

XAM over OSD

Sami Iren
Seagate

Kalman Meth
IBM

Rich Ramos
Xyratex

September 2007

Abstract

XAM and OSD are both object-based technologies that are being standardized by SNIA.

XAM provides a standard object-based API for applications. OSD provides a standard interface for managing objects at the storage level hence relieving file systems (and applications) from low-level storage management tasks. Due to its object-based nature it also provides improved scalability and native security. OSD and XAM are complementary technologies. When built on top of an OSD, XAM implementations do not have to worry about how XAM objects and metadata are stored on the physical media; they will be mapped to OSD objects and attributes.

This presentation provides an overall feel on how XAM can operate on top of an OSD system.

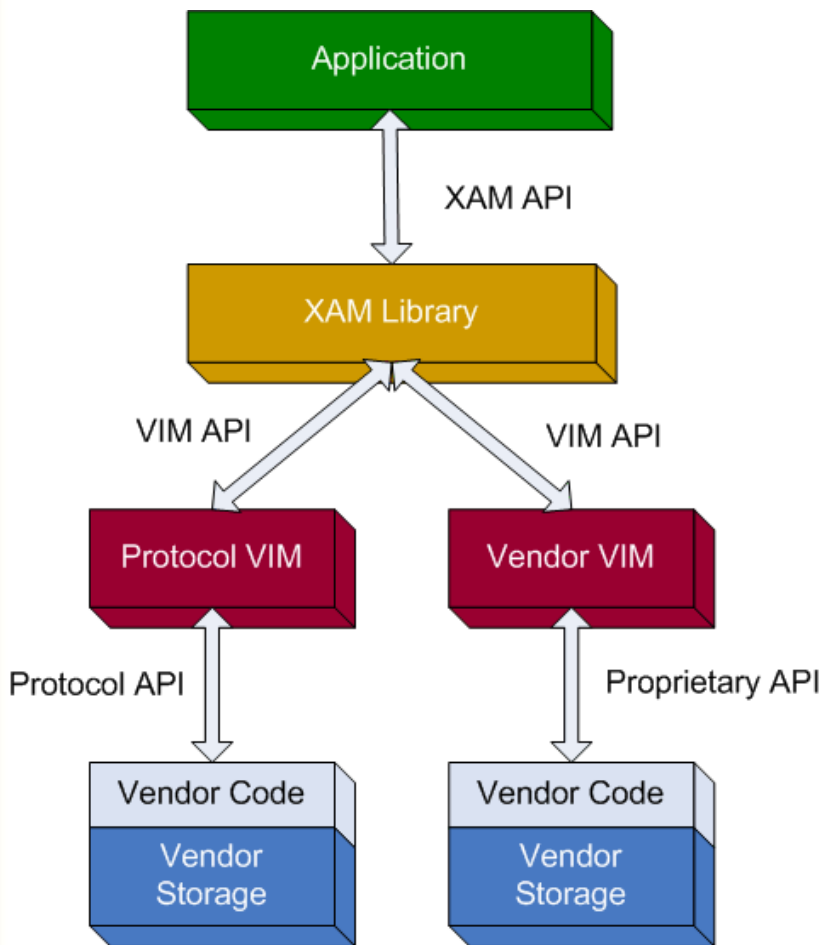
This is an ongoing work; hence open to feedback/suggestions.

Agenda

- XAM Architecture
- OSD Architecture
- XAM over OSD: The Infrastructure
- The Mapping
 - Object Mapping
 - Field Mapping
- Other Relevant Items
 - Management
 - Query
 - Security
- Summary

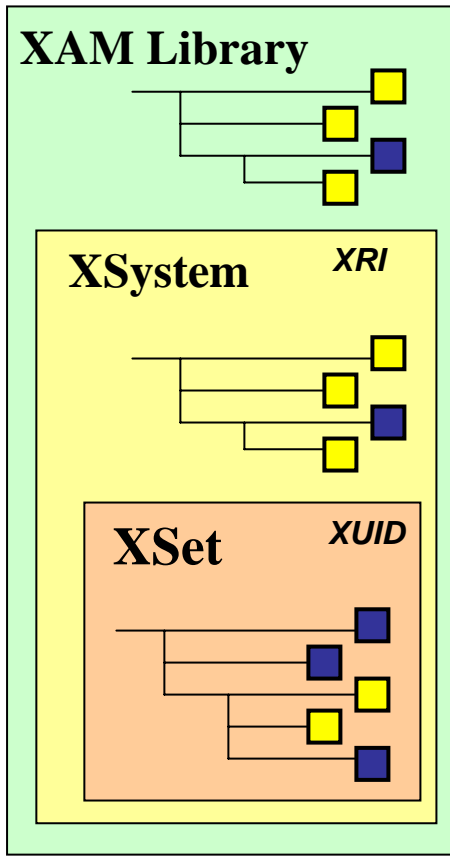
XAM Architecture

XAM Architecture



- **An application uses the libxam.dll to 'connect' to a specified XSystem.**
 - A single application may connect to multiple XSystems simultaneously
 - Multiple applications may connect to a single XSystem simultaneously
- **An XSystem is not identical to a vendor's "storage box", but a logical abstraction which should be viewed as 'bag of storage'.**
- **The application may be required to authenticate at the time the connection to an XSystem is established.**
- **The application uses libxam.dll to store/retrieve "content objects" to/from the XSystem.**
- **These "content objects" are bundles of data and metadata, and are called XSets.**

XAM Objects (1)

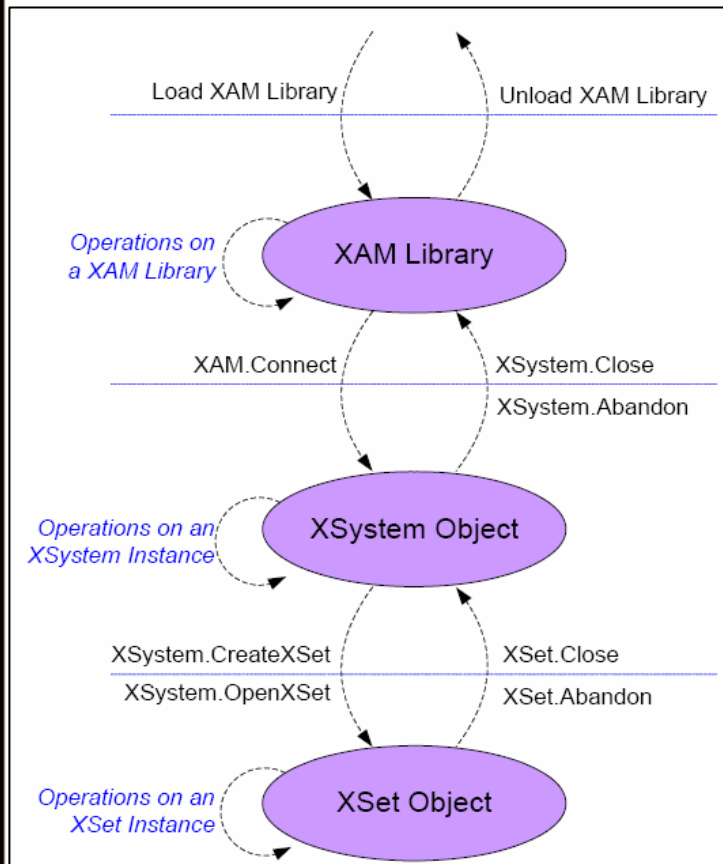


 Properties
 Xstreams

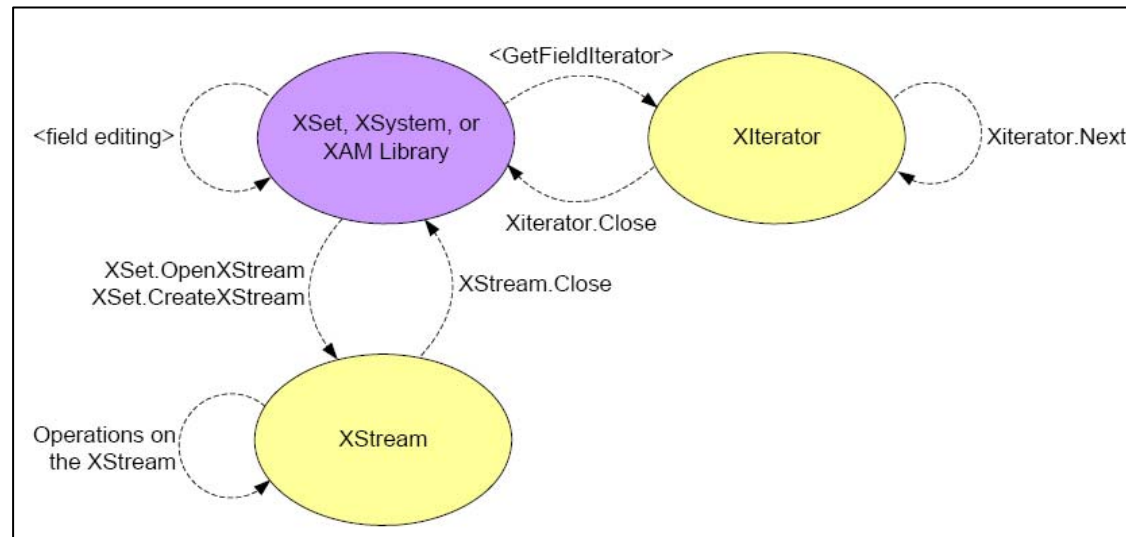
- 3-levels of **primary** objects (hierarchy)
 - XAM Library: top level object for the XAM API library
 - Contains methods to get and set fields describing the configuration of the XAM system
 - Acts as a factory of XSystems
 - XSystem: object that abstracts the connection between application and storage systems
 - Encapsulates any resource management associated with the connection
 - Contains methods used to authenticate operations
 - Acts as a virtual storage system, partitioning the content
 - XSet: object that contains application/user data and metadata
 - Has a globally unique identifier, called XUID (80 bytes)
- Each level of XAM abstraction (XAM Library, XSystem, XSet) contains “fields” (of type “property” or “Xstream”)

XAM Objects (2)

Primary Object Hierarchy

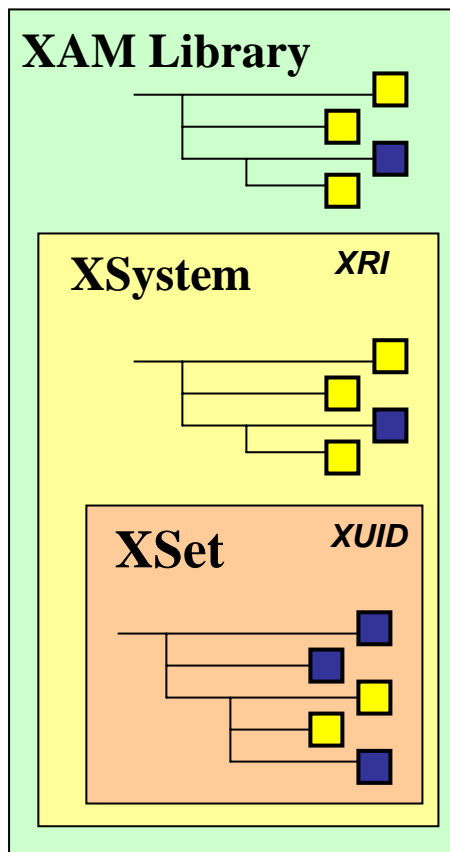


Secondary Object Hierarchy



- One or more of secondary objects can be attached to any XAM primary object.
- Two types of secondary objects:
 - XStream: used to store and retrieve data (type of field)
 - XIterator: used to discover the fields that are attached to a primary object

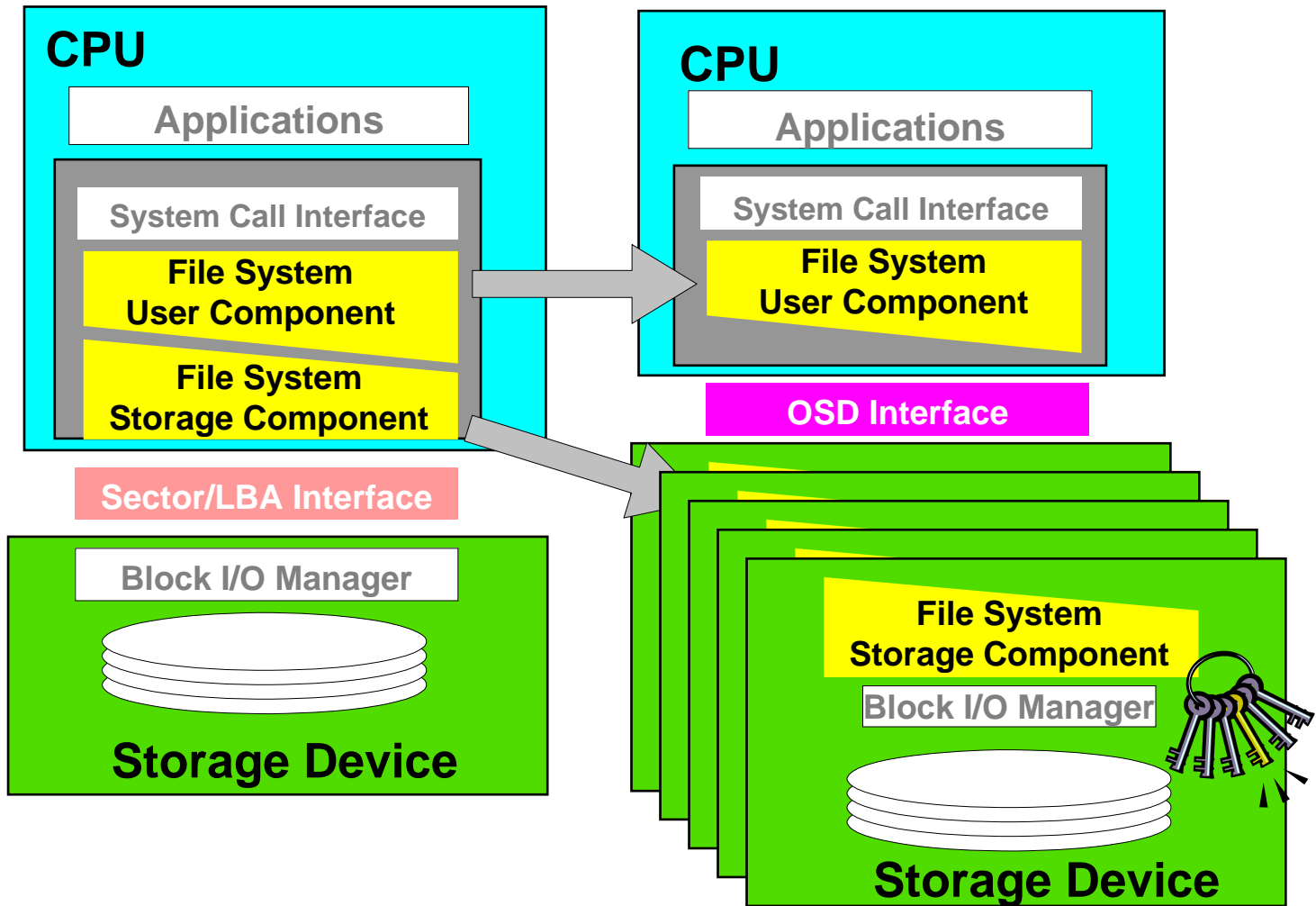
XAM Fields



- Each primary object has two types of Fields:
 - Properties
 - “Simple” Types (Boolean, Int, Float, String, DateTime, XUID)
 - Type checked/enforced by Storage System
 - Manipulated via “Property Get/Set” Methods
 - XStreams
 - Bytestreams, bound in Length
 - Type assumed to be a valid MIME-type, but not checked/enforced by Storage System
 - Manipulated via Posix-style I/O Methods (e.g. open, read, write, close)
- Each Field Has Four Basic Attributes:
 - **Type** – stype for Properties, any other MIME-type for streams
 - **Length** – The actual size of the field’s value
 - **ReadOnly** – Flag indicating whether field is modifiable by applications
 - **Fixed** – Flag indicating whether field is Fixed/Variable content
- Manipulated via “Attribute Get/Set” methods

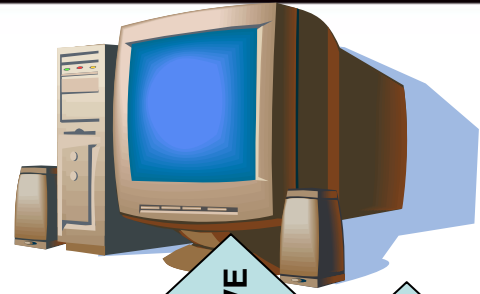
OSD Architecture

OSD Structure

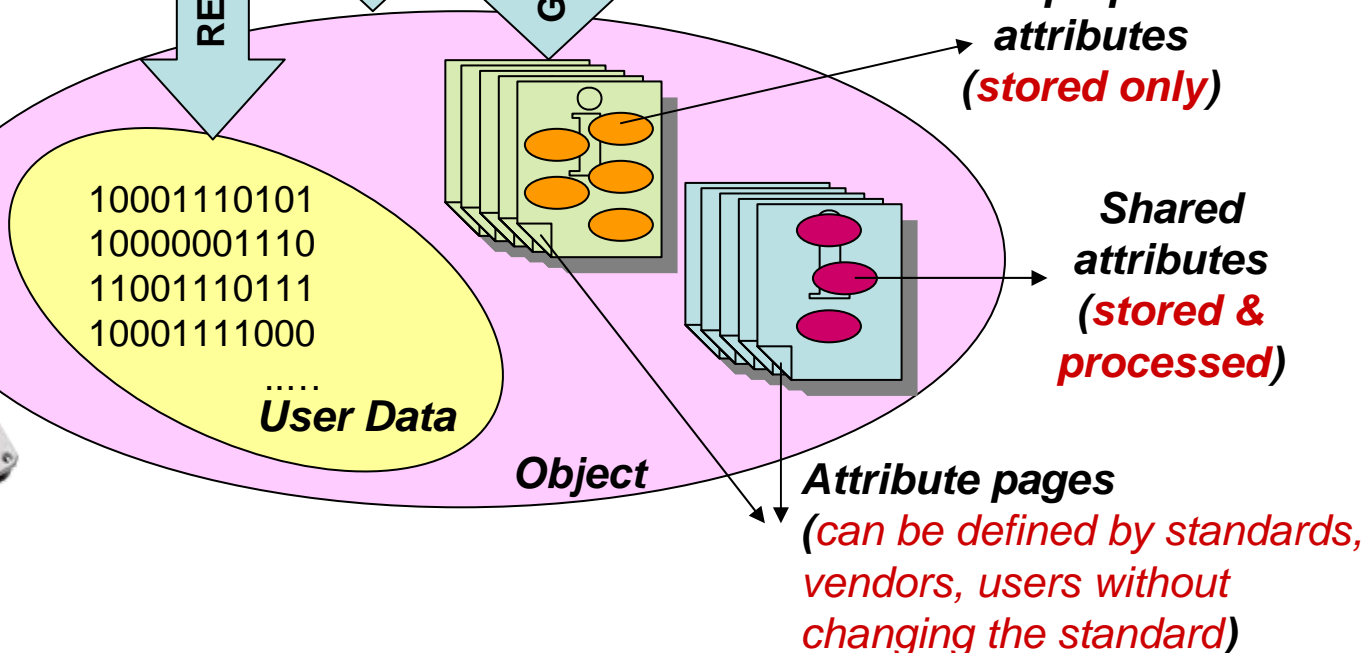
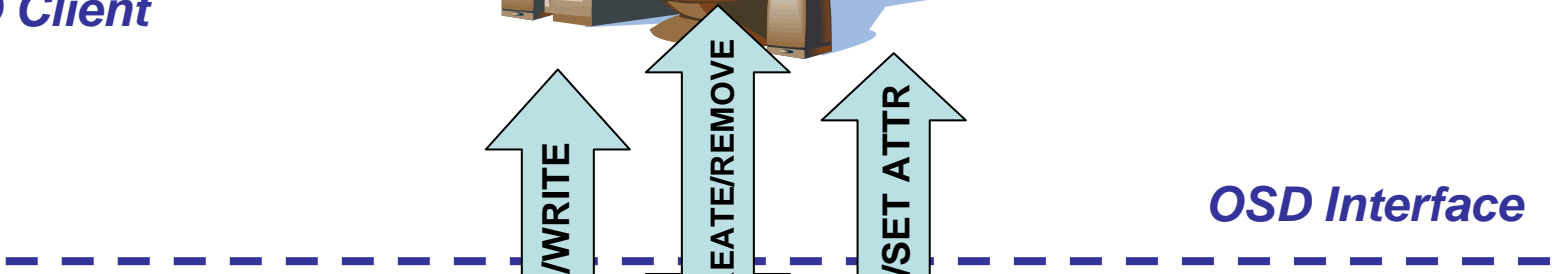
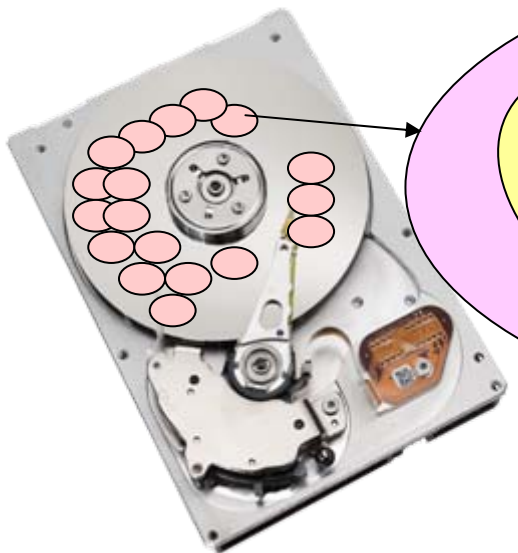


Interface

OSD Client



OSD Target



```

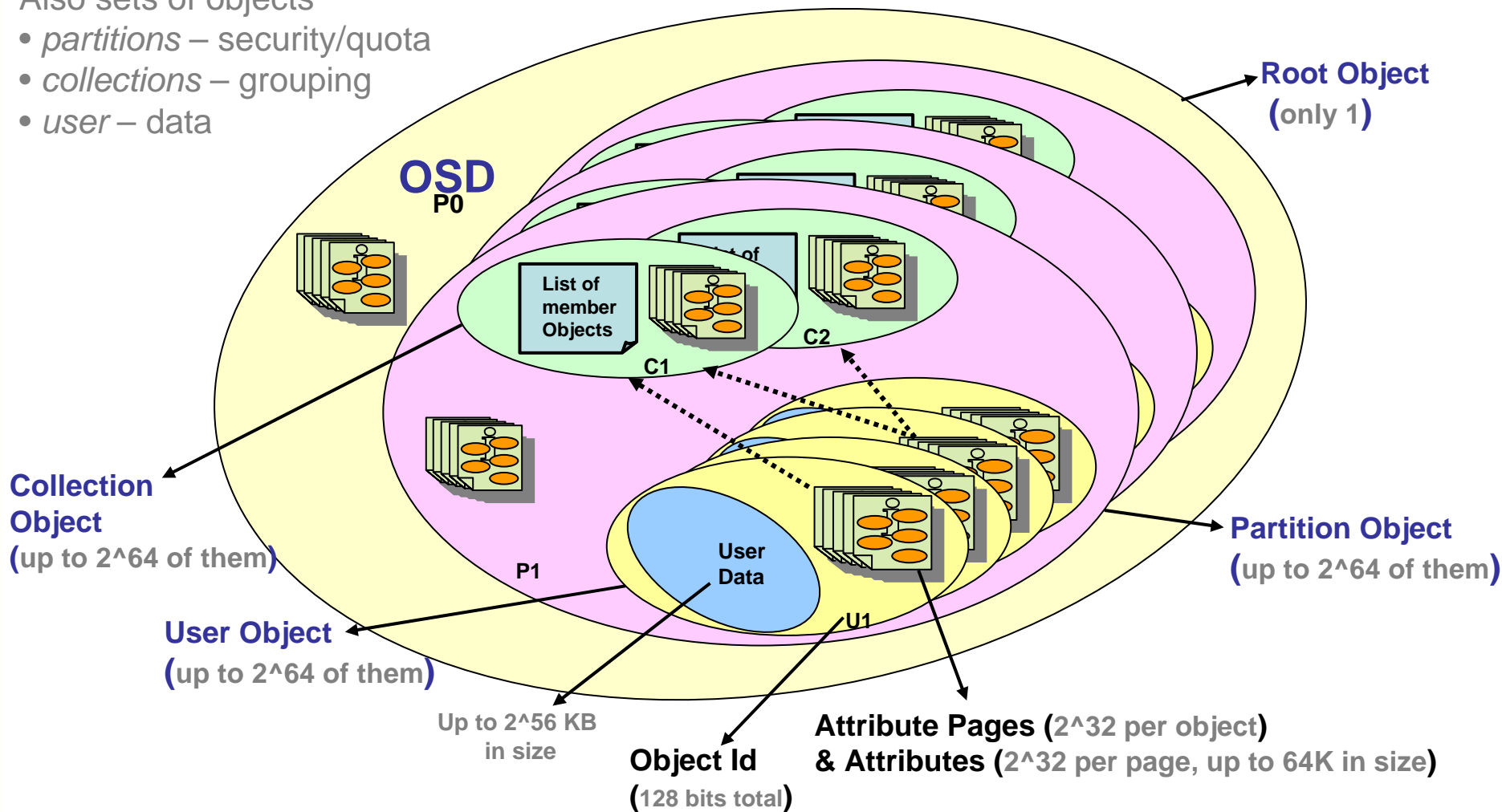
10001110101
10000001110
11001110111
10001111000
.....
    
```



OSD Objects

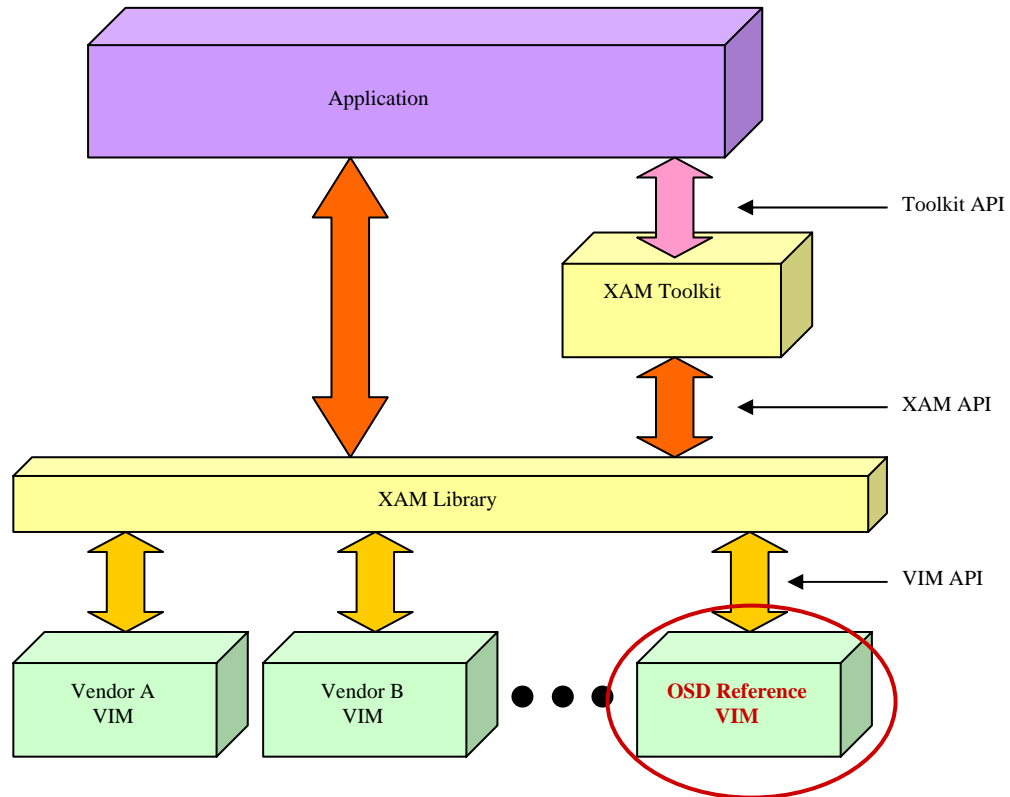
Also sets of objects

- *partitions* – security/quota
- *collections* – grouping
- *user* – data

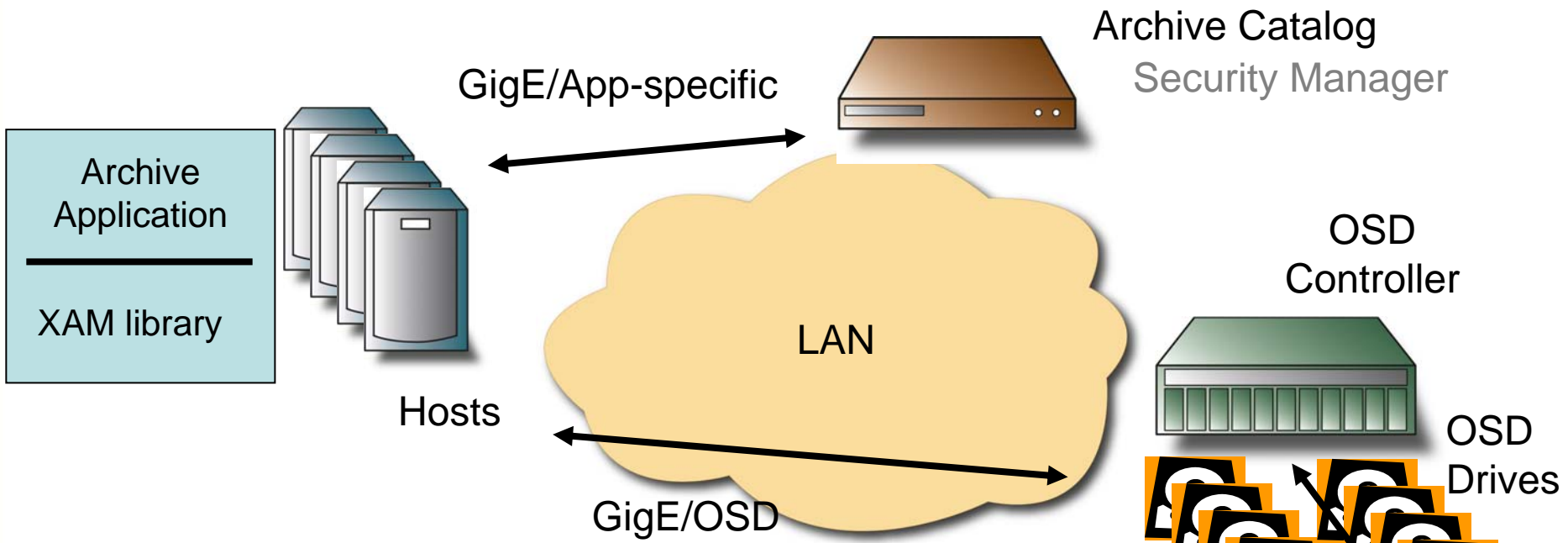


XAM over OSD

Where does this Work Fit?



CAS with OSD



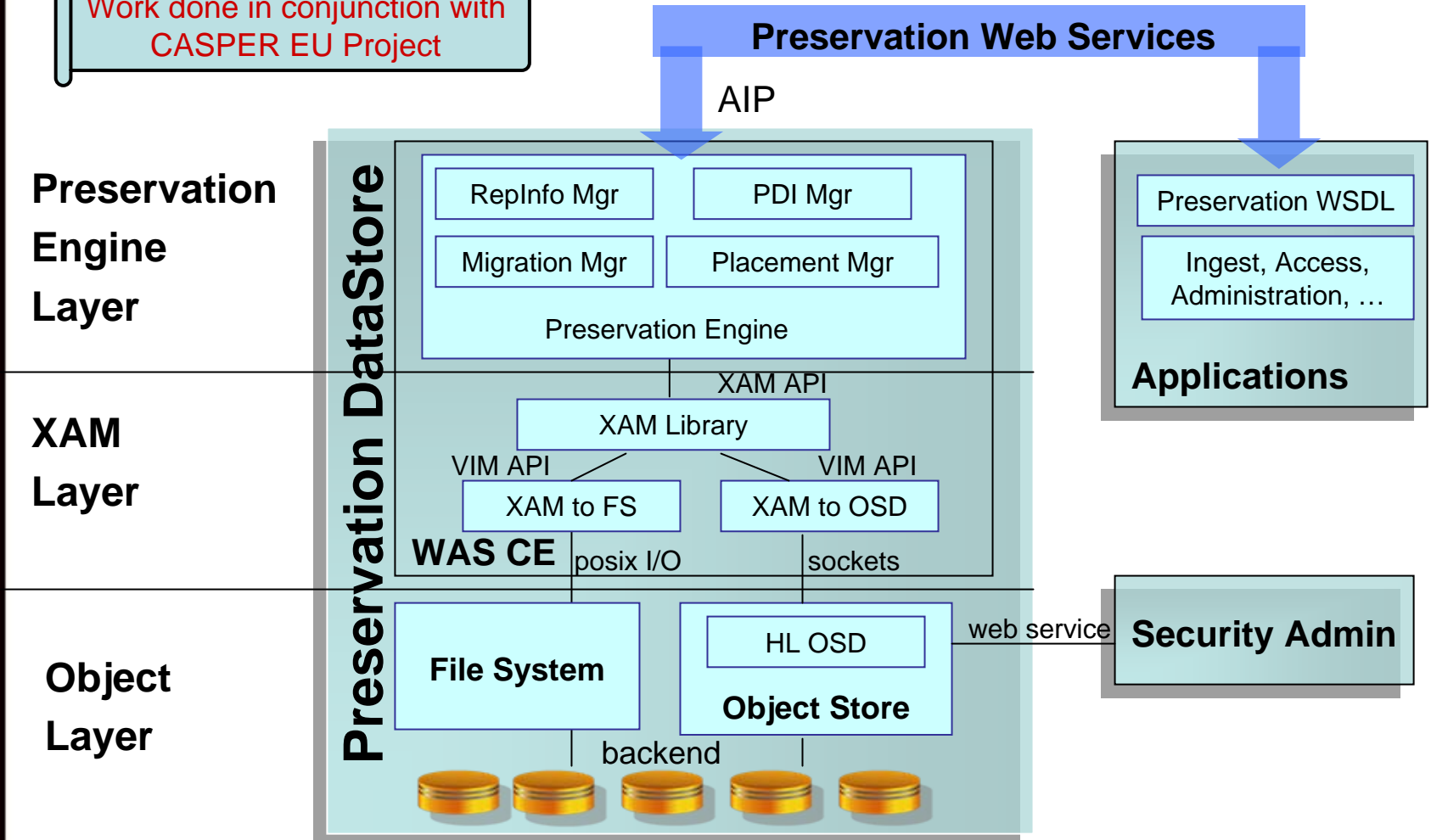
Applications use XAM library, XAM VIM translates to OSD protocol and attributes, any OSD device can be a back-end; CAS doesn't have to have a file system inside

CAS/XAM replaces "top" of file system, OSD replaces "bottom" of file system

FC/OSD

An Early Prototype: IBM's PDS Architecture

Work done in conjunction with CASPER EU Project

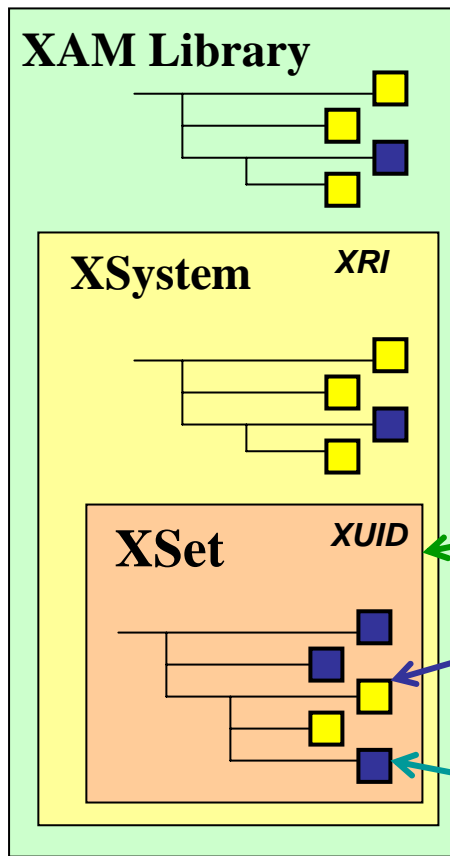


The Mapping

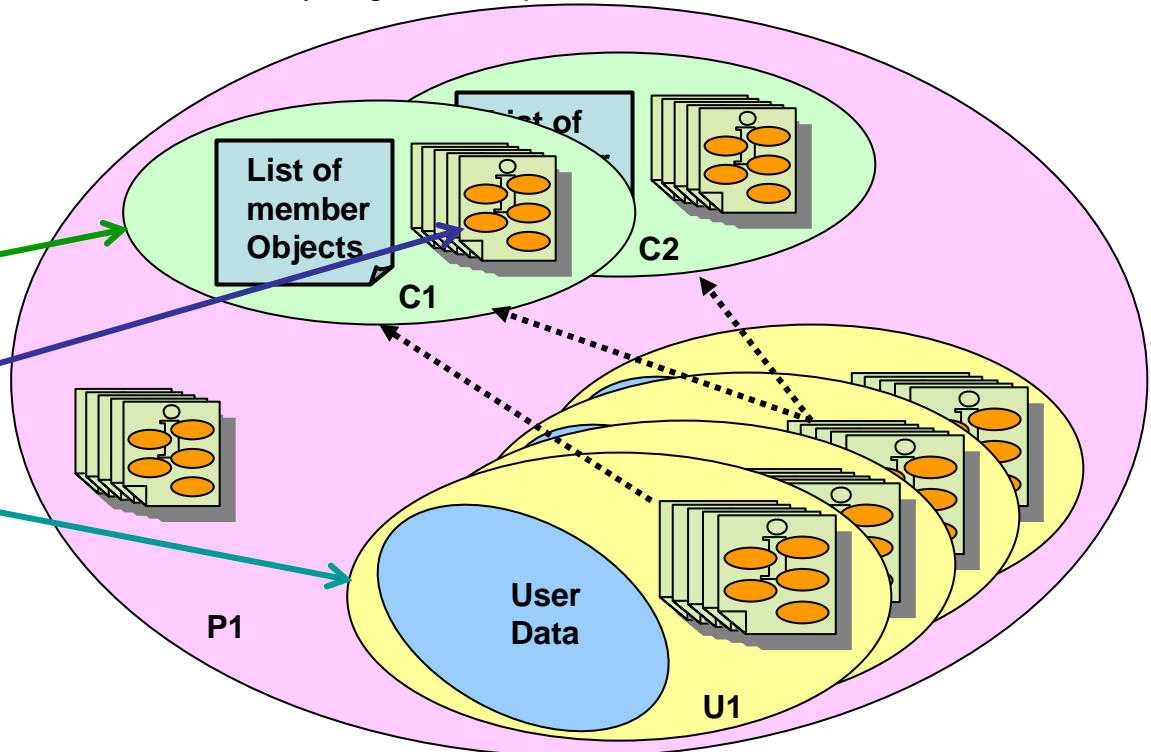
XAM to OSD Object Mapping (Option 1)

Initial OSD twg Recommendation

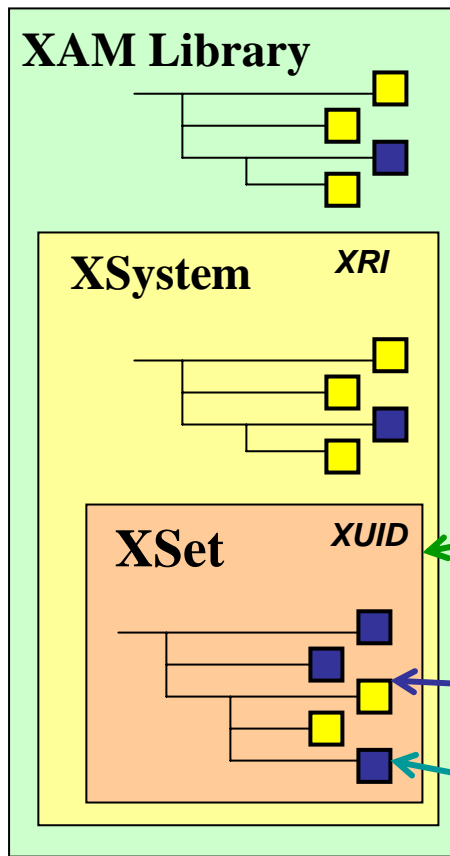
- XSETs are mapped to collection objects
- Properties are mapped to attribute pages
- Xstreams are mapped to user objects
- XSet ID (XUID) is a property, hence, stored in an OSD Attribute Page
- Field names are stored as OSD attributes
- OSD provides ways to iterate through these fields
 - LIST COLLECTION command for XStreams
 - Attribute Directory Pages for Properties



■ Properties
■ Xstreams

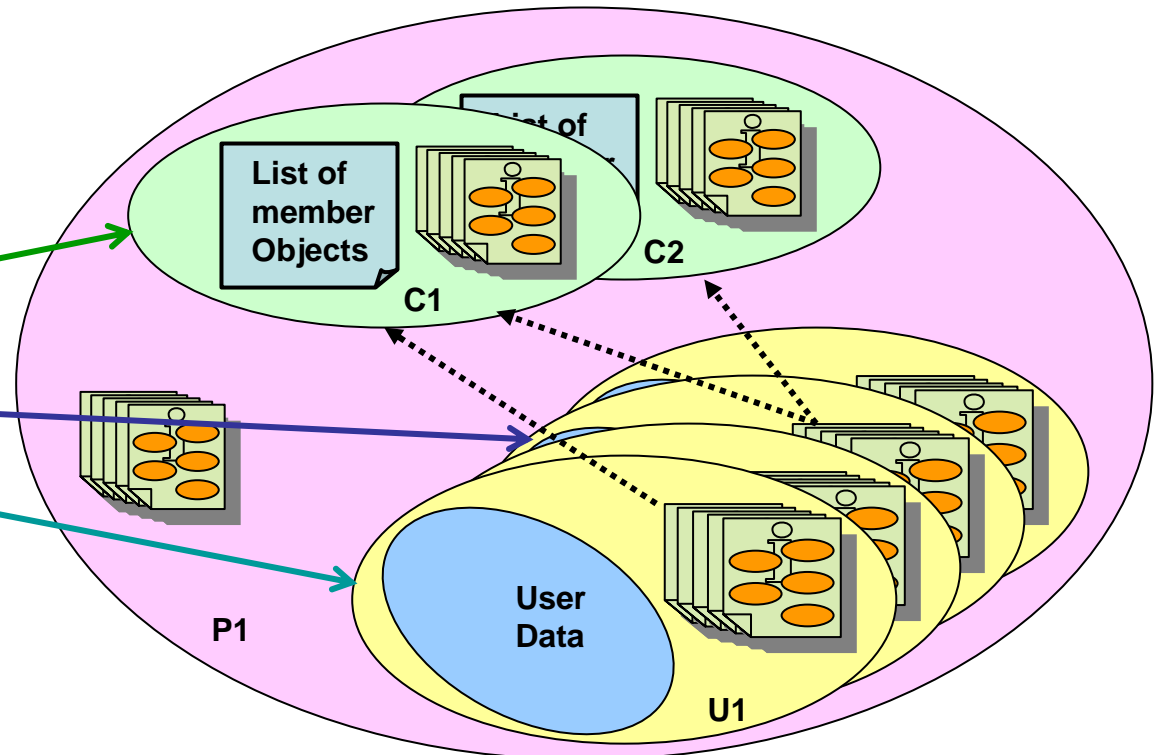


XAM to OSD Object Mapping (Option 2)



- XSETs are mapped to collection objects
- Properties & XStreams are mapped to user pages
- XSet ID (XUID) is a property, hence, stored in an OSD user object
- Field names are stored as OSD attributes
- LIST COLLECTION command is used to iterate through these fields

Properties
 Xstreams



<op> is one of create, set, or get

Method Mappings

Property Methods

Property Methods	Description
<XAM Handle>.<Op>BooleanProperty	Operate on a Boolean property.
<XAM Handle>.<Op>IntProperty	Operate on a int64 property.
<XAM Handle>.<Op>FloatProperty	Operate on a float property.
<XAM Handle>.<Op>XUIDProperty	Operate on a xuid property.
<XAM Handle>.<Op>StringProperty	Operate on a string property.
<XAM Handle>.<Op>DateTimeProperty	Operate on a datetime property.

If **properties** are mapped to **OSD Attribute Pages**:

- create -> SET ATTRIBUTES
- set -> SET ATTRIBUTES
- get -> GET ATTRIBUTES

- DeleteField -> REMOVE
- DeleteField -> SET ATTRIBUTES

Field Methods

Field Methods	Description
<XAM Handle>.DeleteField	Delete a field from the XAM object.

- GetFieldxxx -> GET ATTRIBUTES
- SetFieldxxx -> SET ATTRIBUTES

Field Attribute Methods

Field Methods	Description
<XAM Handle>.SetFieldAsBinding	Set the specified field binding attribute as TRUE.
<XAM Handle>.SetFieldAsNonbinding	Set the specified field binding attribute as FALSE.
<XAM Handle>.GetFieldType	Retrieve the MIME type value for the specified field.
<XAM Handle>.GetFieldLength	Retrieve the field length value for the specified field.
<XAM Handle>.GetFieldBinding	Retrieve the binding attribute value for the specified field.
<XAM Handle>.GetFieldReadOnly	Retrieve the readonly attribute value for the specified field.

Other Relevant Items

XSet Management

- XSet management methods are used to specify how an XSet is to be managed by the XSystem during its lifetime and when it is deleted.
 - Four classes:
 - *Retention*: longevity by which an XSystem provides XSet data access by prohibiting XSet deletion
 - Time-based vs. event-based retention
 - *Deletion*: when should this XSet be deleted?
 - *Storage*: e.g., performance, resiliency, virtualization
 - *Hold*: longevity by which an XSystem enforces read-only access and prohibits deletion
- Mapping to OSD: There are two options;
 - *Opaque*: management is completely handled at the XAM level; OSD simply stores the relevant fields/attributes and does NOT interpret them
 - No further work is required
 - *Shared*: XAM delegates some of the responsibility to OSD
 - E.g., Deletion can be performed by OSD if the XAM fields/attributes can be interpreted by OSD
 - XAM and OSD standards should agree on what those fields/attributes are
 - Alternatively, vendors can define their own attributes to handle this



XAM Query & Jobs

- A XAM job is a way to submit work to the XSystem. To date the only job defined is a query job.
- A XAM query is a way for an application to search content stored on the XSystem.
 - Search is mainly metadata (level 1), but in some cases, content of XStreams may be searched too (level 2).
- XAM Query Language (XAM QL) is used as the search language and is an SQL variant (simpler).
- OSD Mapping:
 - OSD-2 has a QUERY command that searches attributes (i.e., metadata)
 - Can be used for level 1 XAM queries, but the language is pretty basic (cannot fully support XAM QL)
 - Future versions of OSD might support advanced versions of QUERY (metadata search) and FIND (data search) commands

XAM Security

- XAM defines three security disciplines:
 - XAM Application Authentication: The XAM application must authenticate against the XSystem using SASL (RFC4422). SASL is a framework, not a specific authentication mechanism. The specific mechanism is left to the vendor to implement (eg. LDAP, AD,...)
 - XSystem Authorization: What methods can be invoked on a XAM session with an XSystem.
 - XSet Access Control: Determines what methods are allowed on a per XSet basis.
- OSD provides per request authorization
 - Authentication and access control is handled by the security manager (this could be archive catalog or some other part of the system)
 - Most of XAM security defines the role of the security manager in the OSD world

Summary

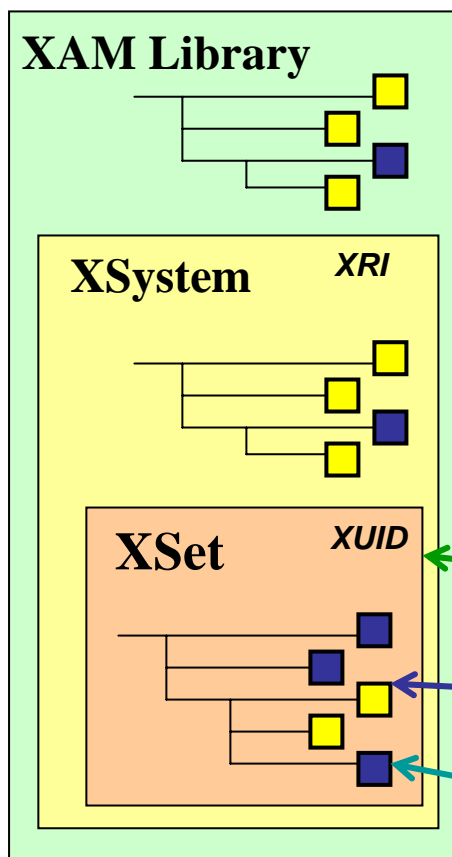
- Both XAM and OSD are object-based *complementary* technologies
 - XAM is the API for applications
 - OSD is the interface to the storage
- Metadata is “key” in both technologies
- “XAM over OSD” provides significant advantages
 - One-to-one object mapping; systems do not have to worry about low-level storage management (eliminates the need for FS functionality)
 - Inherits default OSD benefits: scalability, sharing, security
 - Certain management tasks can be offloaded to OSD (i.e., *shared* management approach)
- Effort has the blessings of OSD and XAM SDK technical working groups
- Ongoing work; we need
 - Volunteers
 - Feedback

Q&A / Feedback

- Please send any questions or comments on this presentation to snia-osd@snia.org
- Standards work
 - www.snia.org/osd
 - www.snia.org/xam
- Presenters
 - Sami Iren (sami.iren@seagate.com)
 - Kalman Meth (meth@il.ibm.com)
 - Rich Ramos (Rich_Ramos@us.xyratex.com)

Backup Slides

XAM to OSD Object Mapping (Option 3)



- XSETs are mapped to partition objects
 - Easy access control for the whole XSet (e.g., HoldXSet method)
- Fields (Properties & Xstreams) are mapped to user objects
- XAM names (XUIDs) are fields; hence stored as OSD user objects
- OSD LIST command is used to iterate through these fields
- Simple to implement & use; common methods to operate on fields
- Costly; inefficient use of storage & name spaces

■ Properties
■ Xstreams

