Technical Council Presentation to OGF on XAM

David Black, SNIA Technical Council, EMC

SNIA
Technical Council

Topics

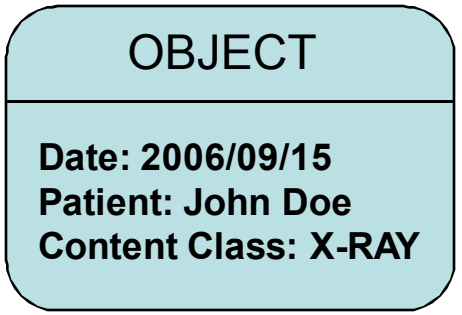
- What is Fixed Content?
- What is MetaData?
- What is SNIA doing?
- What is XAM?
- XAM API
- XAM SDK

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



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

Nutritional Facts
Serving Size 1/2 cup (130g)
Servings per container about 3

Amount per serving	
Calories 130	Fat Cal 5
	% Daily Value
Total Fat 0.5g	0%
Saturated Fat 0g	0%
Cholesterol 0mg	0%
Sodium 260mg	11%
Total Carbohydrates 22g	7%
Dietary Fiber 5g	22%
Sugars 0g	
Protein 10g	20%
Vitamin A 0%	Vitamin C 0%
Calcium 4%	Iron 10%

* Percent Daily Values are based on a 2,000 calorie diet

Extended labeling for LOB uses

Use of MetaData Standards

Writes content and annotates it with metadata, in this case: *to, from, roles, subject* and *number of attachments*

Email Service

Email object stored by XAM SDK

Metadata accompanies content

Content

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 }

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.

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.

SNIA XAM Standard Activities

Your partners and competitors are already participating

Don't be left out!

- **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 a 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)

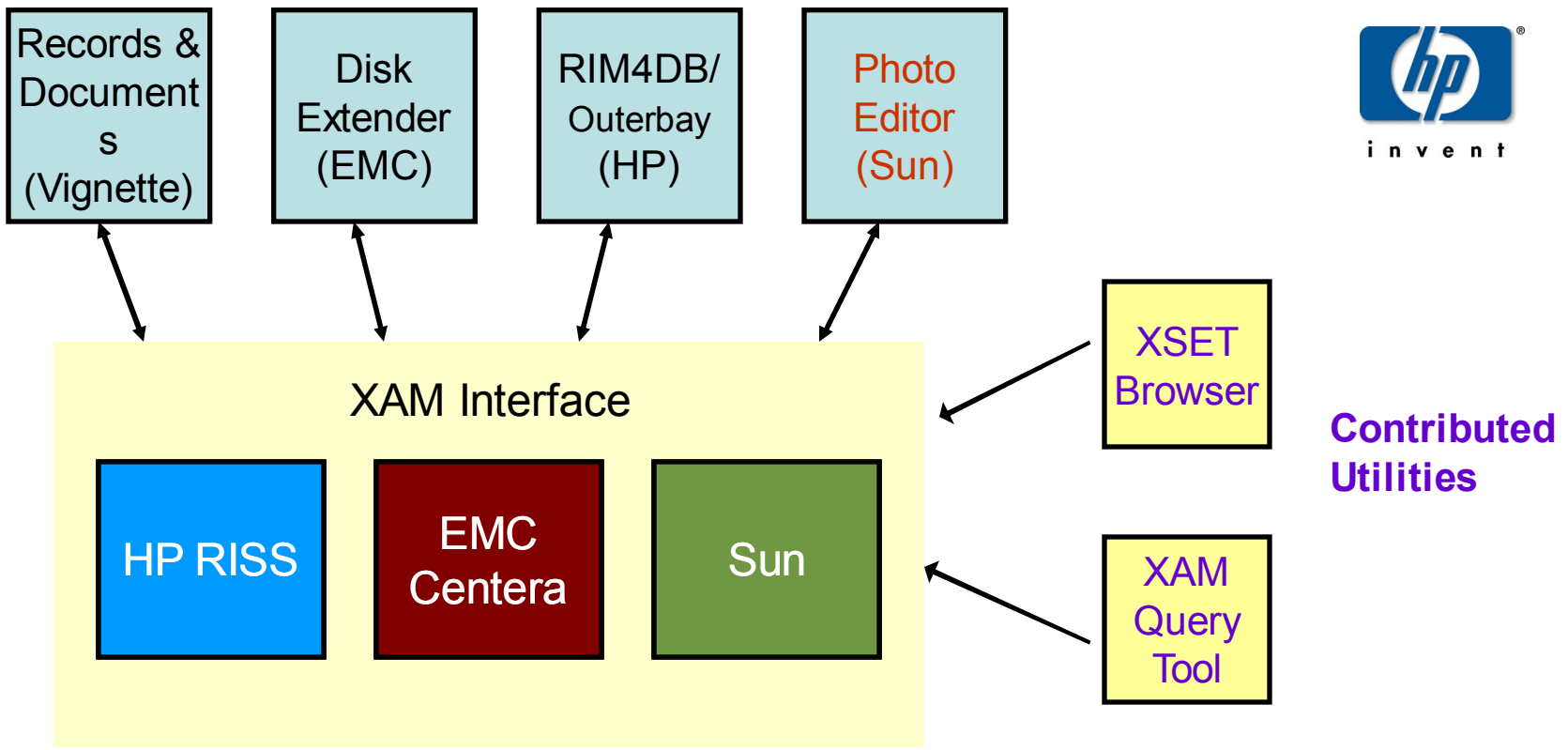
As seen at SNW Fall

First Multi-Vendor demonstration based on XAM

EMC² where information lives®

Sun microsystems

hp invent



What is the XAM Initiative?

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

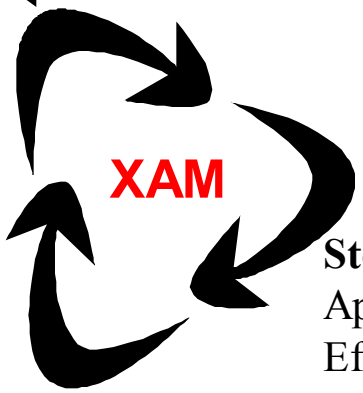
What is XAM?

- 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

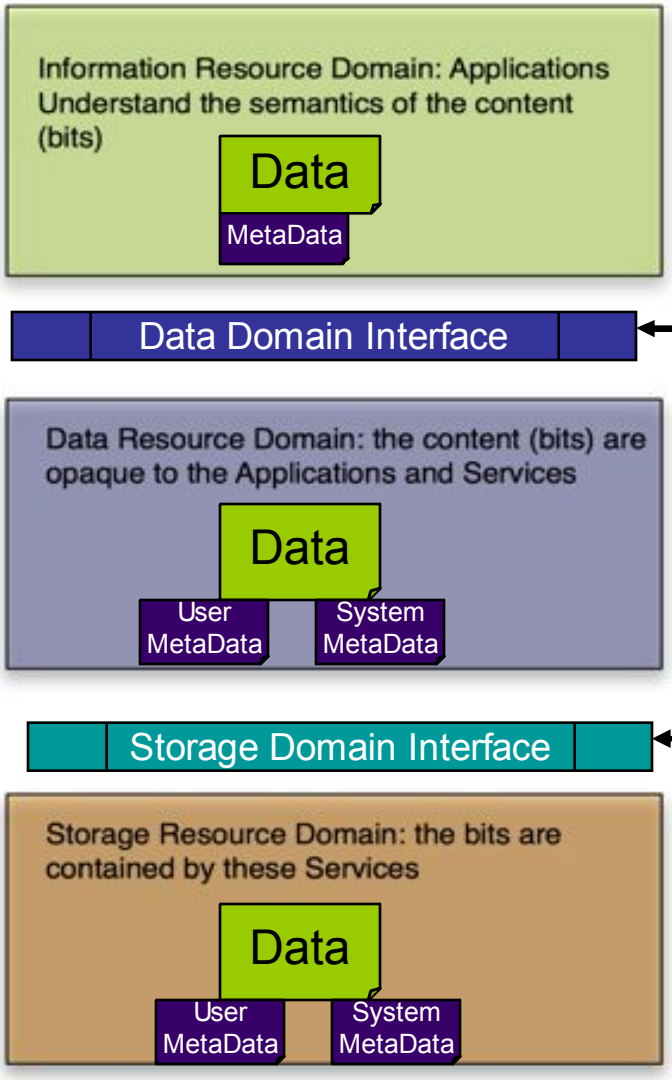
End Users want:
Choices between Application Vendors
Choices between Storage Vendors
Easy migration between vendors/technology
Compliance, Scalability, Performance, $/GB, TCO



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"

XAM is a Data Domain Interface



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 it's 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

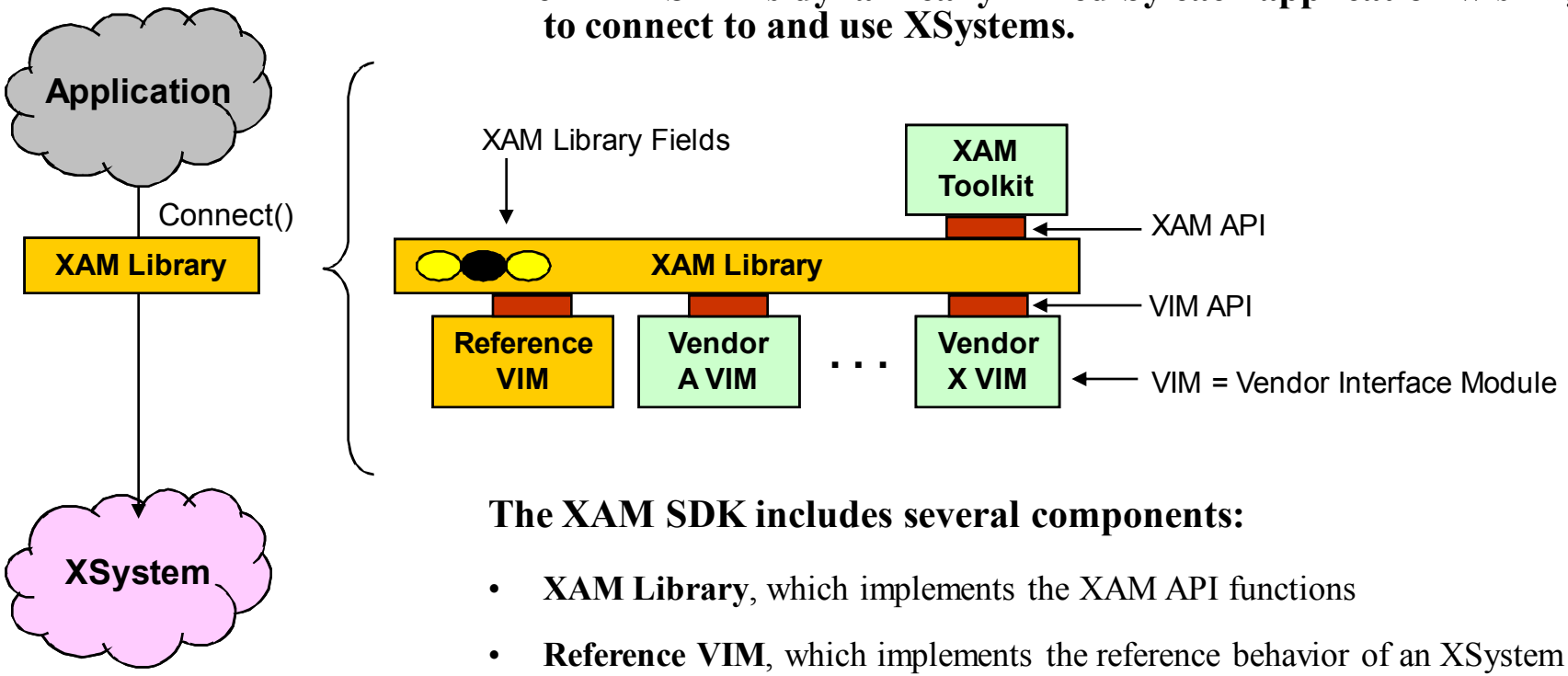
XAM System MetaData

- 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.

XAM SDK – Quick Review

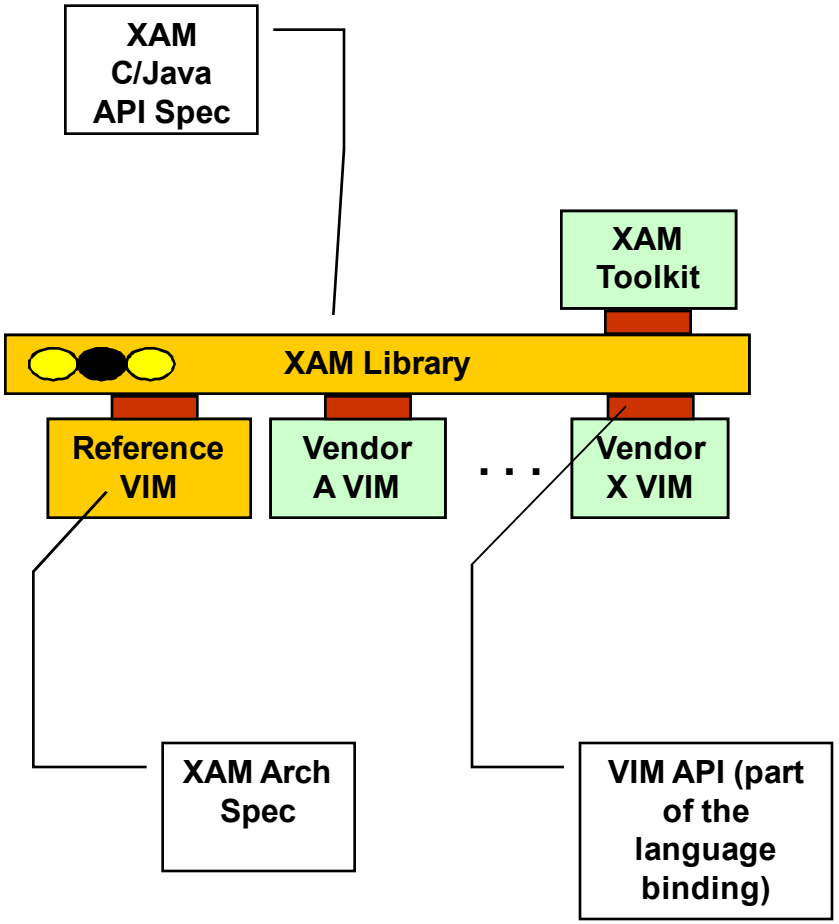


The XAM SDK is dynamically linked by each application wishing to connect to and use XSystems.

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

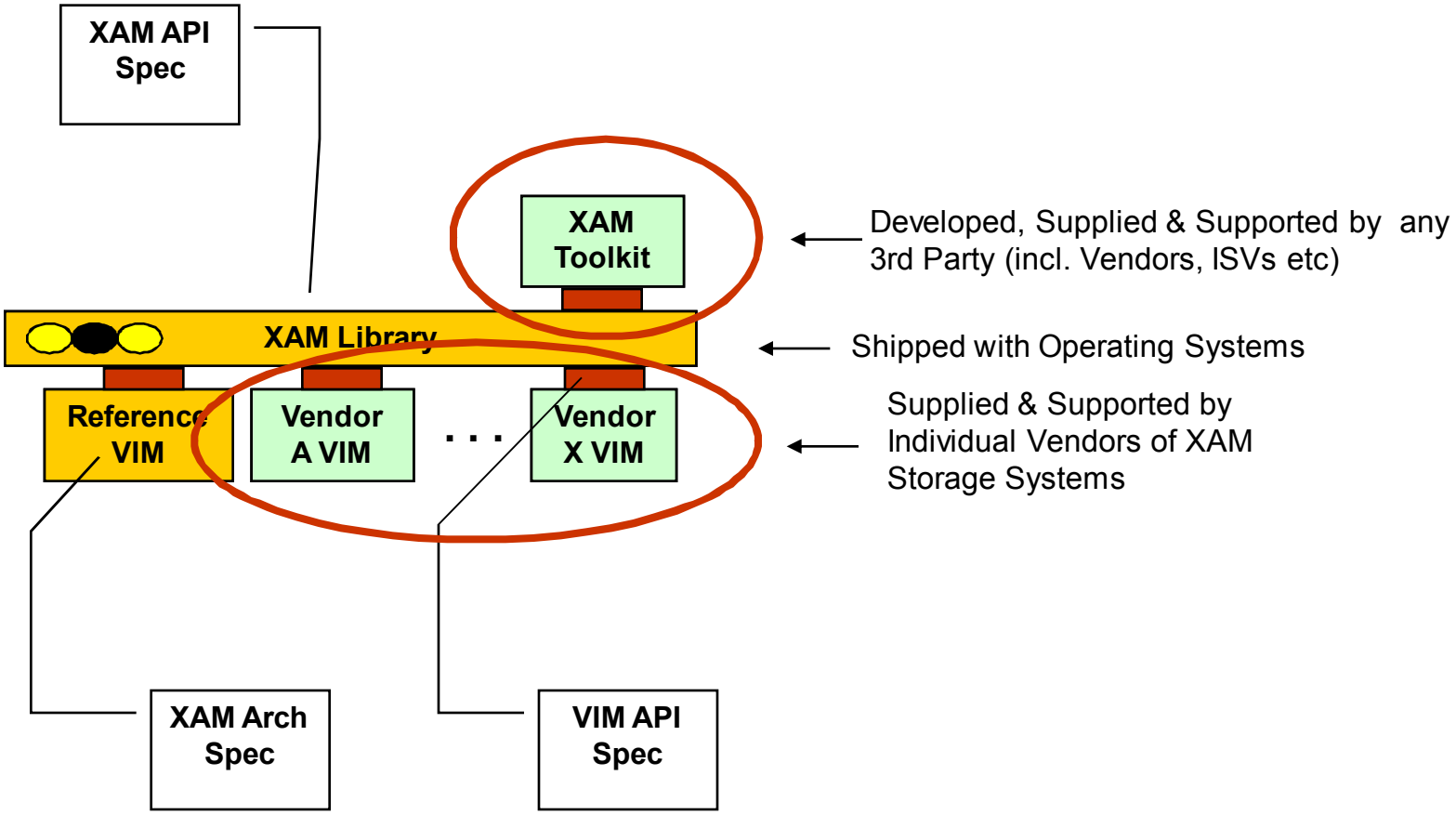
Proliferation Questions



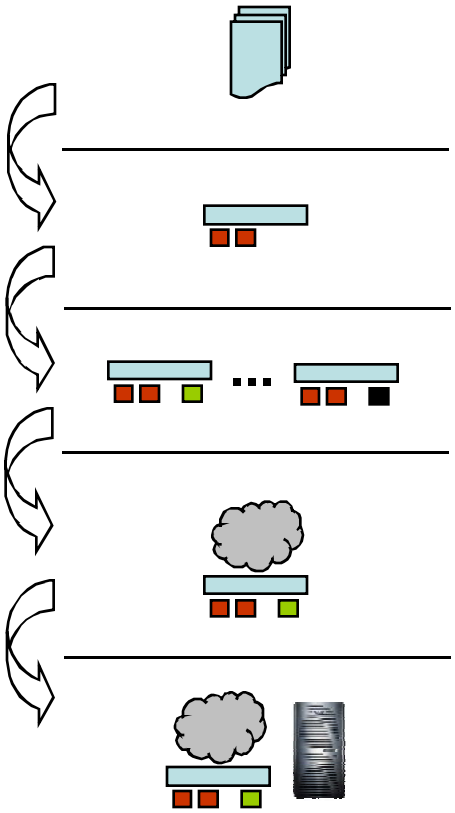
SDK Ecosystem

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

The Low Hanging Fruit



XAM SDK Proliferation - Summary



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

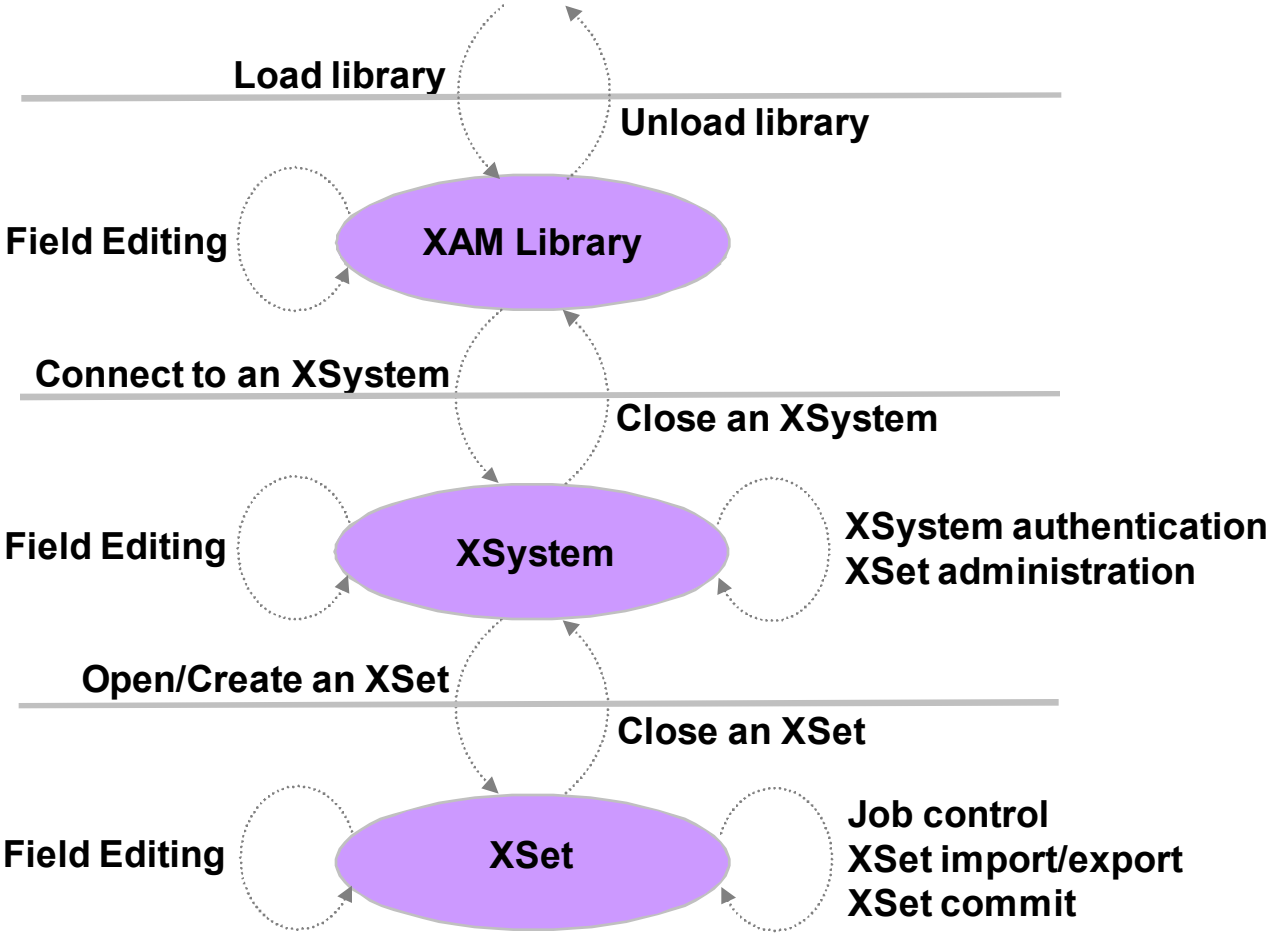
- XAM will consist of a set of components.
  - The 'topmost' library will contain the public XAM interfaces; thus, only the topmost library will be directly referenced by applications that wish to integrate with the XAM API.
  - Extension libraries may also be provided which implement higher levels of 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

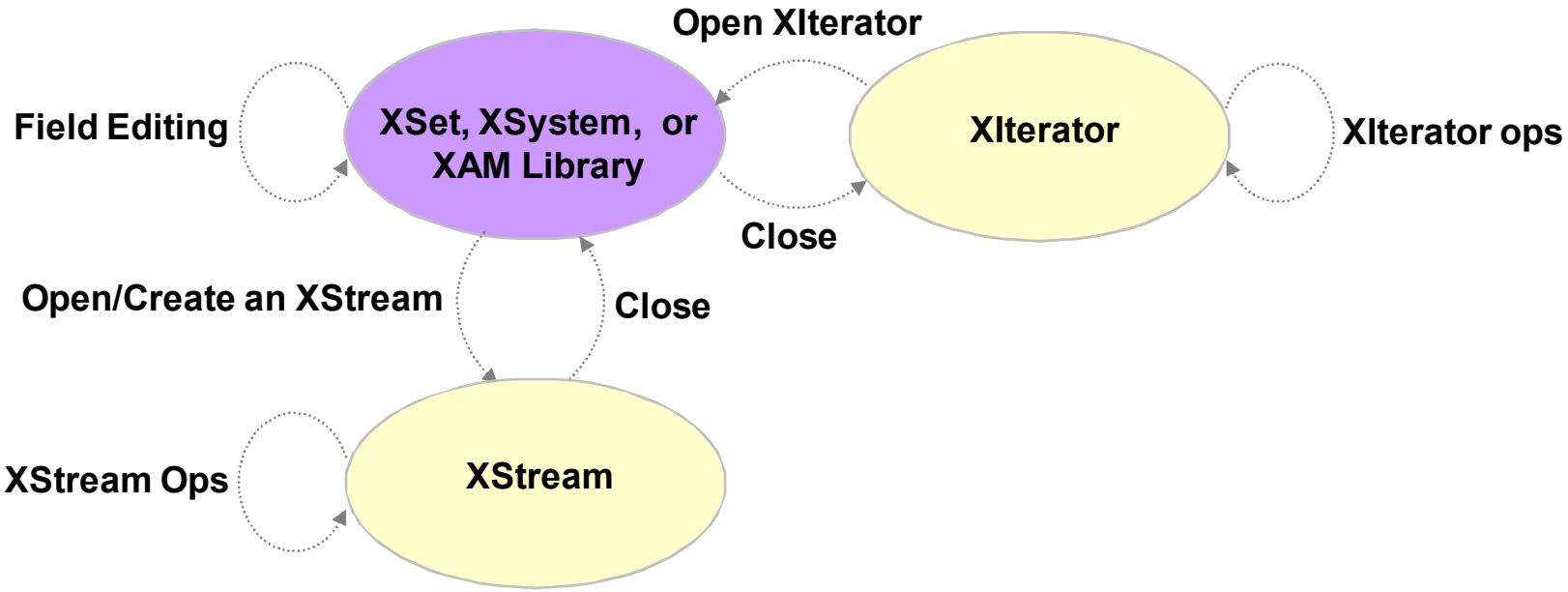
- Provide a generic interface for applications
  - XAM API methods have the same syntax and semantics without regard to the underlying storage. No methods exist that "lock-in" an application to a specific storage system; in fact, the systems themselves should be semantically indistinguishable when viewed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

Organization of the XAM Architecture

- XAM Interface semantics are organized around objects
  - Primary objects
    - XAM Library
    - XSystem
    - XSet
  - Secondary objects
    - XStream
    - XIterator

XAM Primary Objects





Organization of the XAM Architecture

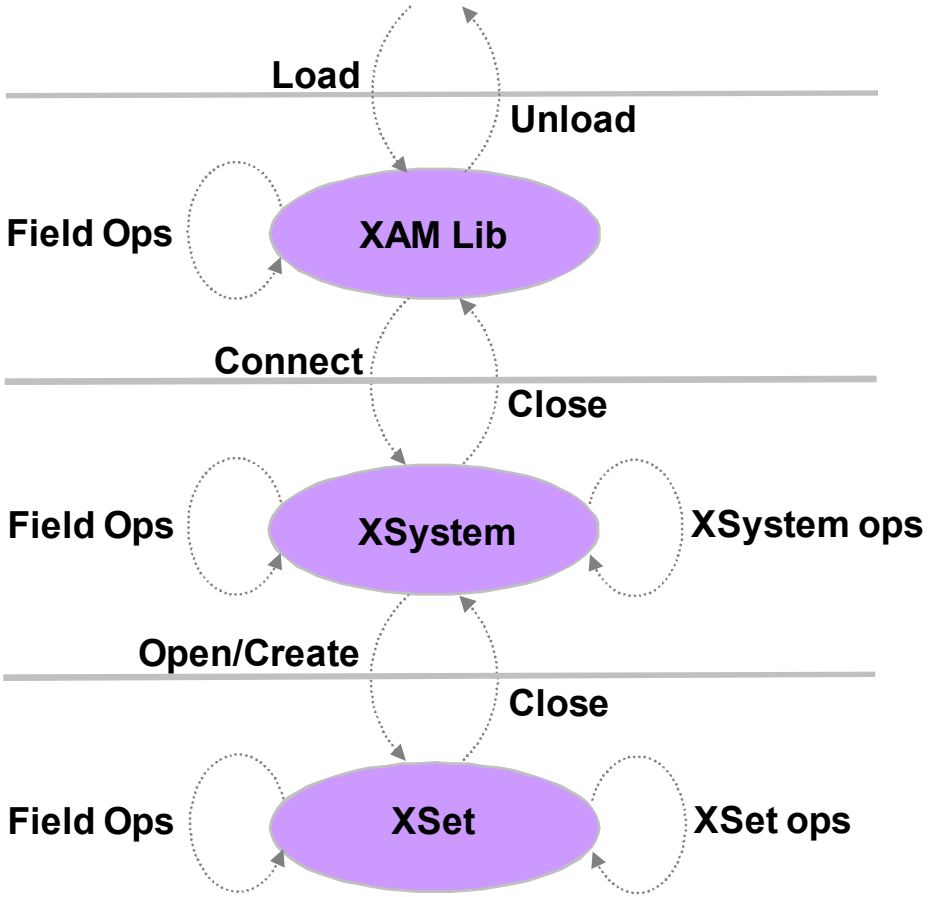
- XAM Interface semantics are organized around objects
  - Primary objects
    - XAM Library
    - XSystem
    - XSet
  - Secondary objects
    - XStream
    - XIterator

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

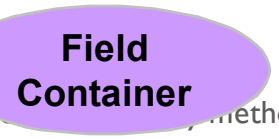


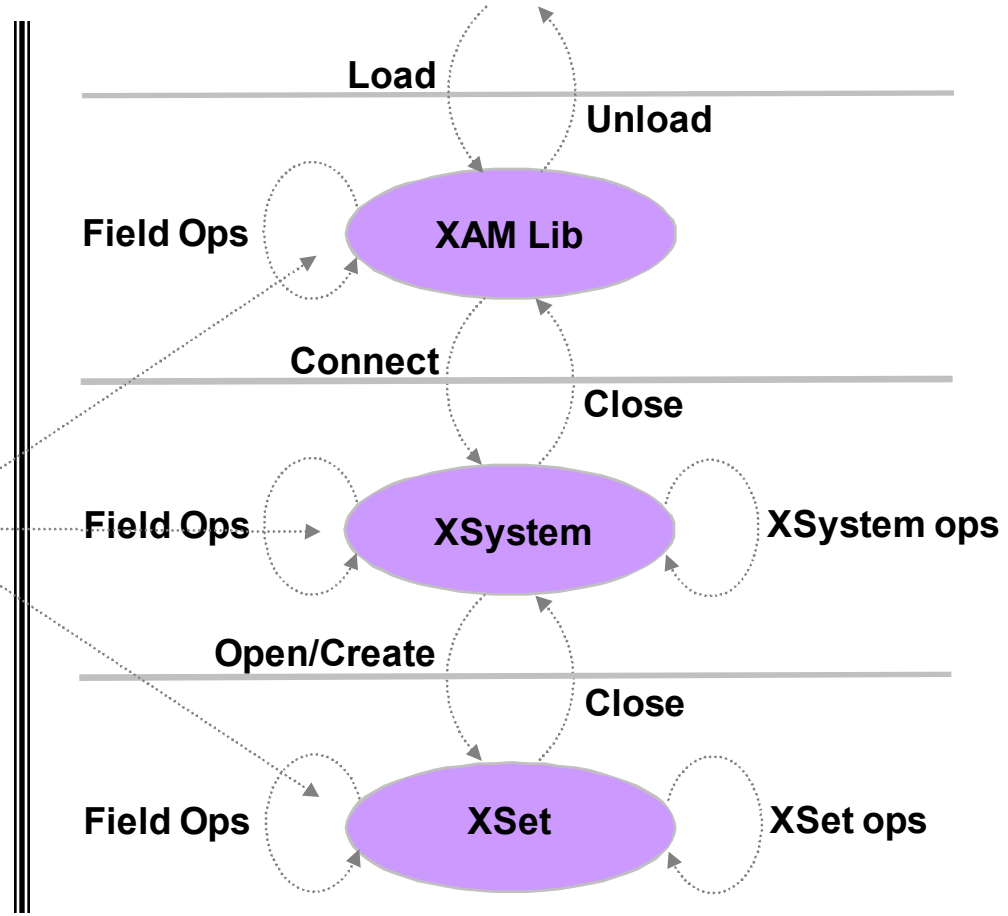
Elements of Primary Objects

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**

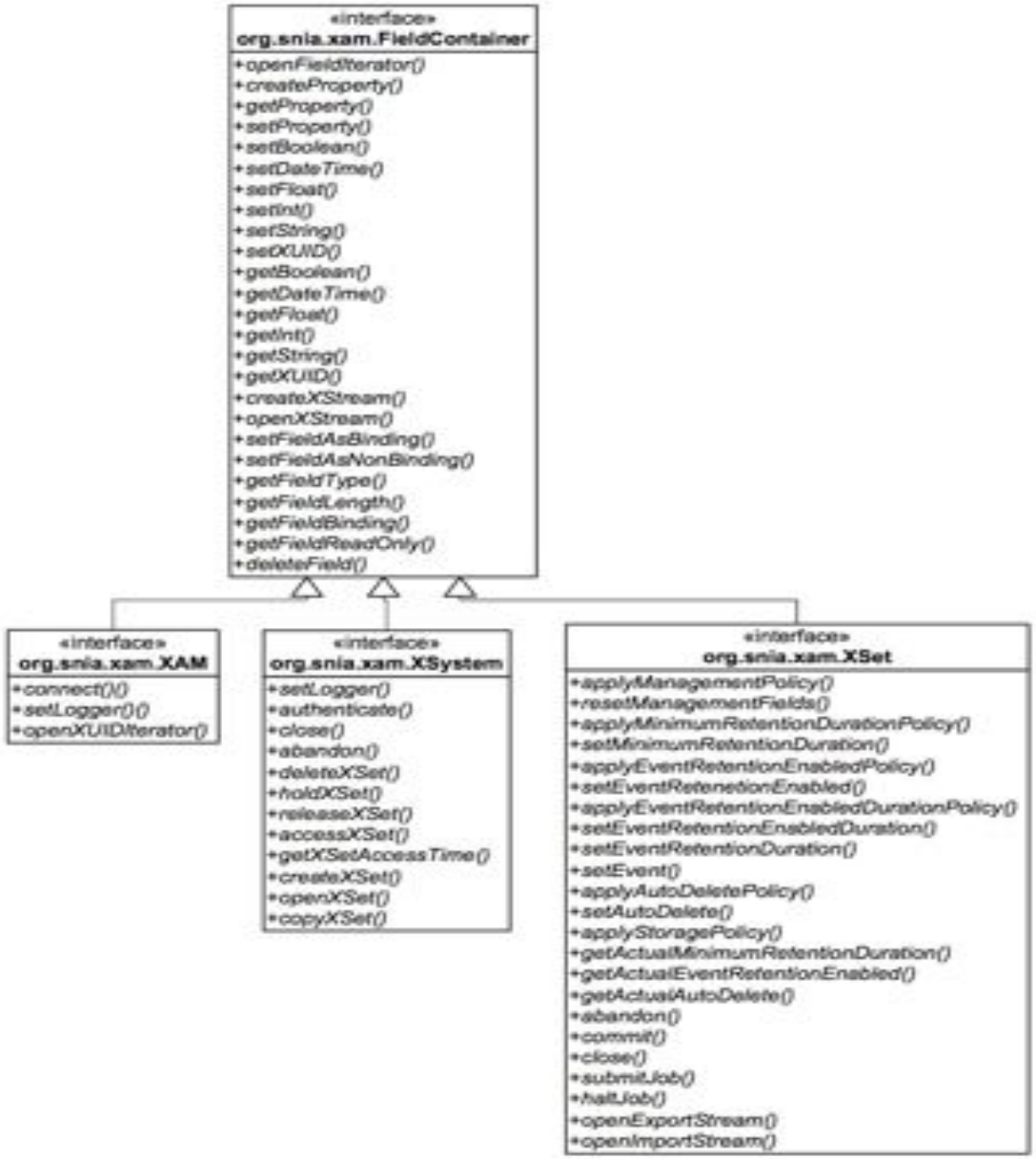
XSystem object
- No constructor is available
  - This must be available by calling a factory method on the XAM Library object
- **Contains fields**

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





UML of the Primary Object



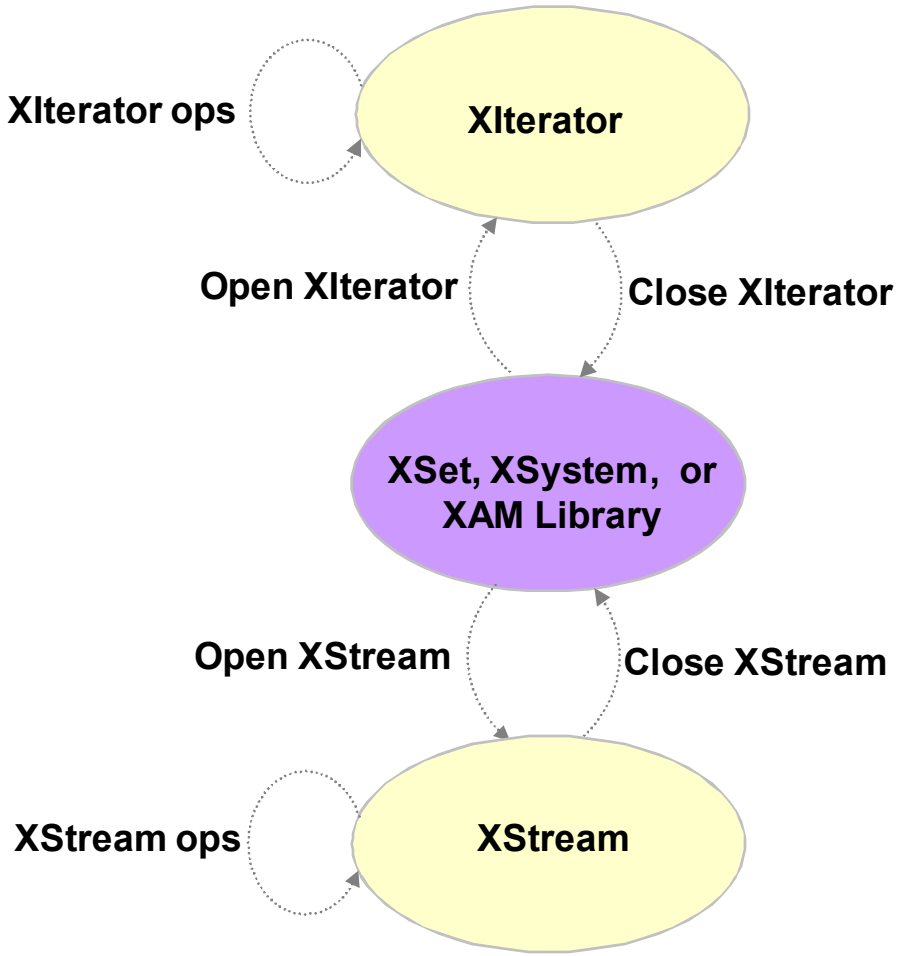
Elements of Secondary Objects

XIterator object

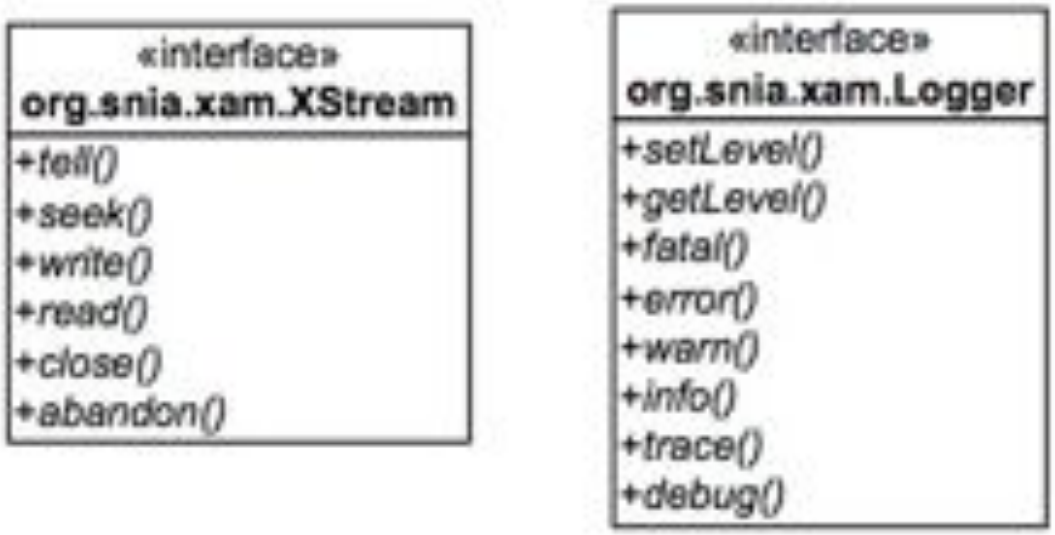
- No constructor is available
  - This must be available by calling a factory method on a Primary object (implements Field Container)

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



XAM Application software stack

- 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

Where To Go

- SNIA XAM Home
  - http://www.snia.org/xam
- SNIA FCAS TWG
  (XAM Technical Work Group)
  - http://www.snia.org/apps/org/workgroup/fcastwg/
- SNIA XAM SDK TWG
  (XAM SDK Technical Work Group)
  - http://www.snia.org/apps/org/workgroup/xamsdktwg/

Thank you!

Technical Council