, but a function call interface that can be mapped onto storage systems provided by a variety of vendors.

Applications that program to the XAM API can realize the following benefits:

- They no longer need to manage the association of disparate files and/or database entries. XAM acts as a "paper clip" that automatically joins together multiple pieces of content and meta-data in an inseparable fashion.
- They can specify all content as "binding" (defined below) and guarantee that the content cannot be modified, and ensure upon retrieval that the content is original and authentic.
- They can specify content as "non-deletable" via XAM's retention period feature.
- They can migrate content between different (or the same) vendor storage systems. This becomes especially important for events such as technology refresh or lease rollover.
- They no longer have to target a specific database or file system when storing content.

Vendor storage systems that support the XAM API are known in XAM as XSystems. The API for communicating with XSystems is described below.

The XAM API

The main functional goal of the XAM API is to provide applications with the ability to unite fixed content with relevant metadata, associate this union with a fixed

identifier, and then store all of the content on a mass storage device. The most important abstraction that XAM provides is known as an XSet. An XSet logically "contains" the union of fixed content and metadata. The identifier associated with an XSet is known as a XUID. The mass storage devices that support XSets and XUIDs are known as XSystems.

XSets

An XSet represents the actual fixed content being managed on an XSystem (defined below). An XSet can contain zero or more pieces of data (e.g. a scanned image), zero or more pieces of metadata (e.g. information about a scanned image), and a variable number of fields (e.g. name/value pairs).

The ability to store content and metadata in one logical XSet construct raises the potential for using the XAM standard as an implementation choice for an OAIS Archival Information Package (OAIS, 2002). An XSet can store the content (Data Object), and multiple pieces of metadata describing that object (e.g. Representation Information and Preservation Description Information). The AIP can be represented and retrieved as one solitary unit, as opposed to, for example, a collection of files stored in a common directory. See the "XAM Example" section for more detail on implementing an AIP.

XSets also allow for the application of policies on the content contained therein. For example, the XAM specification currently supports the ability to specify retention policies on XSETs (e.g. disallow delete for 3 years). Policy support is dependent on the capabilities of the underlying XSystem. Any policy specific to digital curation can be proposed to the SNIA as a new standard. Potential new XAM digital preservation policies include quality control, disaster recovery, and security (ERPA, 2003).

Properties, XStreams, and Fields

An XSet property is a name/value pair. Properties contain simple values such as strings or integers. An application can add zero or more properties to an XSet.

An XStream is an arbitrarily large binary data stream such as a JPEG file, XML file, or other data set. An application can add zero or more XStreams to an XSet.

A XAM field is the name given to a property or an XStream. XAM allows applications to iterate over fields in an XSet and get/set values.

XSets are populated with properties and/or XStreams, and then "committed" to an XSystem.

XSystems

An XSystem is defined as a mass storage device that supports the XAM standard. XSystems represent an opportunity to research a new data storage access method. Current data storage research, for example, includes the CASTOR effort occuring at CERN (CASTOR, 2008), the High Performance Storage System effort (HPSS, 2008), and the OceanStore project (OCEAN, 2008). The mass storage interfaces discussed in these efforts include both file system and object-based interfaces; the XAM API is purely an object-based interface to a mass storage device.

XUIDs

A XUID is the "claim check" that allows an application to retrieve an XSet from an XSystem. When an XSet is stored to an XSystem, the XSystem generates a unique XUID representing that XSet and returns it to the application. The XUID is an opaque

string of characters.

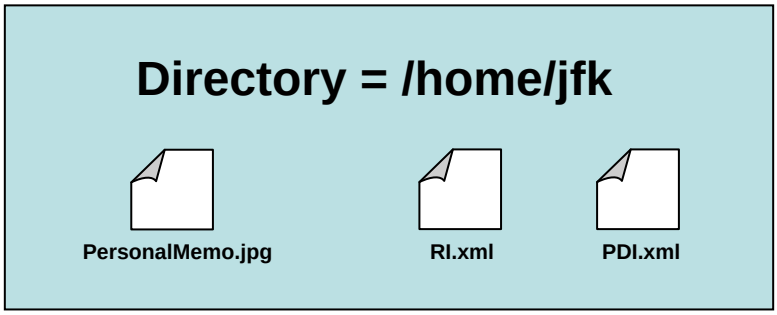
XUID creation can be influenced through XAM's concept of "binding". Any element placed as a field within an XSet (e.g. an XStream or a property) can be labeled as binding. Labeling fields of an XSet as binding results in a hash of those fields. The results of the hash are represented within the XUID. Identical XSets stored to an XSystem may result in the exact same XUID, which offers the potential for solving the problem of structured collisions (CORDRA, 2008) across repositories.

A XUID is essentially a new form of a Persistent Identifier (PI, 2008). A XUID has characteristics that differentiate it from traditional PIs:

- It is created by the XAM storage system and not the application.
- It cannot be "renamed".
- It cannot be "moved" by giving it a different "pathname". XUIDs are location independent and do not have absolute pathname contexts.
- The form of a XUID is in part governed by the content within the XSet represented by the XUID. Modification of said XSet can either leave the XUID unchanged or result in a new XUID; this choice is up to the application. New XUID creation does not automatically remove the old XUID; the original XSet remains intact upon the XSystem.

A XAM Example

It has been posited that the XAM API would be a useful way to represent an OAIS archival information package. Consider the following depiction of a scanned document from the archives of United States President John F. Kennedy. Assume that the scanned document has resulted in a file named "PersonalMemo.jpg" and will become the "Data Object" within the AIP. The digital curator has created two additional XML files that are associated the scanned image. These files represent AIP Representation Information (how to interpret the Data Object) and AIP Preservation Description Information (meta-data about the Data Object). These files represent a submission into an archive. Clearly this submission is lacking in completeness and detail; the reader is encouraged to focus instead on how the elements of an AIP *could* be stored as an XSet and "remembered" via a XUID.



The following code fragment describes how these three pieces of information are added into an XSet and stored onto an XSystem. For the sake of brevity this example lacks completeness and ignores error handling.

```
XSystem sxSystem;
XSet sXSet;
XStream DO;
XStream RI;
XStream PDI;
XUID    myXUID;
..

// create an XSET

sXSet = sxSystem.createXSet(XSet.MODE_UNRESTRICTED);

// create XStream objects within the new XSET

DO = sXSet.createXStream("DO",true, "image/jpeg");
RI = sXSet.createXStream("RI",true, "text/xml");
PDI = sXSet.createXStream("PDI",true, "text/xml");

// call a utility routine that loads file content into each XStream

Load_file_into_XStream(DO, "/home/jfk/PersonalMemo.jpg");
Load_file_into_XStream(RI, "/home/jfk/RI.xml");
Load_file_into_XStream(PDI, "/home/jfk/PDI.xml");

// commit the XSET into the XSystem

myXUID = sXSet.commit();
..
```

The final line of this code fragment represents the "ingest" process of the OAIS functional model. A set of disjoint files are being submitted into a repository as one unified archival information package (an XSet). The XSet is submitted to the archive (via the commit() method), and a unique, persistent identifier is being returned (myXUID). This XUID, in the case of EMC Centera, is a cryptographic hash of all the content within the XSet, and therefore unique (within the limits of cryptographic hash algorithms such as SHA). Note that the use of a cryptographic hash may be implemented by some vendors but not by all. The XUID becomes the "handle" or "persistent identifier" by which applications retrieve the AIP.

By default the data within this AIP is immutable; any changes to the content therein will result in a new XUID, and the old one is preserved. Additionally, a successful "read" of this XUID can automatically check the authenticity and originality of the AIP (e.g. Centera VIM verifies cryptographic hash during retrieval).

The use of XUIDs is a unique opportunity for digital curators. It is an opaque string of alphanumerics that is neither a filename nor a database entry. It serves as a "paper clip" that can glue together the disparate pieces of content that often make up an AIP.

This example does not describe any of the policies that can be assigned to XSets, such as the ability of an application to specify a retention period (e.g. keep this content forever).

XAM Implementation Use Case

The first version of the XAM standard has been approved, and working code is available. SNIA is currently working on the release of a "reference VIM". A reference VIM would allow for the development of XAM applications without the need for direct communication with a particular vendor's XSystem.

In order to highlight the fact that the XAM standard is "shipping code", the following table describes an XSystem manufactured by EMC (Centera), and comments

on the functionality provided by EMC's currently shipping XAM SDK.

SNIA XAM Feature	EMC's VIM and Centera
XSystem Support	✓
Create, Read, Write, Delete XSets	✓
Retention	✓
Electronic Shredding	✓
Checksum validation of binding fields and XStreams.	✓
Support for non-binding fields and XStreams	✓
Metadata addition/extraction	✓
Level 1 Meta-data Query	✓
Level 2 XStream Query	Not currently supported in EMC VIM 1.0.

EMC's Centera product possesses a variety of features that are not addressed by the first version of SNIA's XAM standard. These features are as follows:

Replication

The Centera system supports the replication of XSets to a remote Centera system.

XSet-based Protection

The Centera system protects XSets using either a CPM approach (Content Protection Mirroring) or a CPP approach (6+1 Content Protection Parity).

Background Validation

The Centera system supports the automated validation of XSets. Background processes running inside of Centera systematically scan and open XSETs and compare the archived data to the hash value embedded within the XUID. Corrupt XSETs are automatically repaired using the CPM or CPP redundancy.

Scalability

While the XAM specification places no limits on the number of XSets that an XSystem can hold, Centera is currently scalable up to 25 million objects per disk (CenteraObjects, 2008), with the smallest Centera system containing sixteen disks.

Audits of Deleted Content

The Centera system supports audited reports of deleted XSets.

XAM and Digital Curation Research

Given this brief description of XAM, it is the assertion of this paper that XAM contains numerous value propositions that are directly applicable to the challenges of digital curation. Potential areas of research are enumerated below.

The Unity of Content and Metadata

When a piece of content is placed into an XSET, and its associated metadata is added to it, and the entire XSET is subsequently stored onto an XSystem, the metadata and content become inseparable. It is not necessary to place metadata into a separate database or file; with XAM these items always travel together. This may reduce the burden on applications to manage the storage and retrieval of metadata as an entity separate from content.

A Unique, Persistent, Tamper-proof Identifier

The unique naming and persistent storage of XUIDs by an XSystem is mandated by XAM. The burden of creating (and remembering) unique identifiers has shifted to the storage system. By specifying the "binding" form of XUID creation, applications can guarantee the authenticity of the content being retrieved via the XAM API.

Simplified Capacity Upgrades

Adding capacity to a digital archive is an experience which can require extensive training in storage technologies. There is nothing within the XAM standard which makes capacity upgrade of an XSystem inherently easier. However, XSystems present a flat address space for storing content (as opposed to presenting a file system). This provides a storage vendor with an opportunity to *hide the configuration details of file systems, databases, and raw storage*. Properly designed XSystems can consume raw storage "on-the-fly" without extensive end-user involvement.

File System and Database Management

Digital archives that present file system and/or database interfaces require ongoing maintenance of those interfaces. Administrators must learn the details of the management tasks for those interfaces. Current solutions often involve different types of databases (Oracle, SQL) and different types of file systems (NFS, CIFS). Each interface requires learning a different set of management tools.

The XAM standard offers the possibility of eliminating the need for a database by using the XAM API to query metadata and fixed content. The location-independent nature of XAM does not require an externally visible file system.

Storage Vendor Neutrality and Migration

The selection of a storage vendor's product as the centerpiece of a digital archive is a critical decision. The promise of the XAM API is that XSETs are portable between storage systems. This provides a "second source" for the storage vendor selection (as opposed to "vendor lock-in").

There are a number of different reasons for replacing a storage vendor's product, including cost, power utilization, performance, quality, and technology refresh (e.g. a migration to the same vendor with larger capacity configurations). The XAM specification guarantees the portability of XSets between XSystems, which also includes continual use of the exact same XUIDs.

SNIA has fully addressed this issue through the inclusion of XSet import and export function calls. The XSet.Export() routine retrieves an XSet from a vendor's XSystem and stores it in a canonical form; XSet.Import() creates an XSet on a different XSystem using this canonical form. The XUID is preserved throughout this operation.

Flexible XSet Creation

XSets can contain multiple numbers and combinations of content, metadata, and fields. XSets can also be layered and embedded XUIDs can point to other XSets. This allows the creation of XSets that adhere to system models like the Open Archival Information Systems (OAIS) Reference Model.

One of the foundational elements of the OAIS model is the Archival Information Packet (AIP). The AIP is made up of many different "sub-elements", including package descriptions, packaging information, content information, and preservation description information (OAIS, 2002). The flexible XSet model allows for all of this information to be inserted into one XSet, or each sub-element can be stored as its own XUID and referenced by a parent AIP XSet.

Regardless of the implementation, an archival information package can be represented with one globally-unique persistent identifier. XAM binding techniques can also be used to conclusively prove that the AIP is authentic.

TRAC Assessment

An XSystem, as defined by SNIA, would appear to be a trustworthy repository. In order to validate this assertion it would be worthwhile to validate shipping XSystems (e.g. Centera) by using the Trustworthy Repository Audit & Certification checklist (TRAC, 2008).

Xsystem Administration and Policies

The XAM Standard has not addressed the standardization of administrative functions and system policies for XSystems. The requirements for standard management tools for digital archives should be communicated to SNIA for consideration in future versions of XAM.

Conclusions

The XAM API contains many features that are attractive to developers of digital curation software. The XAM specification itself is the attraction; implementations by mass storage vendors are just beginning. Knowledge of the XAM API at this stage of its lifecycle offers the opportunity for digital curation researchers to begin visualizing and building tools and applications that utilize the standard.

The overlap of XAM and current digital curation research is large:

- Persistent identifiers
- The unity of metadata and content
- Structured collision resolution
- Applying policy to content
- Mass storage accessibility

Becoming proficient with the XAM API is a matter of weeks (if not days) for experienced programmers familiar with the C and/or Java programming languages. Engagement with the SNIA XAM community can be accomplished via the www.snia.org/forums/xam website.

Finally, it is worth mentioning that the XAM standard will be evolving, and input

from users (such as Digital Curators) is valued and welcomed by SNIA.

Acknowledgements

Many thanks to Gil Press, David Black, Mike Horgan, Graham Stuart and Wayne Adams (all of EMC), for their assistance with this paper.

The full list of organizations currently involved in the XAM Initiative can be found on the SNIA website: http://www.snia.org/member_com/member_directory/.

References

[Internet journal]Year100. (2007, January). 100 Year Archive Requirements Survey. Retrieved July 2, 2008, from http://www.snia.org/forums/dmf/programs/ltacsi/100_year/.

[Internet journal]Castor. (2008, July). CERN Advanced Storage Manager. Retrieved July 1, 2008, from http://castor.web.cern.ch/castor/.

[Internet journal]CenteraObjects. (2008, March). EMC Expands Centera. Retrieved November 10, 2008, from http://www.emc.com/about/news/press/2008/20080313-01.htm.

[Internet journal]CenteraPartners. (2008, October). Application integrated with Centera. Retrieved October 7, 2008, from http://www.emc.com/partners/velocity/isv/isv-cas-specialty/centera-proven-program-for-isvs.htm.

[Internet journal]CORDRA. (2008, July). An Introduction to CORDRA (Content Object Discovery and Registration/Resolution Architecture). Retrieved July 1, 2008, from http://cordra.net/introduction/.

[report]ERPA. (2003). *ERPA Guidance Digital Policy Preservation Tool*. Electronic Research Preservation and Access Network, September, 2003. Retrieved July 1, 2008, from http://www.erpanet.org/guidance/docs/ERPANETPolicyTool.pdf

[Internet journal]HPSS. (2008, July). High Performance Storage System. Retrieved July 1, 2008, from http://www.hpss-collaboration.org/hpss/index.jsp.

[report]OAIS. (2002). *Reference Model for an Open Archival Inforamtion System*. Consultative Committee for Space Data Systems, January, 2002. Retrieved June 18, 2008, from http://public.ccsds.org/publications/archive/650x0b1.pdf.

[Internet journal]OCEAN. (2008, July). The OceanStore Project Overview. Retrieved July 1, 2008, from http://oceanstore.cs.berkeley.edu/info/overview.html.

[Internet journal]PI. (2008, July). Persistent Identifiers. Retrieved July 1, 2008, from http://www.persistent-identifier.de/.

[Internet journal]SNIA. (2008, June). Storage Networking Industry Association about page. Retrieved June 18, 2008, from http://www.snia.org/about.

[report]Todd. (2006). *Comparing the XAM API with File System Programming*. Master's Thesis, Computer Science Department of the University of New Hampshire, December, 2006. Retrieved June 18, 2008, from http://cs.unh.edu/ToddThesis.pdf.

[report]TRAC. (2008, June). *Trustworthy Repositories Audit & Certification.* Retrieved November 10, 2008, from http://www.crl.edu/PDF/trac.pdf.

[Internet journal]XAM Benefits. (2008, June). XAM Mission and Goals. Retrieved June 18, 2008, from http://www.snia.org/forums/xam.

[Internet journal]XAM Genesis. (2007, June). XAM Technical Tutorial (Slide 7). Retrieved October 7, 2008, from http://www.snia.org/forums/xam/resources/XAM_eXtensible_Access_Method_Technical_Tutorial.pdf.

[Internet journal]XAM Plugfest. (2007, October). Computer World XAM Interop Article. Retrieved October 7, 2008, from http://www.computerworld.com/action/article.do?command=viewArticleBasic&taxonomyName=storage&articleId=9042678&taxonomyId=19&intsrc=kc_top.

[Internet journal]XAM Press Release. (2008, April). SNIA XAM Specification Boosted by Software Development Efforts; Nears Public Release. Retrieved June 18, 2008, from http://www.snia.org/forums/xam/news/.

[Internet journal]XAM v1.0. (2008, August). Announcing the Availability of the EMC Centera SDK for XAM v1.0; Nears Public Release. Retrieved November 11, 2008, from http://www.snia.org/forums/xam/news/.