eXtensible Access Method (XAM) - a new fixed content API

Mark A Carlson, SNIA Technical Council, Sun Microsystems, Inc.
The material contained in this tutorial is copyrighted by the SNIA.

Member companies and individuals may use this material in presentations and literature under the following conditions:
- Any slide or slides used must be reproduced without modification
- The SNIA must be acknowledged as source of any material used in the body of any document containing material from these presentations.

This presentation is a project of the SNIA Education Committee.
Abstract

eXtensible Access Method (XAM) - A new fixed content API

XAM Provides:
- Interoperability: Applications can work with any XAM conformant storage system; information can be migrated and shared
- Compliance: Integrated record retention and disposition metadata,
- ILM Practices: Framework for classification, policy, and implementation
- Migration: Ability to automate migration process to maintain long-term readability
- Discovery: Application-independent structured discovery avoids application obsolescence.

Learning Objectives:
- For Storage Vendors: what is needed to implement a XAM interface (VIM) to their products;
- For Application Vendors: what it means to have a standard interface for supporting any fixed content storage device;
- For End Users: the value of vendor choice in fixed content applications and storage devices.
Topics

- What is Fixed Content?
- What is MetaData?
- What is SNIA doing?
- What is XAM?
- XAM API
- XAM SDK
What is Fixed Content?

- A type of data classification that indicates the bits are no longer changing
  - Classifying this way enables storage systems to meet the requirements of this type of data
- Most data is created “fixed”
  - Photos, videos, published/emailed documents, etc.
- 70-90% of data becomes fixed at some point
  - Even transactional data becomes fixed typically within a week
- Fixed content data is GROWING at 90% year over year
What is Metadata?

- Metadata allows for the creation of self-describing objects
- Self-describing objects enable content portability across client applications
- Metadata and location independence enable ILM across the managed storage resources
  - Intelligent decisions can be made to allocate specific content types to specific storage resources

<table>
<thead>
<tr>
<th>OBJECT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date: 2006/09/15</td>
</tr>
<tr>
<td>Patient: John Doe</td>
</tr>
<tr>
<td>Content Class: X-RAY</td>
</tr>
</tbody>
</table>
The need for MetaData Standards

- Which can contains corn?
- Open the cans.
- How much does it cost?
- Ask the clerk.
- How many calories does it have?
- Ask the vendor.
- How does the store automatically manage inventory?
- They can’t.
Standardized labeling allows multiple vendors to consistently represent information to consumers.

Extended labeling for LOB uses
Use of MetaData Standards

Email object stored by XAM SDK

```java
com.acme.email.from = "bugs bunny"
com.acme.email.from.role = "analyst"
com.acme.email.to = "daffy duck"
com.acme.email.to.role = "trader"
com.acme.email.subj = "what's up doc?"
com.acme.email.numattach = 2
{ Email contents }
{ Attachment #1 }
{ Attachment #2 }
```

Email Analysis Program

Can access Email metadata and, without the help of the Email Service, analyze whether the sender is allowed to send to the recipient. For example, a stock analyst may not be allowed to send information to a trader.

XAM specifies how metadata is represented, but not the actual metadata field names and values.

Further work is needed to standardize metadata names and allowed values for application domains like Email, Health, and Document Management.
The SNIA XAM Initiative is chartered to drive adoption of XAM specification, and ensure that the specification fulfills market needs for a fixed content data management interface standard.

The SNIA Fixed Content Aware Storage Technical Work Group (FCAS TWG) is the center of technical activities related to new application-level interfaces for storage of unchanging data (fixed content) and associated metadata.

The SNIA Software Development Kit Technical Working Group (XAM SDK TWG) is chartered to develop SNIA Software which implements current and future versions of the XAM Specification(s).

Your partners and competitors are already participating.

Don’t be left out!
“Information independence for applications and storage”
XAM makes this possible

As seen at SNW Spring
Multi-Vendor demonstration based on XAM

Commercially Available Applications
- Records & Documents (Vignette)
- Disk Extender (EMC)
- RIM4DB/Outerbay (HP)
- Photo Editor (Sun)

Custom Application
- XSET Browser
- XAM Query Tool

Contributed Utilities

XAM Interface
- HP RISS
- EMC Centera
- Sun
The XAM initiative is

a SNIA Initiative

driven by the storage industry

to define and promote adoption of a standard application programming interface (the XAM API)

between “Consumers” (application and management software)

and “Providers” (storage systems)

of Fixed Content storage services
What is XAM?

- **XAM is a SNIA Architecture**
  - The XAM Architecture spec defines the normative semantics of the API for use by applications and implementation by storage systems

- **XAM is an Application Programming Interface (API)**
  - The XAM Java API spec defines the binding of the XAM Architecture to the Java Language
  - The XAM C API spec defines the binding of the XAM Architecture to the C Language

- **XAM is SNIA Software**
  - The XAM SDK provides a common library and reference implementation to promote widespread adoption of the standard
Why XAM?

The industry will benefit from a standardized access method to Fixed Content

End Users want:
Choices between Application Vendors
Choices between Storage Vendors
Easy migration between vendors/technology
Compliance, Scalability, Performance, $/GB, TCO

Application Vendors want:
Annotate Data with associated Metadata
Indicate basic Storage Management Policies
Speak same language to all types of Devices
Manipulate billions if not trillions or “records”

Storage Vendors want:
Application Support for their Products
Efficiently Store Application Data and Metadata
Integrate Basic Storage Management Capabilities
Manage billions if not trillions of “records”
Resource Domains are a way of classifying services into specific areas that each deal with a different aspect of the problem.

*An information domain application creates data and associates MetaData with it*

Certain Data Domain Interfaces can accommodate both Data and MetaData (XAM, Filesystems with extended attributes).

MetaData aware Data Services interpret System MetaData as the Data requirements for its lifecycle and implement policies for retention, placement, lifecycle, etc.

Storage Domain interfaces (based on blocks or objects) provide virtualized Containers for the Data bits and the management of those containers.

Storage services are employed to meet those requirements at this point in the data’s lifecycle, however the storage services are unaware of the data’s requirements.
XAM System MetaData

- XAM specifies *property fields* that are interpreted by an Xsystem as System MetaData.
- It does this by using a reserved field namespace: `.xam.*`, `.xsystem.*` and `.xset.*` are reserved.
- Either the Content Application or a separate Information Management application can manipulate these fields to cause the underlying system to treat the data according to the specified requirements, i.e. Retention.
XAM SDK TWG Charter

- Develop SNIA Software that implements the XAM Library.
- Develop SNIA Software that implements a Reference Vendor Implementation Module (VIM) on top of an existing filesystem.
- Develop sample XAM Client Applications as SNIA Software to provide simple unit tests for portions of the XAM Specification(s).
- Develop documentation as appropriate for the above deliverables.
The XAM SDK is dynamically linked by each application wishing to connect to and use XSystems.

The XAM SDK includes several components:

- **XAM Library**, which implements the XAM API functions
- **Reference VIM**, which implements the reference behavior of an XSystem
- a framework which allows plug-able **Vendor VIMs**
- optional **XAM Toolkit** Libraries for convenience functions
Proliferation Questions

SDK Ecosystem

- Standardization Process
- Development & QA
- Integration & Distribution
- Interoperability Certification
- Licensing Schema
- Support & Maintenance

XAM
C/Java
API Spec

XAM Toolkit

XAM Library

Reference VIM

Vendor A VIM

Vendor X VIM

XAM Arch Spec

VIM API (part of the language binding)
The Low Hanging Fruit

Developed, Supplied & Supported by any 3rd Party (incl. Vendors, ISVs etc)

Shipped with Operating Systems

Supplied & Supported by Individual Vendors of XAM Storage Systems

XAM API Spec

XAM Toolkit

XAM Arch Spec

VIM API Spec

XAM Library

Reference VIM

Vendor A VIM

Vendor X VIM
SNIA’s “FCAS TWG” maintains and periodically publishes set of normative XAM standard Specs

XAM Standard defined by SNIA’s “FCAS TWG”:
  - direct influence by SNIA FCAS membership
  - indirect influence by Storage Vendors assimilating feedback from ISVs, End-Users

SNIA publishes normative specs for
  - XAM Architecture
  - XAM C API
  - XAM Java API

XAM Standard updated at most once a year

XAM Standard Versions must be backwards compatible
SNIA “XAM Software TWG” Develops and Maintains beta-quality ‘Gold’ Distribution’ of XAM SDK under BSD License

‘Gold’ XAM SDK (XAM Library, Reference VIM) developed and maintained by SNIA’s “XAM Software TWG” member companies
Specific “reference” platforms (i.e. Windows)

Beta-quality code of SNIA’s XAM SDK (‘Gold’ Distribution) available to SNIA TWG Member Companies under BSD-type license

Periodic work in progress releases to public

After SNIA membership vote, released as Version 1.x

SNIA’s “XAM SDK TWG” provides the last tier of support for the XAM SDK ‘Gold’ distribution
Storage Vendors (e.g. EMC, IBM, HP, Sun, ...) derive their individual product-quality XAM SDK Derivatives from SNIA’s ‘Gold Distribution’
Supported for their operating systems as well (Solaris, HP-UX, AIX)

Storage Vendors responsible for porting, QA, distribution and ongoing maintenance of their XAM SDK Derivative according to Storage Vendors’ product schedule/plans including releases of XAM SDK Derivatives and Service Packs

Storage Vendors responsible for interoperability testing of their XAM SDK Derivative against other vendors’ released VIMs

Storage Vendors must feed bug fixes, enhancements and new features back to SNIA’s “XAM SDK TWG”, for inclusion into the next ‘Gold’ Distribution

Storage Vendors provide 1st tier of support for their XAM SDK Derivative
ISVs Integrate and certify their apps with a chosen XAM SDK Derivative
Based on the platforms/OS they will support

ISVs integrate their applications with one or more chosen XAM SDK Derivative(s), under the Member Company’s respective licensing schema

ISVs responsible for interoperability testing and certification of their s/w applications against the chosen XAM SDK Derivative(s)

Ongoing support and problem resolution is negotiated between the Member Company and the ISV, as today

ISVs ship their integrated applications to End-Users under their respective licensing schema
ISVs, Storage Vendors ship their products to End-Users with certified interoperability guarantees

Integrated product licensed to End-User under ISV’s and Member Company’s licensing schema

ISVs, Member Company provide direct support to End-User

Member Company provides direct support to ISVs

SNIA “XAM SDK TWG” provides direct support to Member Companies
SNIA’s “FCAS TWG” maintains and periodically publishes set of normative XAM standard specs

SNIA’s “XAM Software TWG” Develops and Maintains beta-quality ‘Gold’ Distribution’ of XAM SDK under BSD License

SNIA’s Member Companies (e.g. EMC, IBM, HP, HDS, ...) derive their individual product-quality XAM SDK Derivatives from SNIA’s ‘Gold Distribution’

ISVs Integrate and certify their apps with a chosen Member Company’s XAM SDK Distribution

ISVs, Member Companies ship their products to End-Users with certified interoperability guarantees
The XAM SDK

developing XAM Library software

- XAM will consist of a set of components.
  - The ‘topmost’ library will contain the public XAM interfaces; thus, only the topmost library will be directly referenced by applications that wish to integrate with the XAM API.
  - Extension libraries may also be provided which implement higher levels of functionality (e.g., placing an export method, an import method, and a delete method in series to create a ‘move’ function). When such libraries are provided, applications may wish to reference these libraries as well.

- The actual implementation of the interfaces will be in the VIMs (Vendor Interface Modules). A XAM Library may utilize one or more VIMs.

- Components will be produced in both C/C++ and Java
Design Goals

- Provide a generic interface for applications
  - XAM API methods have the same syntax and semantics without regard to the underlying storage. No methods exist that “lock-in” an application to a specific storage system; in fact, the systems themselves should be semantically indistinguishable when viewed from the XAM API.

- Minimal yet complete
  - Keep the interface simple and small (e.g., have as few API methods as possible, and keep these methods easy to use and understand), while ensuring that the methods make all forms of data manipulation possible. If functionality could have been achieved by composing other methods (in a way that sufficiently ensures performance and scalability), then a new method is not created for that function.

- Expose no implementation detail
  - Do not expose any internal functionality that would serve to place restrictions on storage system vendors.
XAM Interface semantics are organized around objects

- Primary objects
  - XAM Library
  - XSystem
  - XSet

- Secondary objects
  - XStream
  - XIterator
XAM Primary Objects

Load library

Unload library

Field Editing

XAM Library

Connect to an XSystem

Close an XSystem

Field Editing

XSystem

Field Editing

Open/Create an XSet

Close an XSet

XSet

Field Editing

Job control
XSet import/export
XSet commit

XSystem authentication
XSet administration
XAM Secondary Objects
(with relationship to Primary Objects)

- XSet, XSystem, or XAM Library
- XStream
- XIterator
- XIterator ops

- Open XIterator
- Close

- Field Editing
- Open/Create an XStream
- XStream Ops
- XStream Ops
XAM Interface semantics are organized around objects

- **Primary objects**
  - XAM Library
  - XSystem
  - XSet
- **Secondary objects**
  - XStream
  - XIterator
Elements of Primary Objects

XAM Library object
- No constructor is available
  - This is a singleton
  - This must be available as a static object or thru the use of a static accessor method
- Contains fields

XSystem object
- No constructor is available
  - This must be available by calling a factory method on the XAM Library object
- Contains fields

XSet object
- No constructor is available
  - This must be available by calling a factory method on the XSystem object
- Contains fields

Diagram:

- XAM Lib
- Load
- Unload
- Connect
- Close
- Open/Create
- XSystem ops
- XSet ops

Field Ops

Diagram details include:
- XAM Lib
- XSystem
- XSet

Diagram arrows indicate:
- Load
- Unload
- Connect
- Close
- Open/Create
- XSystem ops
- XSet ops

Diagram legend:
- XAM Lib
- XSystem
- XSet

Diagram notes:
- Field Ops

Diagram elements:
- Load
- Unload
- Connect
- Close
- Open/Create
- XSystem ops
- XSet ops

Diagram connections:
- XAM Lib to XSystem
- XSystem to XSet
- XSet to XAM Lib

Diagram documentation:
- eXtensible Access Method (XAM) - A new fixed content API
- © 2008 Storage Networking Industry Association. All Rights Reserved.
Elements of Primary Objects

XAM Library object
- No constructor is available
  - This is a singleton
  - This must be available as a static object or thru the use of a static accessor method
- Contains fields

XSystem object
- No constructor is available
  - This must be available by calling a factory method on the XAM Library object
- Contains fields

XSet object
- No constructor is available
  - This must be available by calling a factory method on the XSystem object
- Contains fields

Field Ops

XAM Lib

Load
Unload

Field Ops

XSystem

Connect
Close

Field Ops

XSet

Open/Create
Close

Field Ops

eXtensible Access Method (XAM) - A new fixed content API
© 2008 Storage Networking Industry Association. All Rights Reserved.
UML of the Primary Object

```
<interface>
org.snia.xam.XAM
+ connect()
+ setLogger()
+ openXUIDiterator()
</interface>

<interface>
org.snia.xam.XSystem
+ setLogger()
+ authenticate()
+ close()
+ abandon()
+ deleteXSet()
+ holdXSet()
+ releaseXSet()
+ accessXSet()
+ getXSetAccessTime()
+ createXSet()
+ openXSet()
+ copyXSet()
</interface>

<interface>
org.snia.xam.XFieldContainer
+ openFieldIterator()
+ createProperty()
+ getFloat()
+ setProperty()
+ setBoolean()
+ setDate	
+ fieldLength()
+ fieldLength()
+ fieldBinding()
+ fieldReadonly()
+ deleteField()
</interface>

<interface>
org.snia.xam.XSet
+ applyManagementPolicy()
+ setMinimumRetentionDurationPolicy()
+ setMinimumRetentionDuration()
+ applyEventRetentionEnabledPolicy()
+ setEventRetentionEnabled()
+ applyEventRetentionEnabledDurationPolicy()
+ setEventRetentionEnabledDuration()
+ setEventRetentionDuration()
+ setEvent()
+ applyAutoDeletePolicy()
+ setAutoDelete()
+ applyStoragePolicy()
+ getActualMinimumRetentionDuration()
+ getActualAutoDelete()
+ abandon()
+ commit()
+ close()
+ submitJob()
+ haltJob()
+ openExportStream()
+ openImportStream()
</interface>
```
**Elements of Secondary Objects**

**XIterator object**
- **No constructor is available**
  - This must be available by calling a factory method on a Primary object (implements Field Container)

**XStream object**
- **No constructor is available**
  - This must be available by calling a factory method on a Primary object (implements Field Container)
UML of the Secondary Objects

```
interface org.snia.xam.XStream
+tell()
+seek()
+write()
+read()
+close()
+abandon()
```

```
interface org.snia.xam.Logger
+setLevel()
+getLevel()
+fatal()
+error()
+warn()
+info()
+trace()
+debug()
```
The VIM Interface roughly maps to the public XAM API

- Each object in the XAM API should have an analog in the VIM Interface
- Each method in the XAM API should have an analog in the VIM Interface
- NOTE: For Java - the VIM instances are interacted with “directly” by the application (no need to proxy/copy)
Control Flow

- Note that in all cases, the VIM is accessed thru the XSystem
  - There is no public interface that exposes the VIM to the application.

- Possible control flows:
  - Objects created by a VIM are directly passed to the application.
  - Objects created by a VIM are decorated by the XAM Library and the references are indirectly passed to the application; the XAM Library thus holds references to the objects and resolves references for application.
Direct control (used by Java)

1. Application
2. XAM Library
3. Java VIM
4. XSystem
5. XSet

- Application connects to XAM Library
- XAM Library locates VIM from XRI, creates instance of VIM
- New VIM instance
- Authenticate
- Authenticate return
- Create XSet
- New XSet
- Create Property
- Create Property return
Indirect Control (used by C/C++)

- Application
- XAM Library
- JNI VIM
- C VIM

**Sequence Diagram**

1. **Application** connects to **XAM Library** with `connect()`.
2. The library requests creation of a **new VIM instance**.
3. The **JNI VIM** loads the **C VIM** and maintains a reference to it.
4. The **XAM Library** uses `authenticate()` and `createXSet()`.
5. The **JNI VIM** responds with `C::authenticate()` and `C::createXSet`.
6. The **C VIM** returns `C::createXSet return`.
7. An **XSet** is created with `new XSet`.
8. A **createProperty(int)** is added with `createProperty(int)`.
9. The **C VIM** returns `C::createInt return`.

**Notes**:
- **XAM Library** locates VIM from XRI, creates the instance of VIM.
- The **JNI VIM** code loads C-VIM and maintains reference to VIM.
As noted before, the application binds to the XAM API

- Applications should never bind to the VIM interface!
- It is the responsibility of the XAM Library to call into the VIM, not the application.

The VIM interacts with the Storage System

- The XAM Library never interacts directly with the underlying Storage System; all ‘communication’ is routed thru the VIM
XAM Application software stack (C/C++)

C/C++ Application

XAM Java API (JNI)

XAM C API

XAM Library logic (C++)

‘outgoing’ VIM C API

VIM Java API (JNI)

VIM C API

VIM C

A new fixed content API
XAM Application software stack (Pure Java)

C/C++ Application

- XAM C API
- XAM Library logic (C++)
- ‘outgoing’ VIM C API
- VIM C API
- VIM

Java Application

- XAM Java API
- XAM Library logic (Java)
- ‘outgoing’ VIM Java API
- VIM Java API
- VIM Java

XAM Java API (JNI)

XAM Java API (JNI)

© 2008 Storage Networking Industry Association. All Rights Reserved.
XAM Application software stack (Unified)
A ‘stackable’ VIM

- XAM places limited constraints on the implementation of the VIM; only that it must implement the VIM interface
  - A VIM is allowed to call other VIMs.
- A ‘stackable’ VIM is a VIM that is capable of calling into the VIM Interface of other VIMs.
- This fully unifies the software model
  - Any VIM can be used with any XAM Library if an appropriate stackable VIM exists
XAM Application software stack (Fully Unified)

- C/C++ Application
  - XAM C API
  - XAM Library logic (C++)
  - ‘outgoing’ VIM C API
  - VIM Java API (JNI)

- Java Application
  - XAM Java API
  - XAM Library logic (Java)
  - ‘outgoing’ VIM Java API
  - VIM Java API (JNI)
  - VIM C

A new fix

© 2008 Storage Networking Industry Association. All Rights Reserved.
HTTP VIM Example

The HTTP VIM is an example of a stackable VIM.
Where To Go

SNIA XAM Home

- http://www.snia.org/xam

SNIA FCAS TWG
(XAM Technical Work Group)


SNIA XAM SDK TWG
(XAM SDK Technical Work Group)

Refer to Other Tutorials

Please use this icon for referral to other tutorials where appropriate

Check out SNIA Tutorial:
Enter Tutorial Title Here
Q&A / Feedback

Please send any questions or comments on this presentation to SNIA: trackvirtualization@snia.org

Many thanks to the following individuals for their contributions to this tutorial.

- SNIA Education Committee

FCAS TWG
XAM SDK TWG
XAM Initiative
Zoran Cakeljic