



Education

eXtensible Access Method  
Software Development Kit  
**(XAM SDK)**

Mark A Carlson

SNIA Technical Council, Sun Microsystems, Inc.

# SNIA Legal Notice

- 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 and the SNIA Technical Council.

- Why is SNIA Doing Software?
- What are the rules?
- How are they approved?
- The XAM SDK TWG
- Other possible TWGs

# Problem: Adoption and Maturation

- SNIA produces Standards in new areas
  - ◆ New APIs, Protocols, etc, for Storage, Data, Information Domains
- Unknown maturity before vendors implement
  - ◆ Each vendor must interpret the specification and make implementation choices
  - ◆ We must get multiple vendors to do this quickly
- Plug testing is late in the game
  - ◆ After vendors have already made interpretations
  - ◆ Perhaps after specification has been submitted to ANSI, etc.
- Catch-22 of adoption
  - ◆ Vendors are reluctant to implement an immature specification
  - ◆ They look for existing implementations before starting their own

- By creating a reference implementation to complement the specification:
  - ◆ Implementation choices can be made by multiple vendors
  - ◆ Loose areas of the specification can be discovered in the process
  - ◆ Vendors can pick up a working implementation to get started on their own
  - ◆ Plug testing is enhanced by having a canonical implementation to test against

# What is new?

- ▶ **SNIA Technical Work Groups (TWGs) may add Software Work Items to their Charter**
  - ◆ For example: reference implementation, SDK, tools, and associated documentation
- ▶ **SNIA Software is developed under the SNIA IP Policy**
  - ◆ *SNIA Software* defined similar to *SNIA Architecture*
- ▶ **SNIA Software may be Open Sourced**
  - ◆ The license is proposed as part of the Charter/Work Item(s)

# What are the rules?

- ◆ SNIA Software is worked on by member companies who have opted-in to the work of the TWG
  - ◆ Implies obligations to disclose and license IP
  - ◆ Also implies obligations under the inbound and outbound license
    - Marking of code contributions, etc.
- ◆ Other SNIA members only see the code when the TWG publishes
- ◆ Non-SNIA members only see the code after the members have approved

# Process for Approval

- Charter still needs to be approved by the Technical Council
  - ◆ Including the proposed license(s)
- License(s) must be approved by the SNIA Board
  - ◆ Must include a license that is compatible with *commercial software distribution*
- Release of the software outside the TWG needs TC approval as well (just like specifications)
- SNIA Software still must pass a membership vote for public publication
  - ◆ Similar to SNIA Architecture
  - ◆ Implies adoption by the organization
  - ◆ Only Board can forward to membership for a vote

# Software Work in SNIA

## ➤ XAM SDK TWG

- ◆ Develop a Software Development Kit (SDK) for the eXtensible Access Method (XAM) specification
- ◆ Tools and related documentation
- ◆ BSD (3 clause) license

## ➤ Other planned TWGs

- ◆ NDMP
- ◆ MF TWG reference implementation
- ◆ ...

## Overview

## eXtensible Access Method (XAM) is

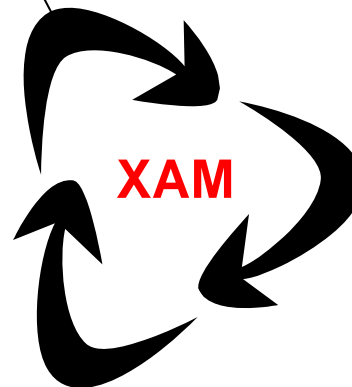
- a SNIA Initiative
- driven by the “Fixed Content Aware Storage” TWG
- to define a standard interface (i.e. API)
- between “Consumers” (application and management software)
- and “Providers” (storage systems)
- of *Fixed Content* storage services

# Why XAM?

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

## Applications 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 of “records”



## End Users want:

Choices between Application Vendors  
Choices between Storage Vendors  
Easy migration between vendors/technology  
Compliance, Scalability, Performance, \$/GB, TCO

## Storage Vendors want:

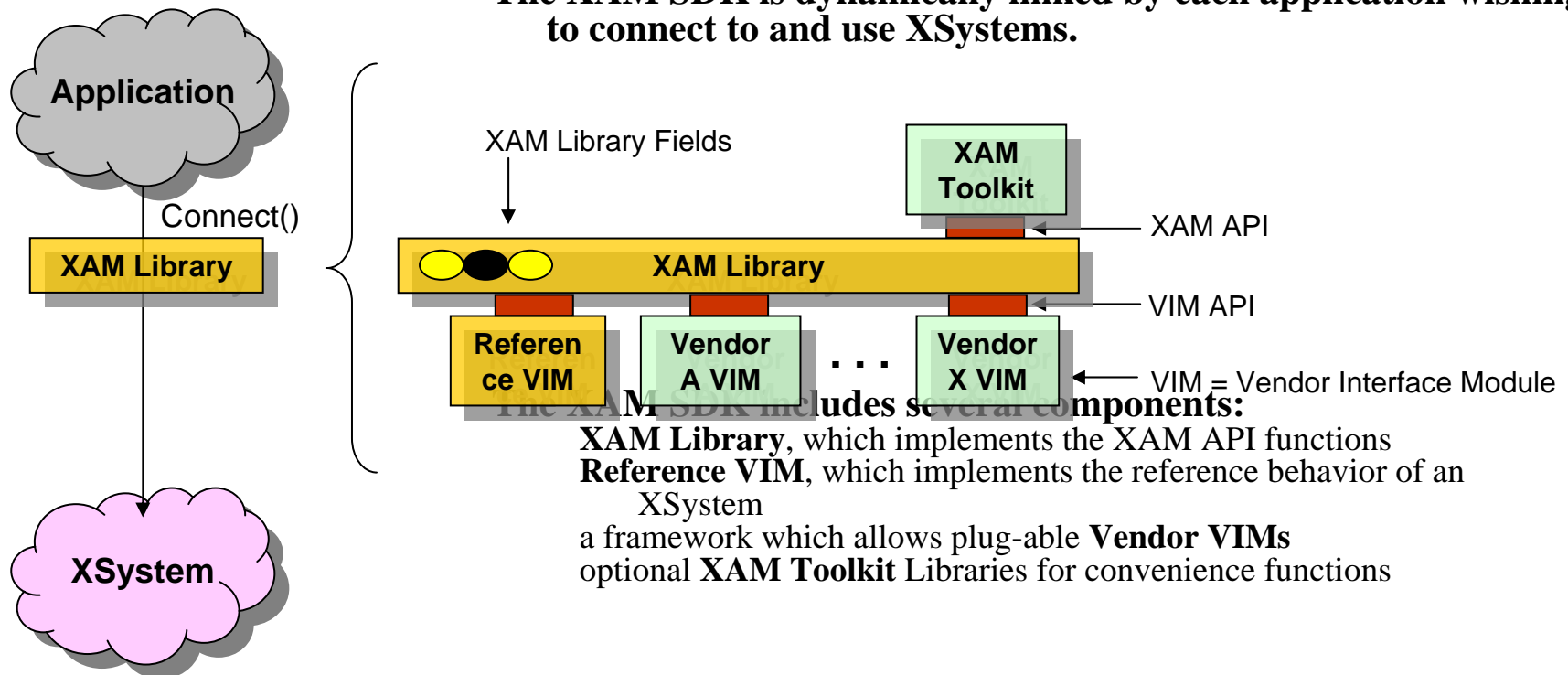
Application Support for their Products  
Efficiently Store Application Data and Metadata  
Integrate Basic Storage Management Capabilities  
Manage billions if not trillions of “records”

# XAM SDK TWG Charter

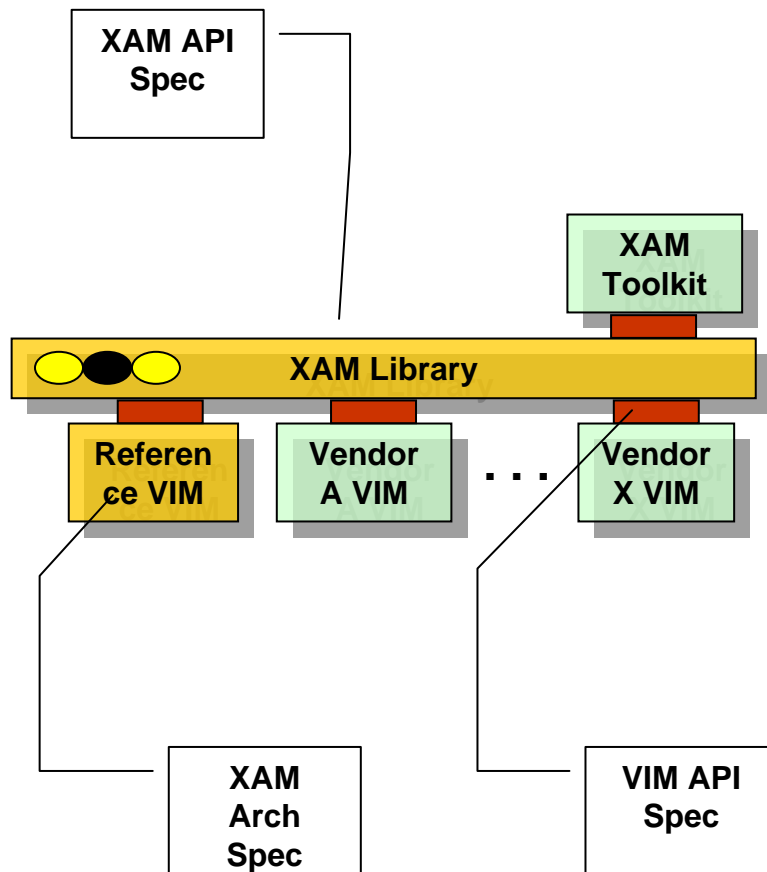
- Develop SNIA Software that implements the XAM Library.
- Develop SNIA Software that implements a Reference Vendor Implementation Module (VIM) on top of an existing filesystem.
- Develop sample XAM Client Applications as SNIA Software to provide simple unit tests for portions of the XAM Specification(s).
- 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.



# Proliferation Questions

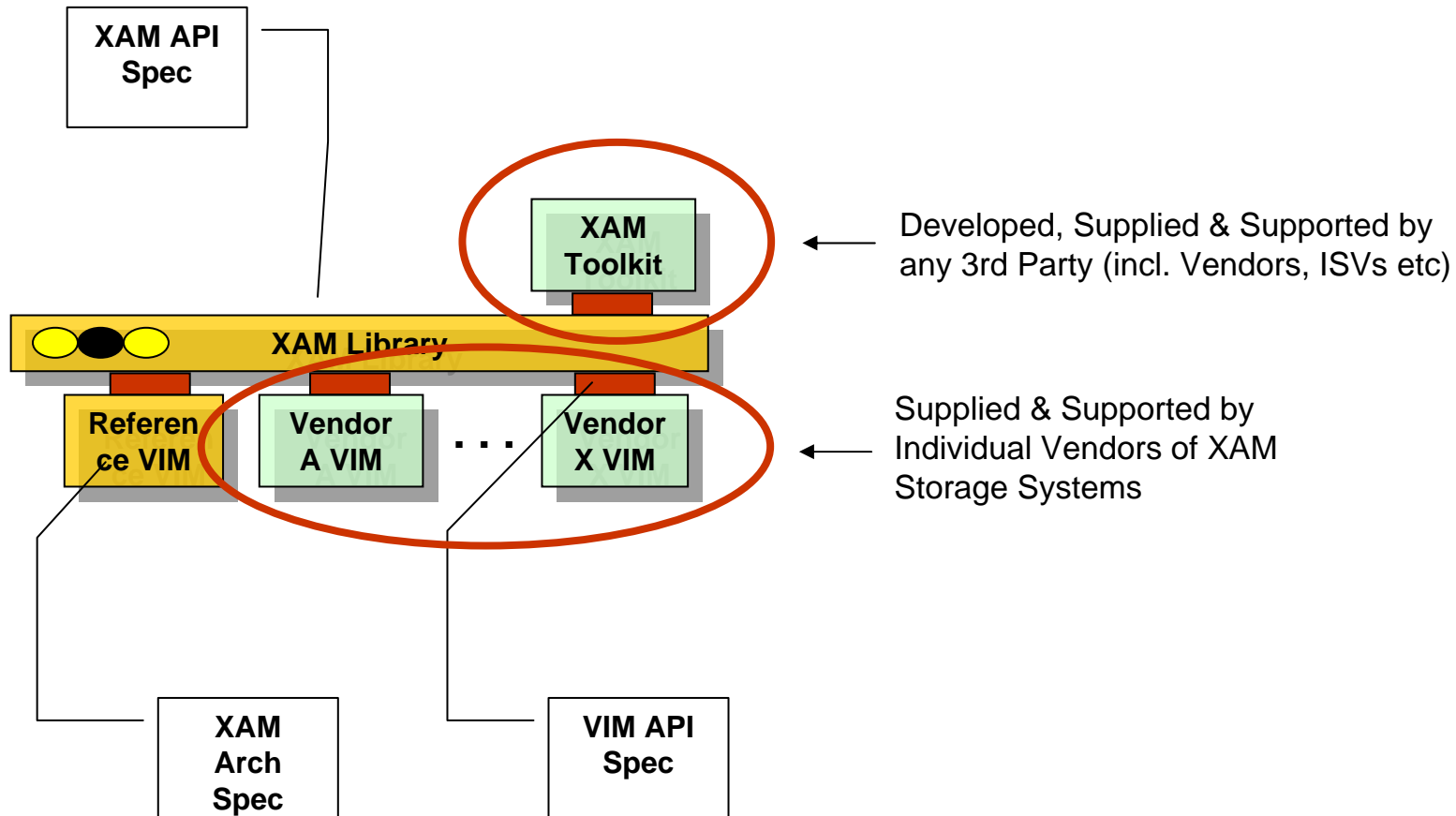


## Who Does This Work?



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

# The Low Hanging Fruit





**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 spec for**

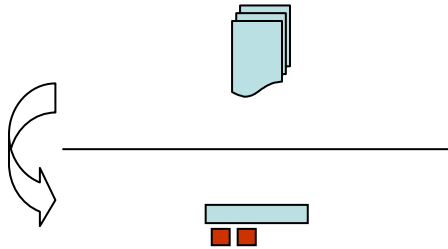
XAM Architecture

XAM API

VIM 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**

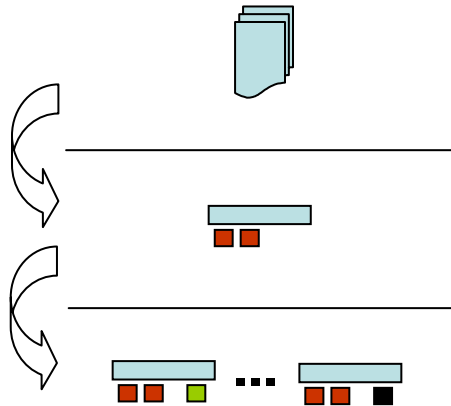
**‘Gold’ XAM SDK (*XAM Library, Reference VIM*) developed and maintained by SNIA’s “XAM Software TWG” member companies**

Possibly for a single language binding only (C/C++)  
Possibly for a single OS platform only

**Beta-quality code of SNIA’s XAM SDK (‘Gold’ Distribution) released on regular schedule to SNIA Member Companies under BSD-type license**

**SNIA’s “XAM SDK TWG” provides the last tier of support for the XAM SDK ‘Gold’ distribution**

# XAM SDK Proliferation – SNIA's Member Companies



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

**Member Companies 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

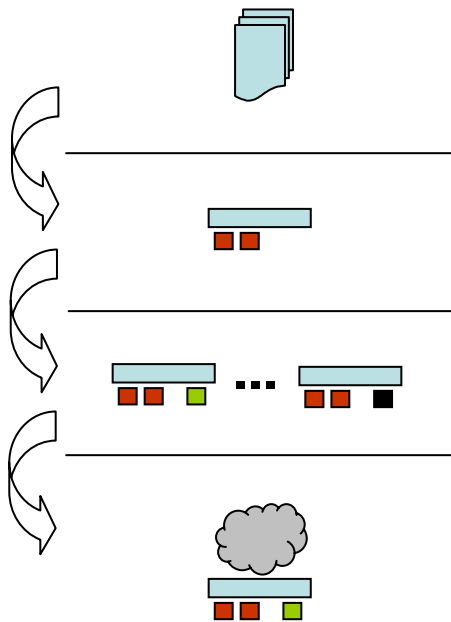
**Member Companies responsible for interoperability testing of their XAM SDK Derivative** against other vendors' released VIMs

**Member Companies responsible for porting and new features back to SNIA's XAM SDK Derivative** into the next generation

**Member Companies responsible for supporting their XAM SDK Derivatives** to the SNIA InterOp Lab, or some authorized 3<sup>rd</sup> party

**It may make sense to centralize cross-certification (of various vendors' XAM SDK Derivatives with various vendors' VIMs) back to the SNIA InterOp Lab, or some authorized 3<sup>rd</sup> party**

# XAM SDK Proliferation – Application Vendors (ISVs)



**ISVs Integrate and certify their apps with a chosen XAM SDK Derivative**

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

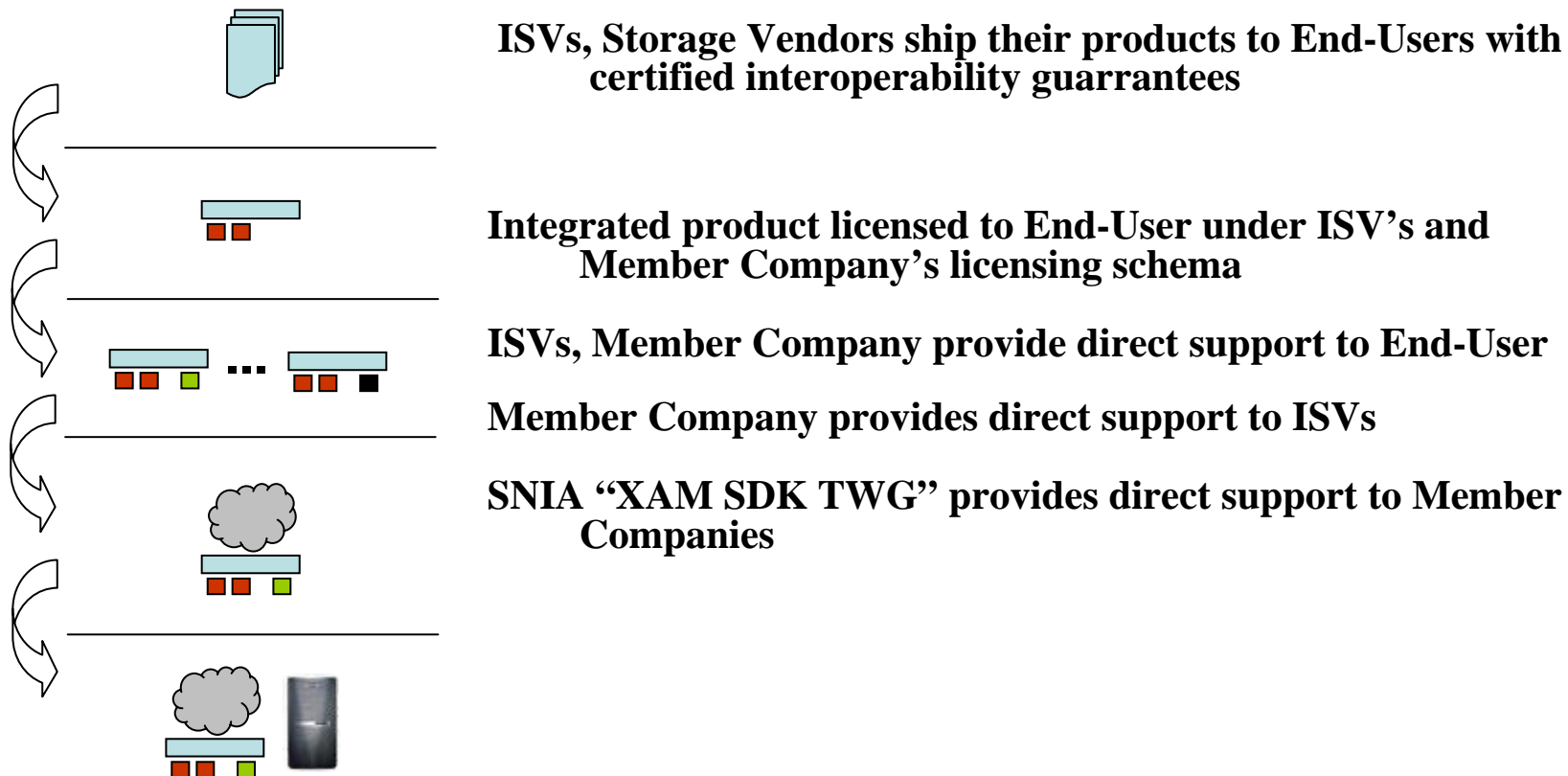
**ISVs responsible for interoperability testing and certification of their s/w applications against the chosen XAM SDK Derivative(s)**

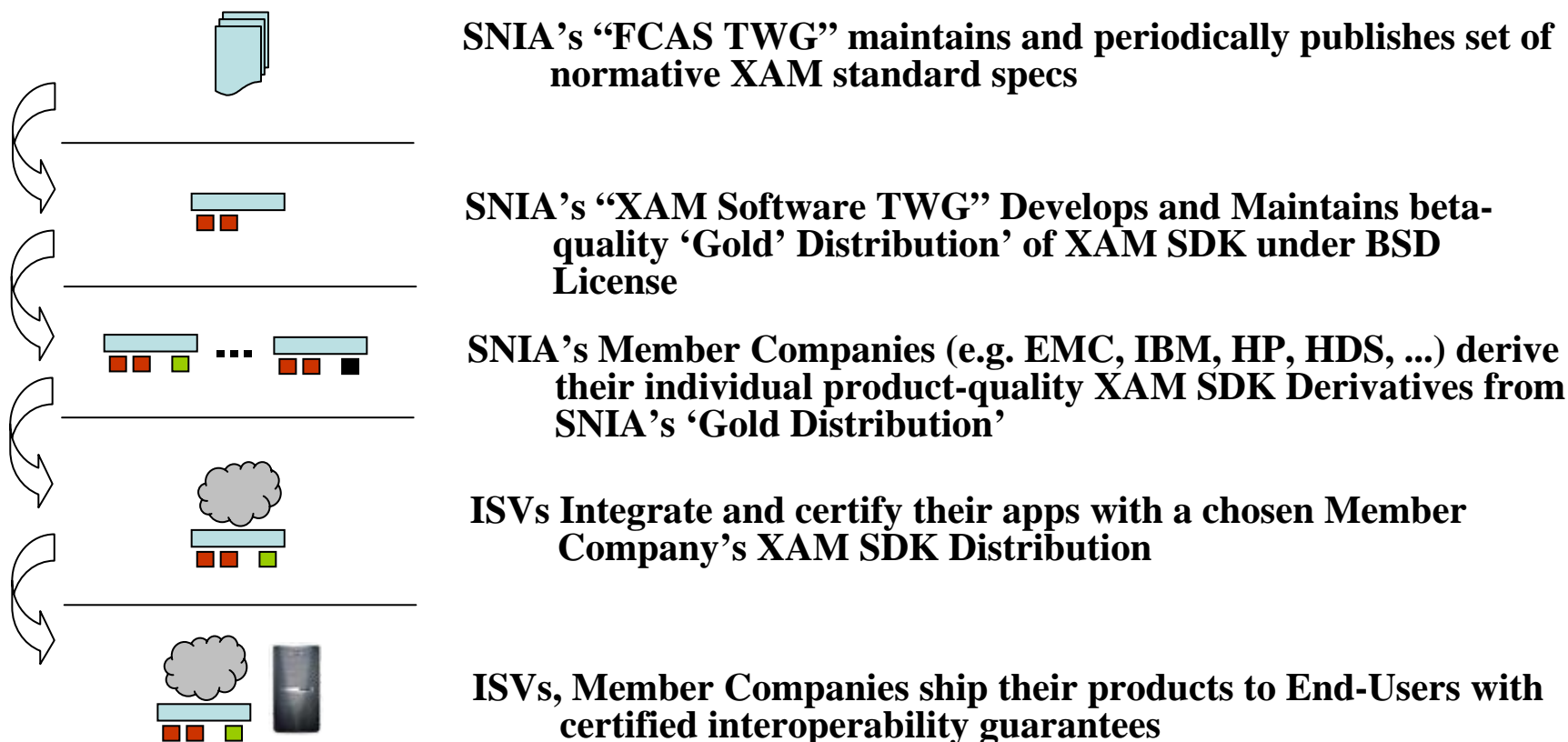
**Ongoing support the Member**

**ISVs ship their respective**

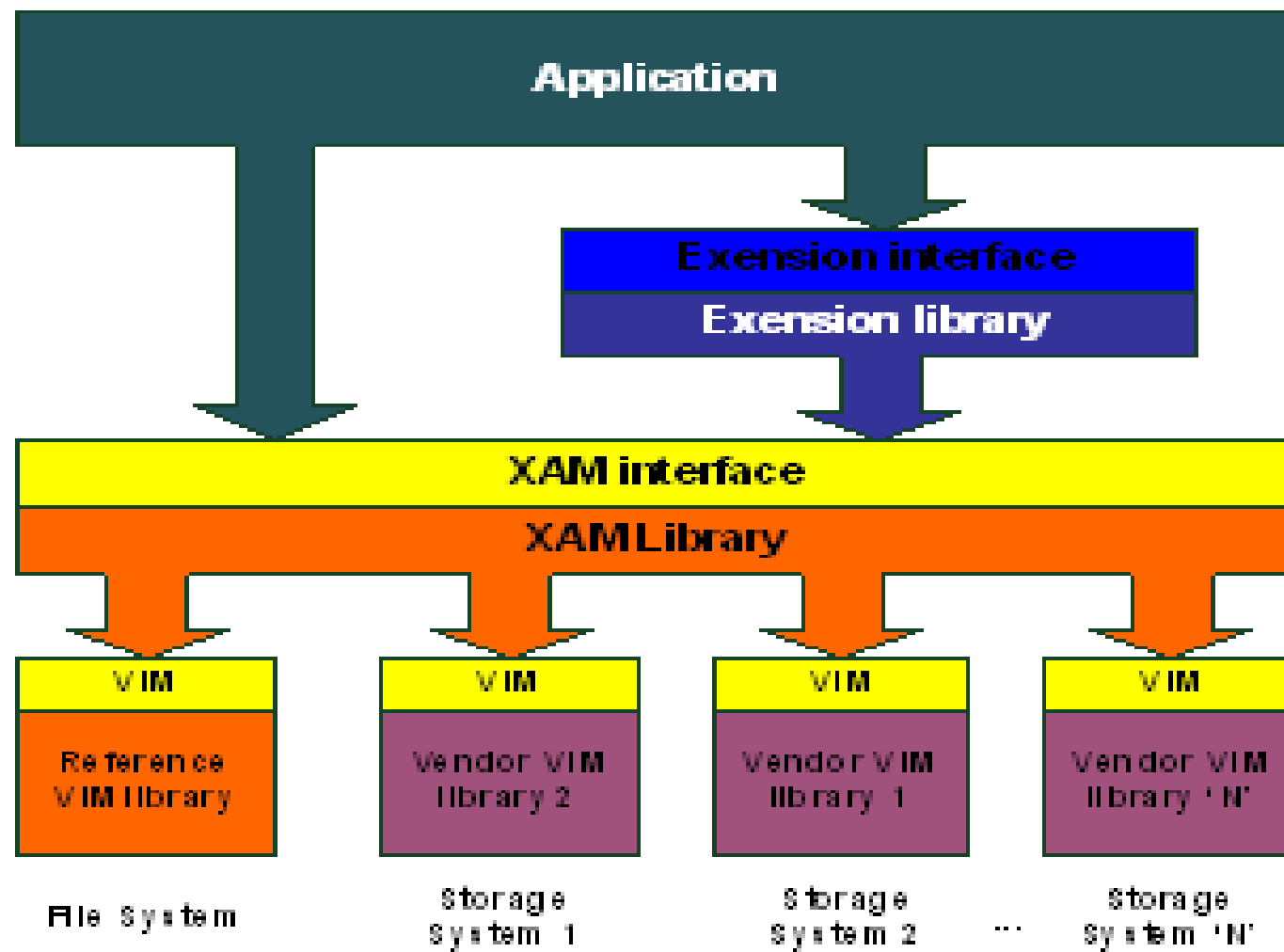
**It may make sense to outsource certification (of an ISV's application with a Member Company's XAM SDK Derivative) back to the SNIA InterOp Lab, or some authorized 3<sup>rd</sup> party**

# XAM SDK Proliferation – End-Users





# The XAM API



# 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

# The XAM SDK

## developing XAM Library software

### ➤ Provide a generic interface for applications

**Design Goals**

- ◆ 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

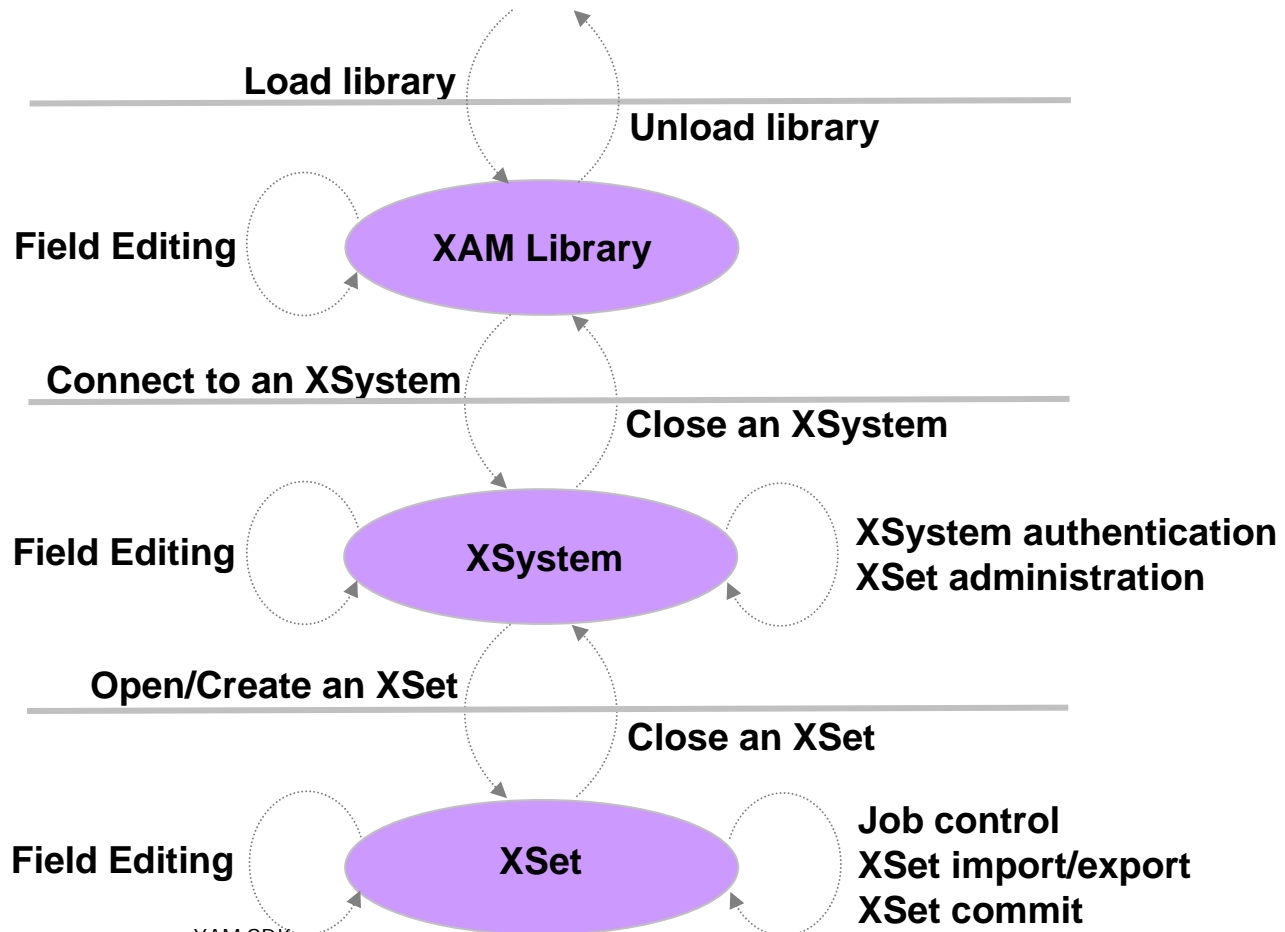
- ◆ Keep the interface simple and small (e.g., have as few API methods as possible, and keep these methods easy to use and understand), while ensuring that the methods make all forms of data manipulation possible. If functionality could have been achieved by composing other methods (in a way that sufficiently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

### ➤ Expose no implementation detail

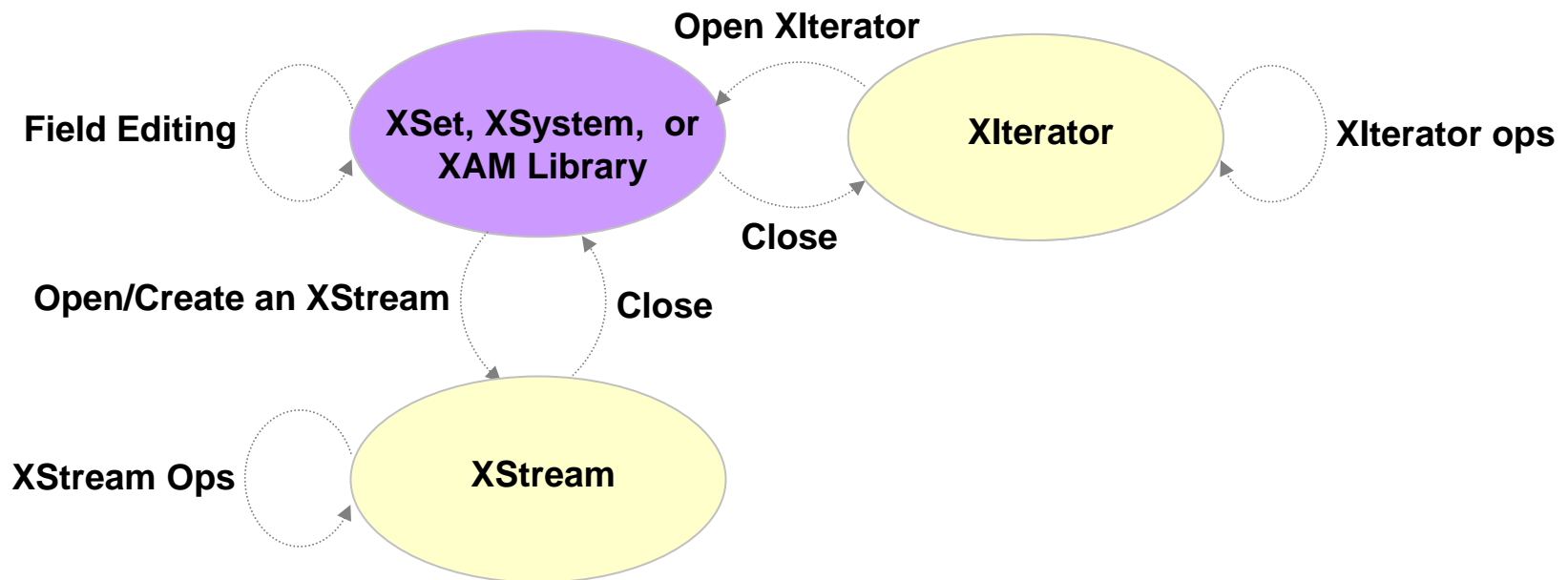
- ◆ Do not expose any internal functionality that would serve to place restrictions on storage system vendors.

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

# XAM Primary Objects



# XAM Secondary Objects (with relationship to Primary Objects)



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

**Given the object oriented semantics of XAM, it seems reasonable to implement the XAM Library using the same object model!**

# Elements of Primary Objects

## XAM Library object

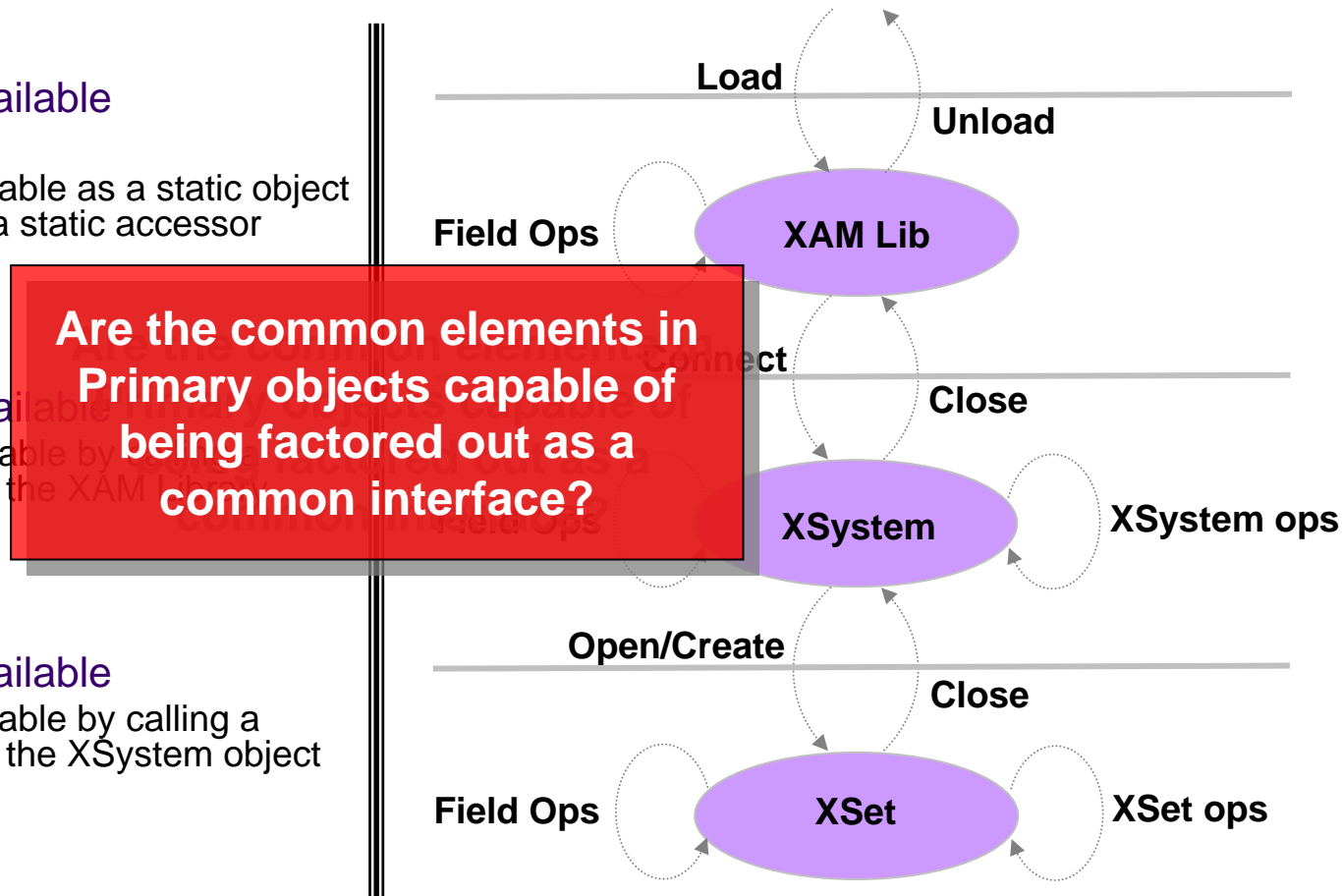
- No constructor is available
  - ◆ This is a singleton
  - ◆ This must be available as a static object or thru the use of a static accessor method

## XSystem object

- No constructor is available
  - ◆ This must be available by a factory method on the XAM Lib object

## 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

Can the lack of a constructor be factored out?

No

1. the XAM Library object is a singleton and uses an accessor method to access it
2. the XSystem and XSet objects are not singletons and use factory methods in different parent objects instead of public constructors
3. the factory methods are concrete and return different types

# 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

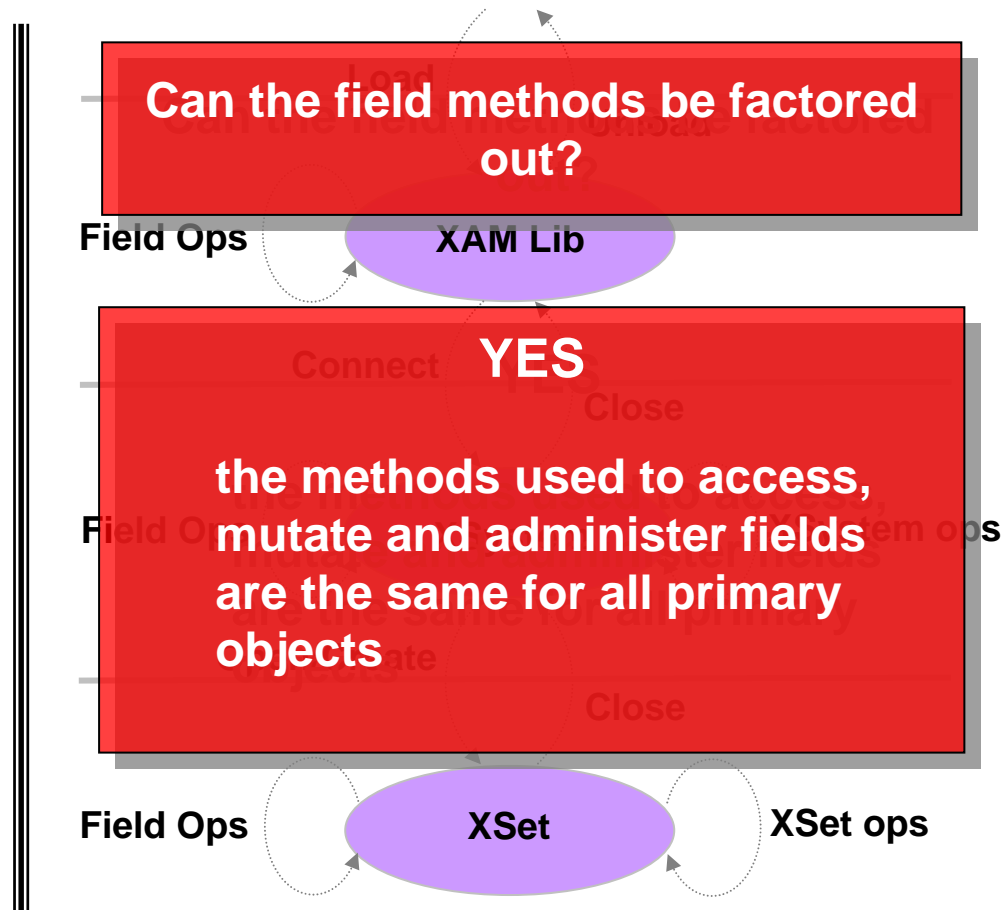
- No constructor is available
  - ◆ This must be available by calling a factory method on the XAM Library object

## ➤ 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

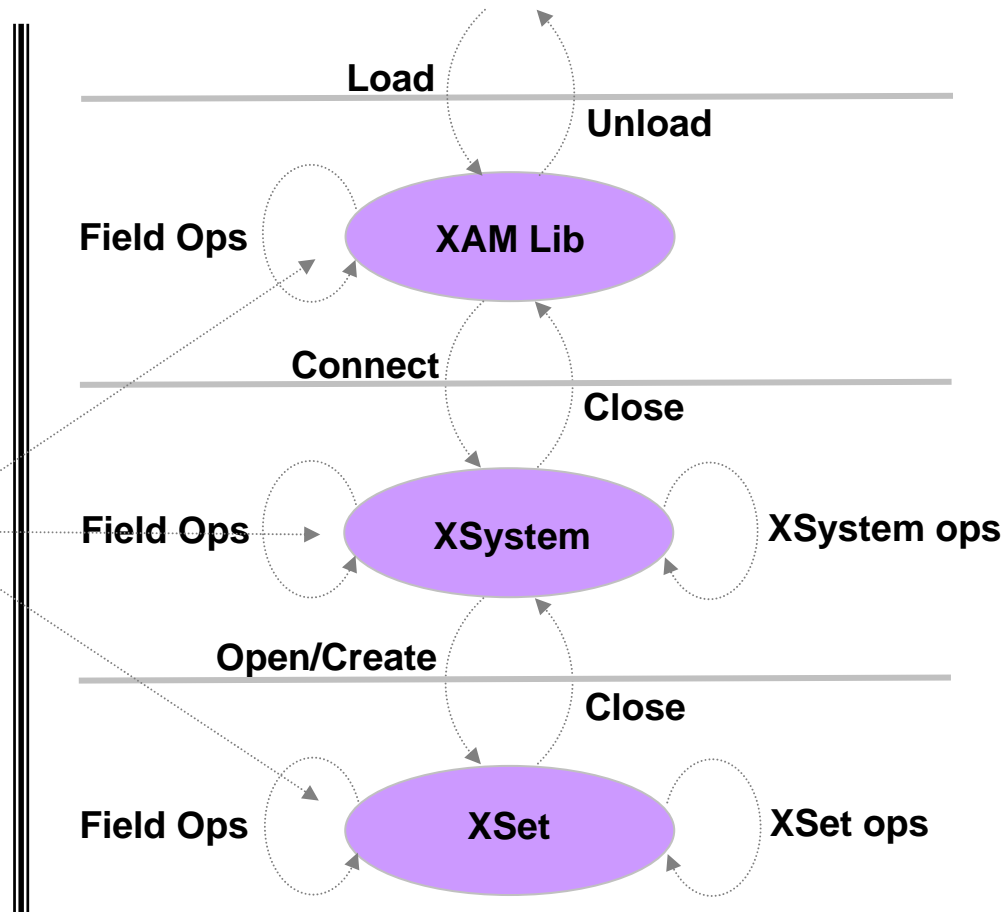
- No constructor is available
  - ◆ This must be available by calling a factory method on the XAM Lib object

## ➤ Contains fields

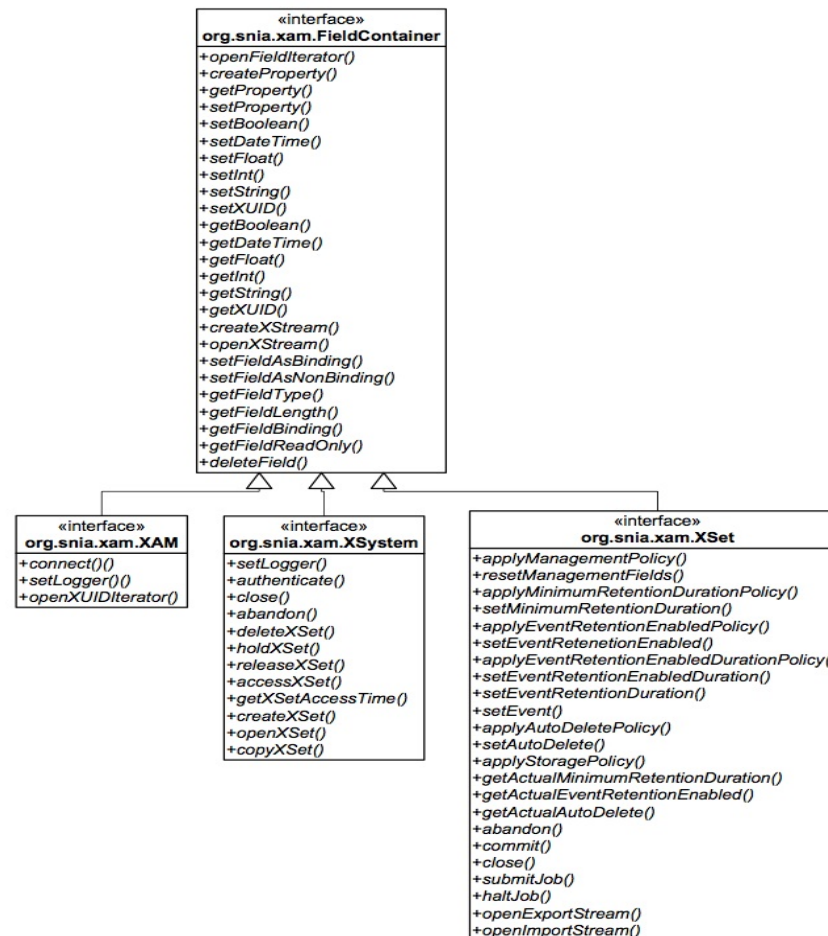
## XSet object

- 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

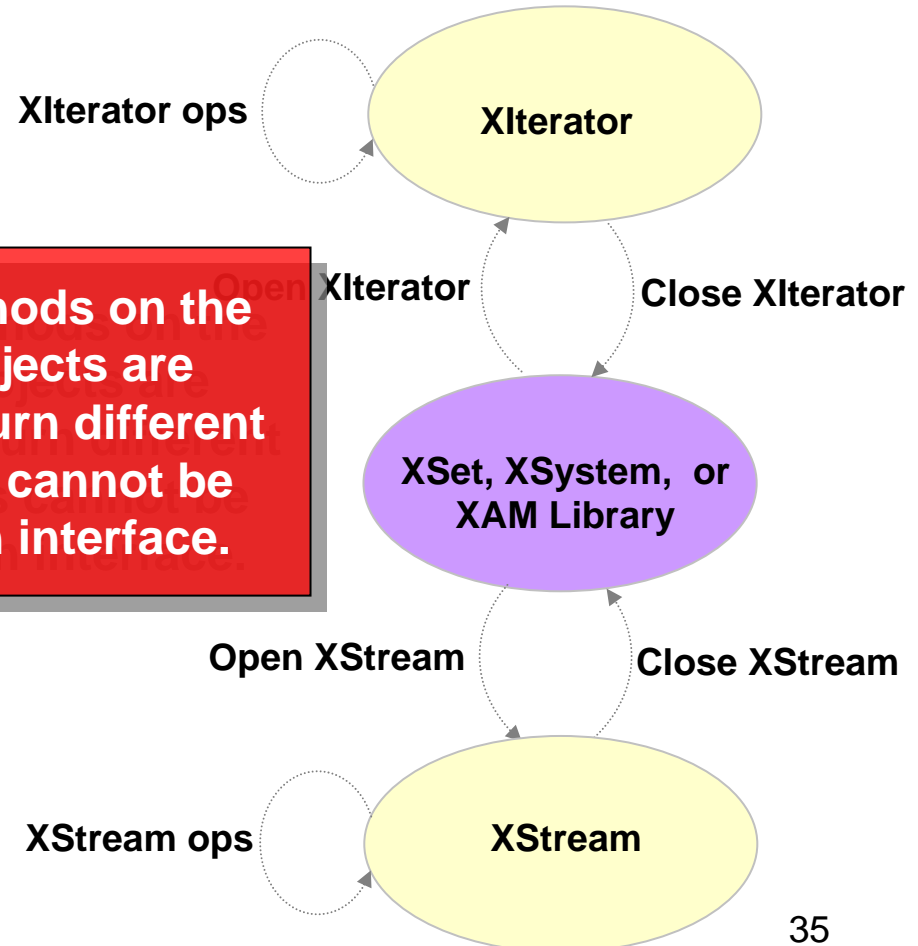
## XIterator object

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

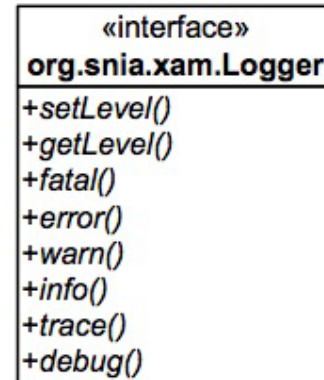
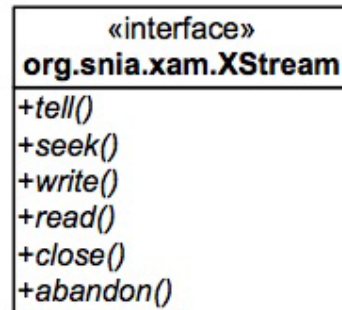
## XStream object

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

**The factory methods on the Secondary objects are concrete and return different types, and thus cannot be factored into an interface.**



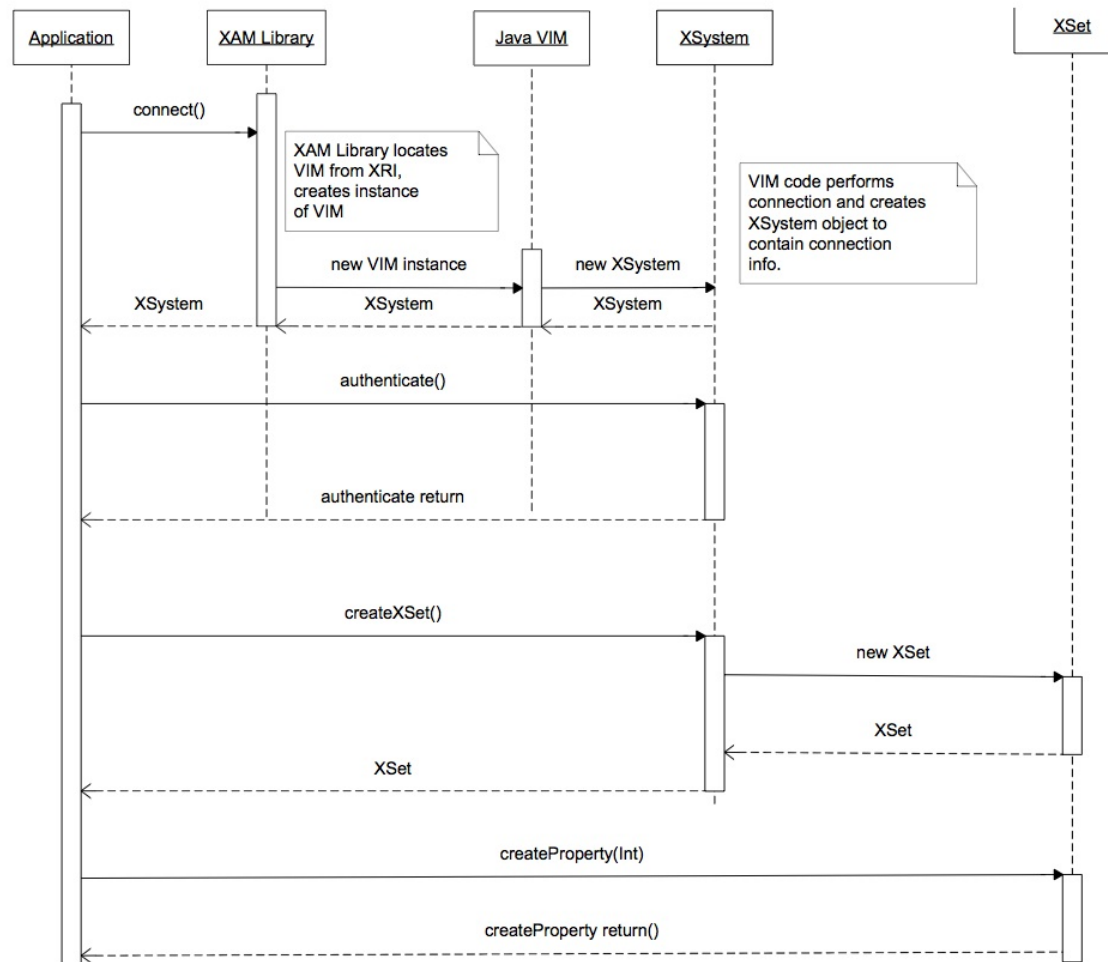
# UML of the Secondary Objects



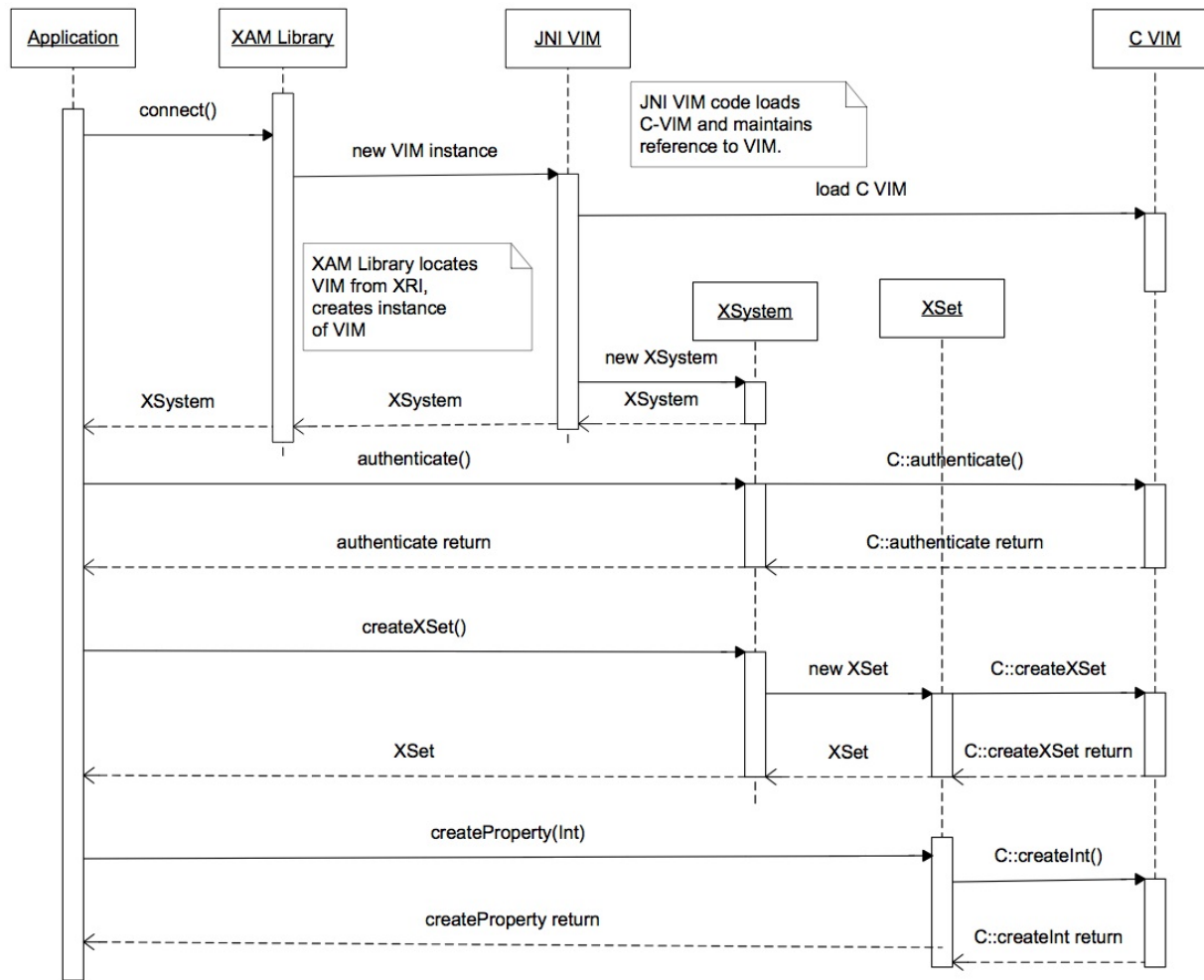
- The VIM Interface roughly maps to the public XAM API
  - ◆ Each object in the XAM API should have an analog in the VIM Interface
  - ◆ Each method in the XAM API should have an analog in the VIM Interface

- Note 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)

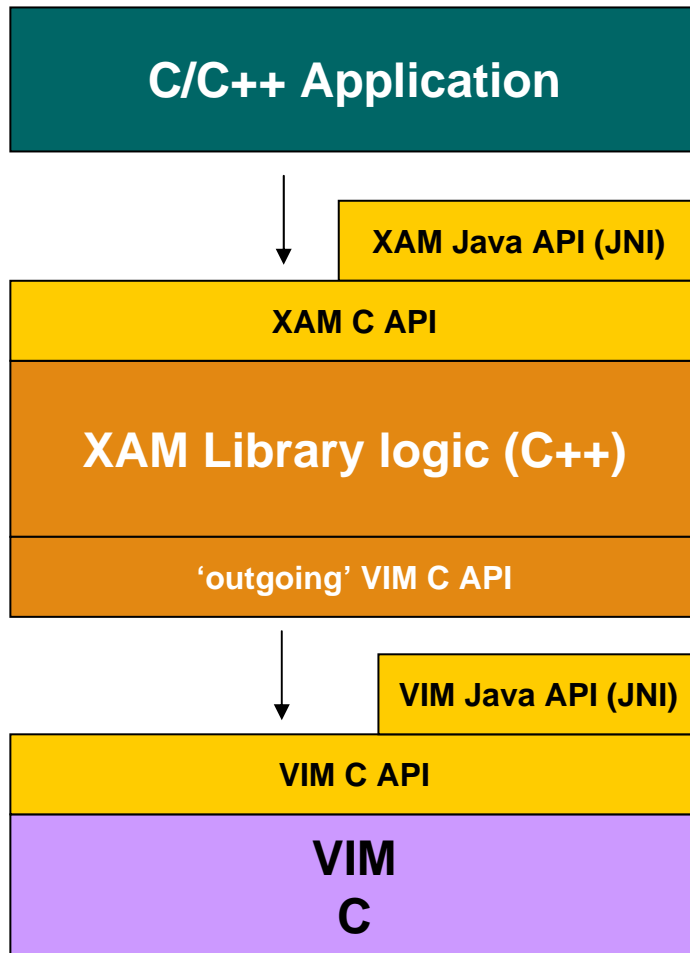


# Indirect Control (used by C/C++)

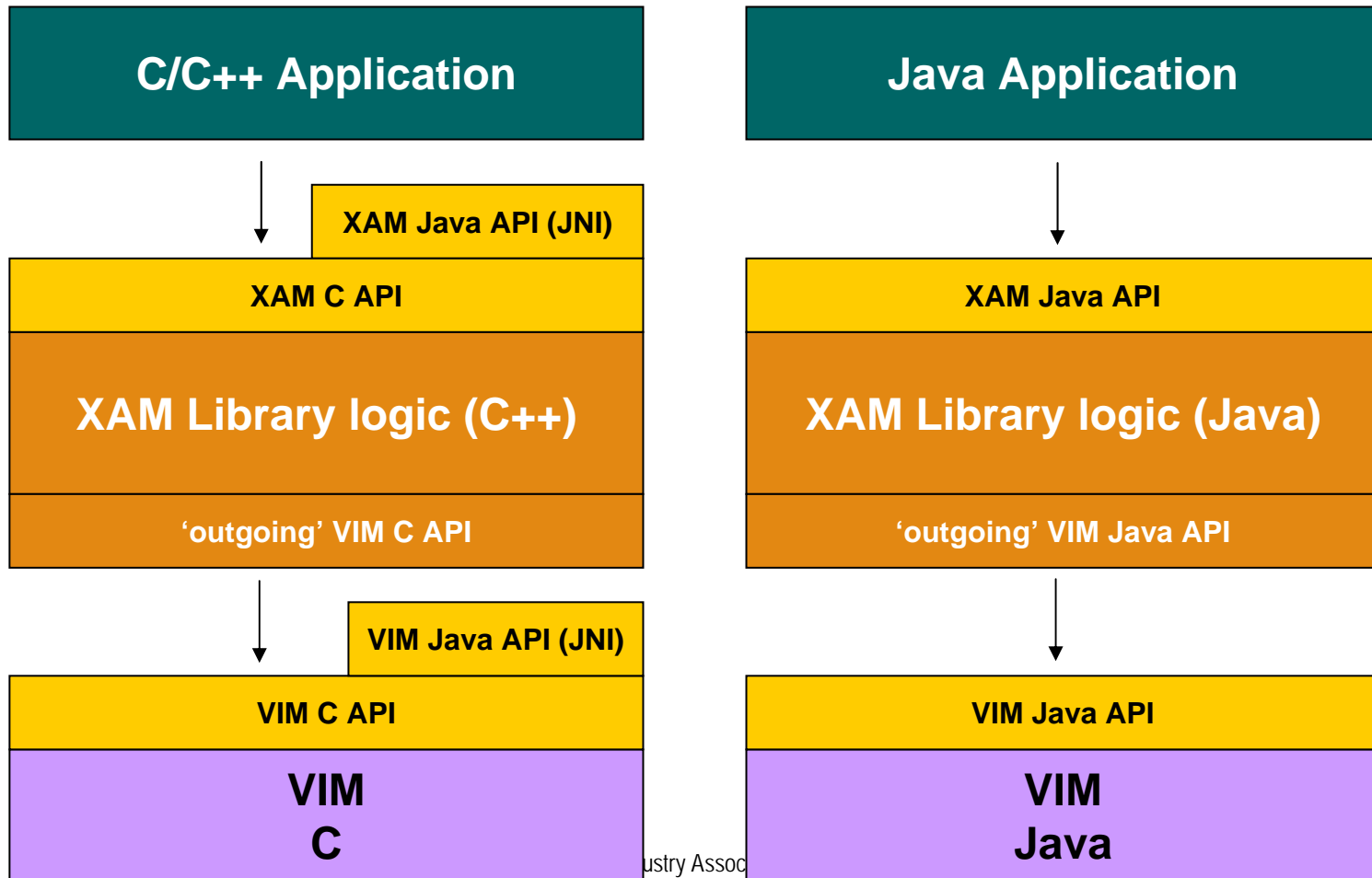


- As noted before, the application binds to the XAM API
  - ◆ Applications should never bind to the VIM interface!
  - ◆ It is the responsibility of the XAM Library to call into the VIM, not the application.
- The VIM interacts with the Storage System
  - ◆ The XAM Library never interacts directly with the underlying Storage System; all 'communication' is routed thru the VIM

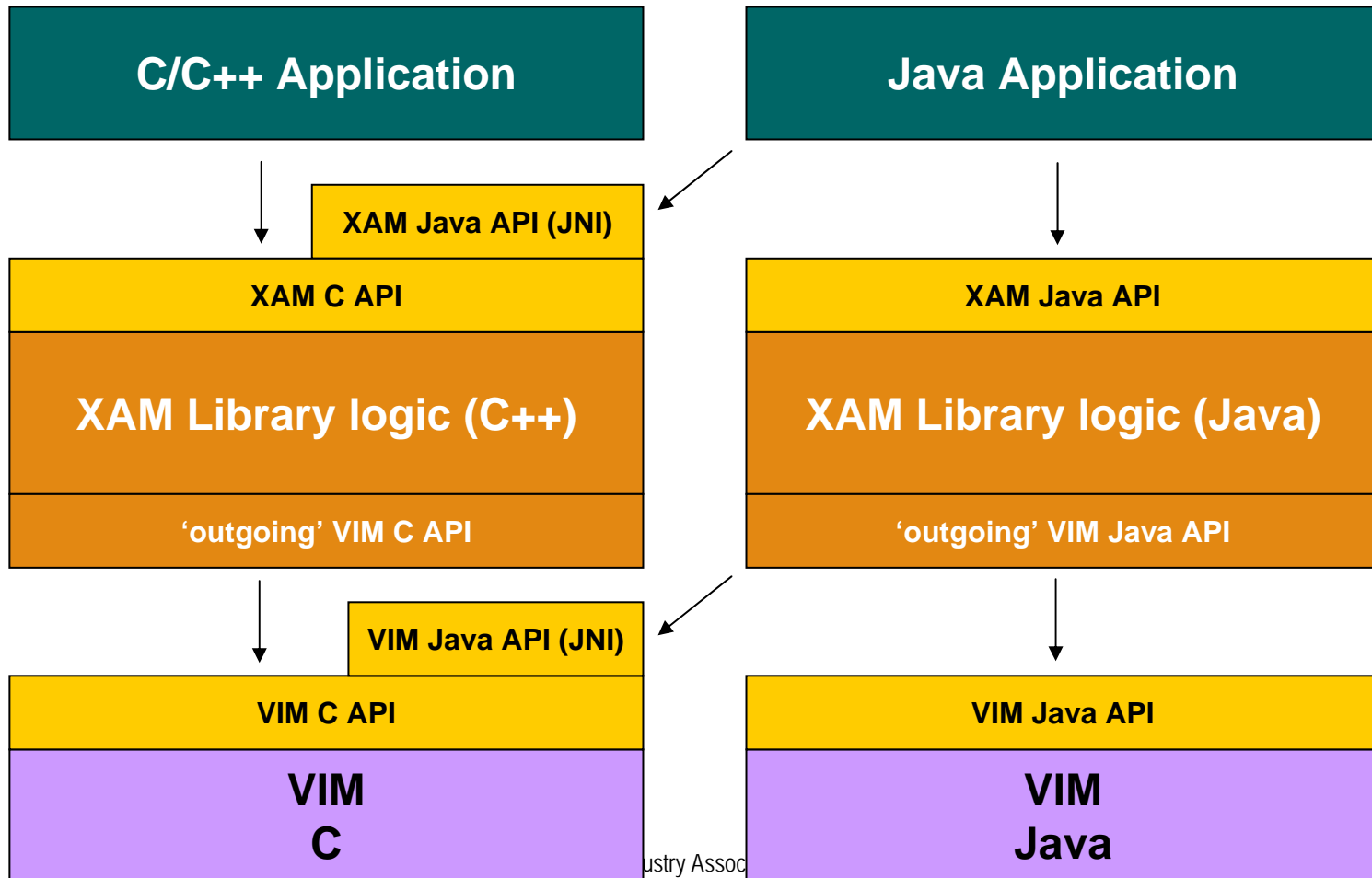
# XAM Application software stack (C/C++)



# XAM Application software stack (Pure Java)

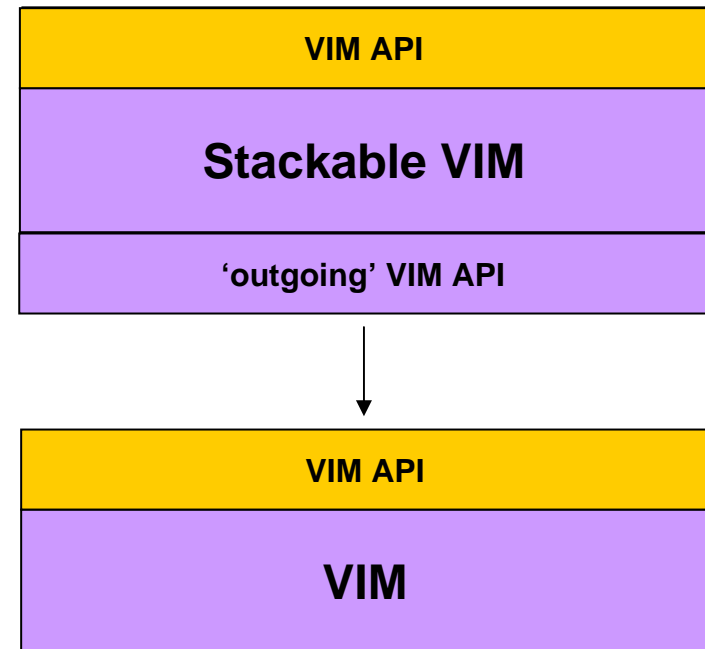


# XAM Application software stack (Unified)

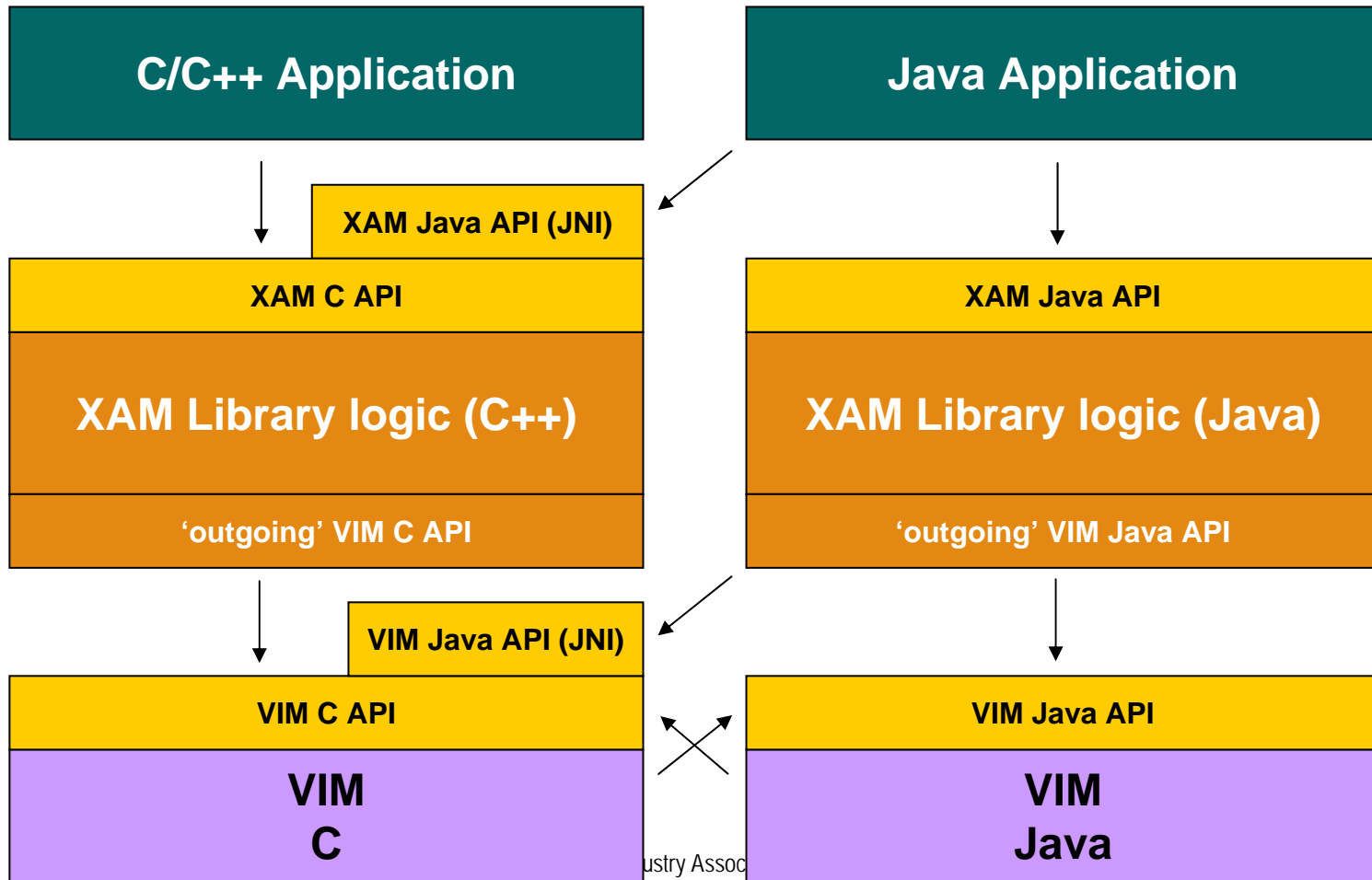


# A 'stackable' VIM

- 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)



# Using a stackable VIM to isolate XAM component

- ◆ The Isolation 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)

- ◆ <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/>

- Please send any questions or comments on this presentation to the SNIA:  
[trackvirtualization@snia.org](mailto: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**



Education

**Thank You!**

Mark A Carlson, SNIA Technical Council, Sun Microsystems