Information Management – Extensible Access Method (XAM) – Part 3: Java API

Version 1.0

"Publication of this Working Draft for review and comment has been approved by the FCAS TWG. This draft represents a "best effort" attempt by the FCAS TWG to reach preliminary consensus, and it may be updated, replaced, or made obsolete at any time. This document should not be used as reference material or cited as other than a "work in progress." Suggestions for changes to this document should be directed to the SNIA Technical Council Managing Director at tcmd@snia.org."

WORKING DRAFT

April 2, 2008

Revision History

Version	Date	Originator	Sections	Comments
1.0	4/2/08	M. McMinn	Various	Foreword - updated doc titles; All - changed status from Internal Use Draft to Working Draft.

Contents

Figures

Tables

Foreword

Parts of this Standard

This standard is subdivided in the following parts:

- Information Management – Extensible Access Method (XAM) – Part 1: Architecture
- Information Management – Extensible Access Method (XAM) – Part 2: C API
- Information Management – Extensible Access Method (XAM) – Part 3: Java API

SNIA Web Site

Current SNIA practice is to make updates and other information available through their web site at http://www.snia.org

SNIA Address

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent to the Storage Networking Industry Association, 500 Sansome Street, Suite #504, San Francisco, CA 94111, U.S.A.

Introduction

Purpose and Audience

This document forms part of the XAM Software Development Kit (SDK). It is a complete reference document for Java application development using the XAM API. It is intended for experienced programmers, for those developing applications that interface with storage systems which support the XAM API, and for those developing components of the XAM Library itself.

For an overview of the SNIA XAM, refer to the Business Overview chapter in the [XAM-ARCH].

Organization

The chapter contents of this document are described as follows:

Chapter	Contents
Chapter 1, “Scope”	Defines the subject of the document and the aspects covered.
Chapter 2, “Normative References”	Lists the referenced documents that are indispensable for the application of this document.
Chapter 3, “Terms and Conventions”	Defines the terms and conventions used in this document.
Chapter 4, “Java API Overview”	Contains an overview of the Java API.
Chapter 5, “Public Java API Reference”	Contains a reference guide to the public Java API for applications.
Chapter 6, “Private (VIM) Java API Reference”	Contains a reference guide to the private Java API for the VIMs.
Annex A, “(normative) Public Interfaces”	Contains the Java code for implementing the methods described in Chapter 5, “Public Java API Reference”.
Annex B, “(normative) VIM Interface”	Provides a series of methods allowing the XAM Library to communicate to VIMs.
Annex C, “(normative) Java-Specific Toolkit”	Defines toolkit functions that will extend the XAM Java API to make it easier to use with Java run-time systems.
Annex D, “(informative) Java API Method Mapping”	Lists the methods in [XAM-ARCH] and the corresponding method name for the Java binding.

1 Scope

This part of the XAM standard specifies the syntax of the Java application programming interface (Java API). This document also specifies how a Java-based XAM Library is to map Java VIM API calls to C-based VIM implementations, and vice versa. It applies to programmers who are generating XAM applications in the Java programming language. It also applies to storage system vendors who are creating vendor interface modules (VIMs) in the Java programming language.

This document does not normatively specify the semantics of the interfaces; the specification of the semantics in the XAM standard is contained in the XAM Architecture Specification [XAM-ARCH]. Any semantics described in this document are intended to be informative and to simplify the understanding of the interfaces described herein.

2 Normative References

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

[IANA-SASL] “Simple Authentication and Security Layer (SASL) Mechanisms” http://www.iana.org/assignments/sasl-mechanisms

[JAVA-SASL] “The Java SASL API Implementation and Deployment Guide”, Sun.com

[XAM-ARCH] “Information Management - Extensible Access Method (XAM) - Part 1: Architecture”, SNIA draft specification.

[XAM-C-API] “Information Management - Extensible Access Method (XAM) - Part 2: C API”, SNIA draft specification.

3 Terms and Conventions

3.1 Terms

For the purposes of this document, the definitions in the [XAM-ARCH] apply.

3.2 Conventions

Typographical conventions used in this document include the following:

Convention	Description
Note:	Contains additional or useful informative text.
CAUTION:	Indicates that you should pay careful attention to the probable action, so that you may avoid system failure or harm.
`Fixed-width text`	Indicates text that you enter at a keyboard or text that is displayed on an output device, such as a screen. This convention is most commonly used for command syntax and examples.
Italicized text	Indicates a property or field name, i.e., *.xset.xuid*.

4 Java API Overview

4.1 Basic XAM concepts

As an interface, XAM abstracts access methods from storage and provides a globally flat namespace. This interface supports the mobility of information, independent from storage, to allow longevity, distribution, and management of information. The XAM interface is intended to achieve interoperability, storage transparency, and automation for Information Lifecycle Management-based practices, long-term records retention, and information assurance (security).

The primary design goals behind the XAM interface are as follows:

- Provide a generic interface for applications: XAM interface methods have the same syntax and semantics without regard to the underlying storage. No methods were created 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: there was a desire to keep the interface as simple and small (e.g., have as few API methods as possible, and keep these methods easy to use and understand), yet at the same time, make sure 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 scalabilty), then a new method was not created for that function.
- Expose no implementation detail: the interface does not expose any internal functionality which would serve to place restrictions on storage system vendors.

XAM consists of a set of libraries. The ‘topmost’ library will contain the public XAM interfaces; thus, only the topmost library will be used by applications that wish to integrate with the XAM API. However, extension libraries may also be provided that 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 use 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. The implementation details of the VIMs themselves are beyond the scope of this document. The XAM API programmer should view the VIM as an internal implementation detail and avoid coding with specific VIMs in mind, if portable code is the goal. For more detailed information on the architecture of XAM, please see [XAM-ARCH].

The architecture of the XAM SDK is briefly illustrated in Figure 1, “XAM architecture”:

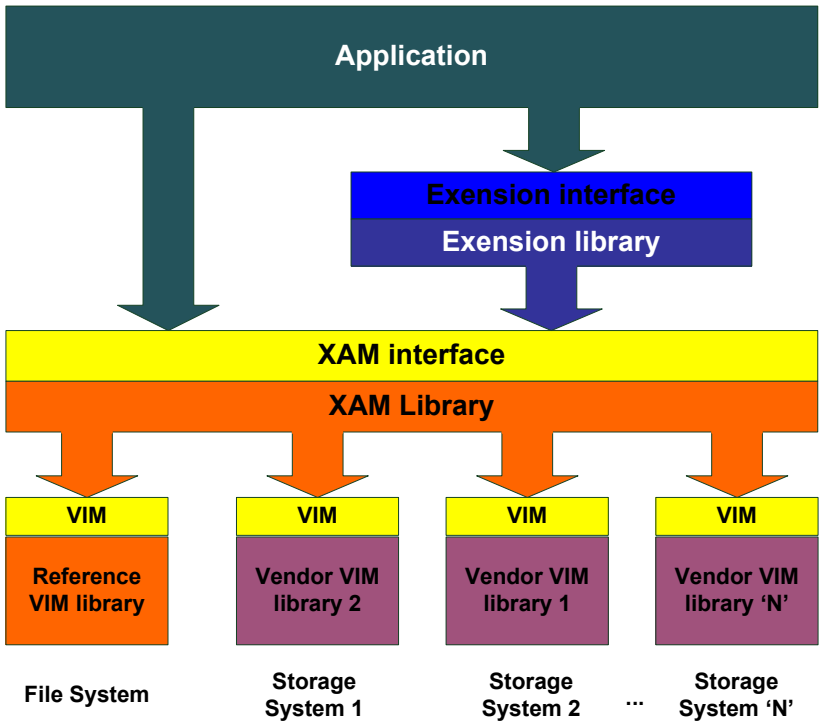

Figure 1 – XAM architecture

4.2 The XAM programming model

The XAM interface programming model supports a hierarchy of class constructs in a containment/ aggregation organization.The top level contains the singleton XAM object itself. Below (inside) the XAM object is one or more XSystems. Finally, each XSystem can contain XSets. Note that all of these object classes contain fields, and these fields are accessed in the same way without regard to the class of object that contains the field.

4.2.1 The XAM Library object

Pronunciation zam: The XAM object is the top level class for the XAM API library.

- It contains methods to get fields that describe the configuration of the XAM system.
- It contains methods to set fields that control the configuration of the XAM system.
- It acts as a factory for XSystems.

4.2.2 An XSystem

Pronunciation ‘ek-sis-tm: An XSystem is the class that abstracts the connection between the application and storage system.

- It encapsulates any resource management associated with the connection.
- It contains those methods used to authenticate operations.

- It acts as a virtual storage system, partitioning content.
- It is a factory for creating XSystem instances.

In this document, we will refer to an XSystem as a single storage unit. Applications can only perform XSystem functions when an XSystem is open; otherwise, run-time errors will be generated.

4.2.2.1 Authentication of XSystem instances

An XSystem instance must be authenticated in order to be useable. As defined in the XAM Architecture Specification, authentication is initiated using the SASL protocol. Client application authors and Storage VIM authors may find the *java.sasl* package to be useful [JAVA-SASL].

Any method executed on any XAM object can potentially throw the non-fatal error *org.snia.xam.AuthenticationExpiredException.* While this exception does not result in a corrupt object, the method must be re-executed in order to affect the XAM object. Applications are encouraged to understand the authentication expiration cycle and take proactive steps to maintain authentication. See Chapter 11 of the XAM Architecture Specification [XAM-ARCH].

4.2.3 An XSet

Pronunciation ‘ek-set: An XSet is the class that contains application data and metadata.

- The XSet is assigned a globally unique identifier when stored. This globally unique identifier is called a XUID (pronounced ‘zoo-id), which stands for **X**Set **U**nique **Id**entifier.
- Data and metadata (content) stored in the XSet is marked as binding or nonbinding. A contract exists between the binding content of the XSet and XUID, such that if any data or metadata in the XSet changes, the identifier will become invalid (or more precisely, a new XSet will be created with a new XUID). Nonbinding content can be changed with no effect on the XUID.

4.2.4 Fields (properties and XStreams)

Pronunciation feeld: A field is the construct where XSets, XSystems, and XAM objects store actual data and metadata. Fields have a number of attributes, which are listed below:

- Fields have names: Field names are assigned by the creator of the field and are unique within the container’s scope.
- Fields have types: Field types are assigned by the creator of the field.
- Fields have values: These values can be changed, but the semantics of what happens to an XSet that contains a field depends on the binding nature of the field.
- Fields have lengths: These lengths are derived from the type and value assigned to the field but cannot be directly set by the application.
- Fields can be binding or nonbinding: This attribute is assigned by the application. Note that only fields on XSets can be marked as binding.
- Fields can be read/write or readonly: These attributes are controlled by XAM and cannot be set by the application.

4.2.4.1 Type and length attributes – properties vs. XStreams

Field types are identified using MIME types. XAM defines some primitive or “simple” MIME types (stypes), which include xam_boolean, xam_int, xam_double, xam_string, xam_datetime, and xuid types. The

associated MIME types are, respectively; “application/vnd.snia.xam.boolean”, “application/vnd.snia.xam.int”, “application/vnd.snia.xam.double”, “application/vnd.snia.xam.string”, “application/vnd.snia.xam.datetime”, and application/vnd.snia.xam.xuid”. These types all have fixed sizes (even the string type). Fields that have one of these MIME types are referred to as properties. Note that when setting the value of a property, the XAM API will validate that the value is of the correct type (e.g., for XUID property fields, that the value actually contains a properly formatted XUID). The mapping between field type and field length is described in Table 1, “Field stypes (a.k.a. simple types)”:

Table 1 – Field stypes (a.k.a. simple types)

Stype	MIME type	Java Type	Length (in bytes)
xam_boolean	application/vnd.snia.xam.boolean	boolean	1
xam_int	application/vnd.snia.xam.int	long	8
xam_double	application/vnd.snia.xam.double	double	8
XUID	application/vnd.snia.xam.xuid	org.snia.xam.XUID	9 to 80
xam_string	application/vnd.snia.xam.string	java.lang.String	0 to XAM_MAX_STRING
xam_datetime	application/vnd.snia.xam.datetime	java.util.Calendar	0 to XAM_MAX_STRING

Other MIME types are also legal. In fact, any MIME type is acceptable. Fields with other MIME types (e.g., non-stypes) are referred to as XStreams. For XStream fields, the associated length is the number of bytes in the value. Unlike properties, XStreams are not validated. The application programmer is expected to validate that the specified value is of the specified MIME type.

4.2.4.2 Binding attribute vs. readonly attribute

Finally, we have the attributes binding and readonly. While these may seem related, they are, in fact, significantly different. To use the XAM API, you must understand the differences between these two field attributes.

Binding fields are those fields whose values participate in the contract of the XSet, binding the name of the XSet to the data of the XSet. Thus, if a field whose binding attribute is set to TRUE is changed, it is said that a new XSet has been created and on storing (committing) the XSet, a new XUID will be generated. The original XSet (and its requisite XUID) will be unchanged. Fields whose binding attribute is set to FALSE (nonbinding) can be changed without affecting the XSet/XUID contract. Thus, if an XSet only has nonbinding fields changed, the XUID will be unchanged when the modified XSet is committed. Because only XSets (not XSystems or XAM objects) can be committed, this field attribute can only be set on XSet fields. The binding attribute can be set by applications.

The readonly attribute controls if the application is allowed to edit a field at all. A field with the readonly attribute set to TRUE will generate a run-time error when any method is used to edit the field. The readonly attribute is set by XAM; applications cannot alter the readonly attribute. Note that while having a field’s readonly attribute set to TRUE may seem similar to setting the field’s binding attribute to TRUE, it is not. A field may be binding and readonly, in which case, an error will occur when trying to edit the field. A field may be binding and read/write (e.g., readonly = FALSE), in which case, the edit is allowed, but on commit of the XSet, a new XSet with a new XUID is created, and the original XSet/XUID pair is unchanged.

4.2.5 The XAsync

Pronunciation eks-’A-sink: The XAsync is an object used to access information about an asynchronous operation. These asynchronous operations allow applications to connect to XSystems and to read and write XSets that are associated with the XSystem, without blocking, or losing control of, the thread that

invokes the method. This object is returned when an asynchronous method is called, which allows applications to poll the status of the operations. The object is also passed as a parameter to any callbacks associated with an asynchronous method.

Applications may use the asynchronous operations to perform potentially long duration operations, without affecting the time-sensitive nature of their process flow. For instance, an application can provide a maximum duration on any of the asynchronous operations by using a timer utility. If the timer expires before the operation is complete, the application code may cancel the outstanding operation by calling the halt method on the associated XAsync object.

4.2.6 The XIterator

Pronunciation ek-’zi-ter-a-ter: The XIterator is a field discovery class. This interface was created because XSets, XSystems, and XAM objects can all have an arbitrary number of fields. (The maximum number of fields on an XSet is 2^63-1; while not actually an arbitrary number, it is still a lot.) The XIterator:

- Allows the discovery of all fields on the XSet, XSystem, or XAM object
- Takes a prefix that allows only a subset of fields to be discovered

4.2.7 XAM status

Pronunciation zam ‘sta-tus: XAM methods executed via the Java language bindings signal errors by throwing a Java exception. Successful execution of the method does not return an explicit error code.

4.2.8 The method hierarchy

The XAM, XSystem, and XSet classes are hierarchical in nature. An application uses a XAM method to create an XSystem instance and an XSystem instance to create an XSet instance. Different methods are

available when working at each level of the hierarchy. The hierarchical relationship between the methods of the XAM API is illustrated in Figure 2, “XAM API method hierarchy”.

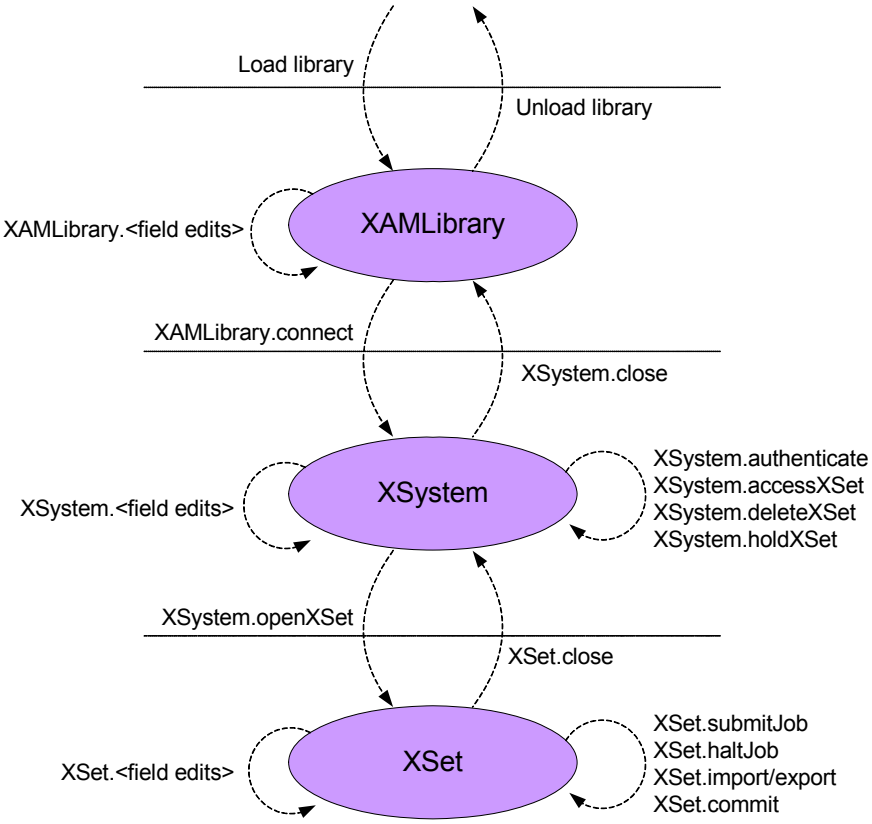

Figure 2 – XAM API method hierarchy

As illustrated in the hierarchy in Figure 2, field manipulation can be done at any level of the hierarchy and on any XSet, XSystem, or XAM object. Property fields can be accessed directly. However, XStream fields require the use of an XStream class to read and write to the field value. The XStream supports POSIX-like semantics, and XStreams open for reading allow seeking within the XStream. In addition, the ability to enumerate the field names of all fields on the XSet, XSystem, or XAM object is also needed at all levels of the hierarchy.

Figure 3 illustrates the relationship between these field methods:

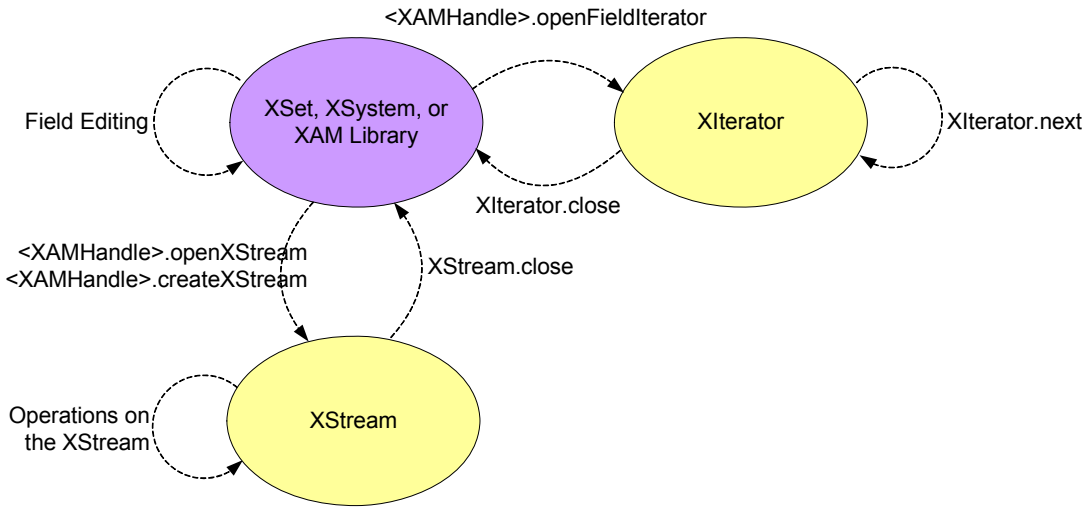

Figure 3 – XAM API field methods (includes properties and XStreams)

4.3 The XAM object interfaces

[XAM-ARCH] details the behavior of XAM Storage Systems and the methods that support the storage systems. The Java language bindings for the XAM API is implemented as a set of interfaces. Each of the interfaces represent a XAM primary or secondary object. The XAM API itself operates with references to these interfaces.

Table 2 – Java XAM interfaces

XAM Object	Java Object
XAM Library	org.snia.xam.XAMLibrary
XSystem	org.snia.xam.XSystem
XSet	org.snia.xam.XSet
-	org.snia.xam.FieldContainer
XStream	org.snia.xam.XStream
XAsync	org.snia.xam.XAsync
xasync_callback	org.snia.xam.XAsyncListener
XIterator	org.snia.xam.XIterator
xam_status	org.snia.xam.XAMException
-	org.snia.xam.vim.VIM

The UML description of the XAM interfaces is shown in Figure 4. This UML shows the inheritance relationship between the interfaces. Methods are listed, without full prototypes, to provide context to the reader. The normative method definitions are in Chapter 5, “Public Java API Reference”.

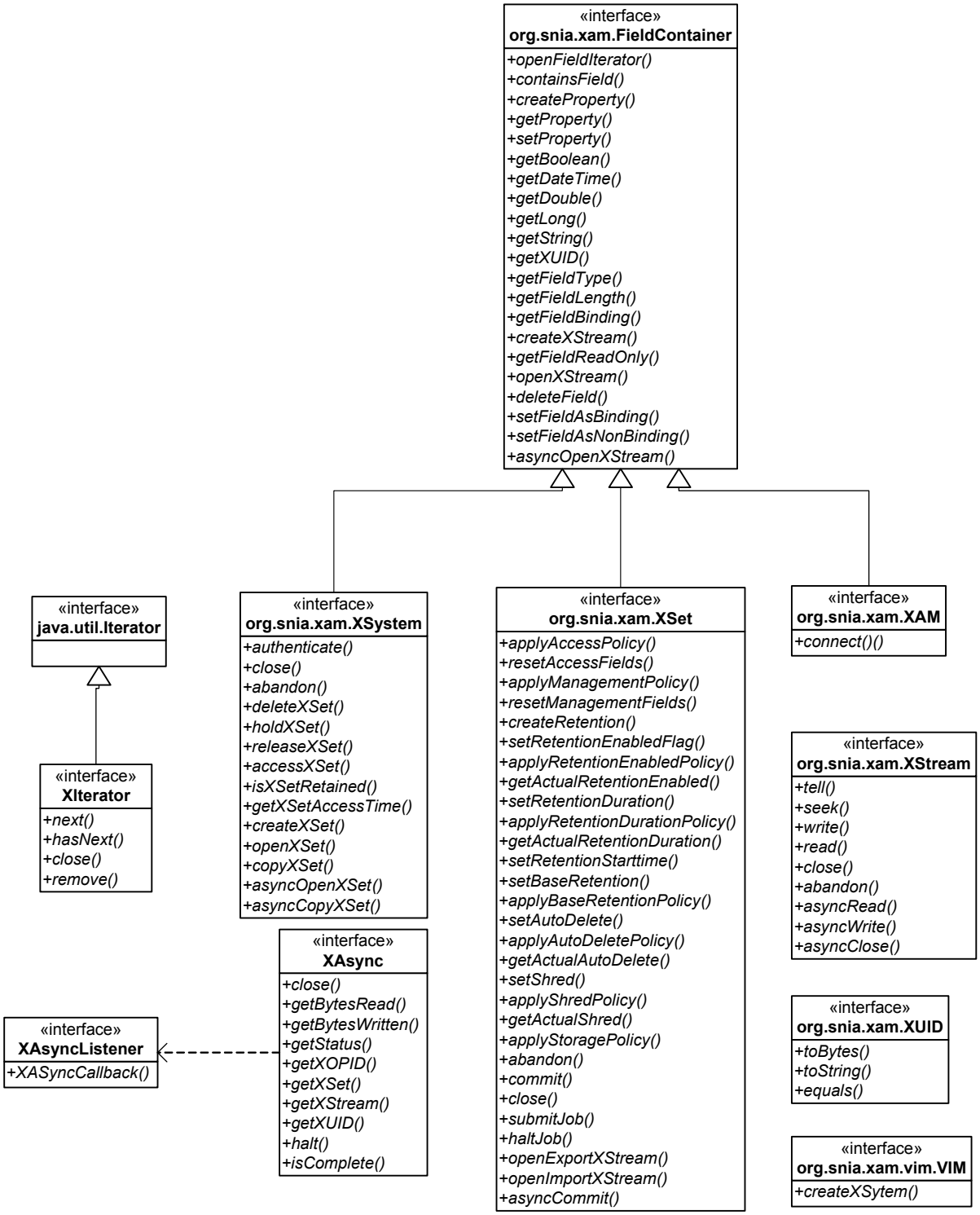

Figure 4 – UML description of XAM interfaces

The intention of the interface design is to allow VIM objects to be manipulated by the application with as little mediation by the XAM Library as possible. A VIM is loaded by the XAM Library in response to a connect method call. The XAM Library will then call connect on that VIM. The VIM constructs an XSystem object, and it is returned to the application. All further calls will be through the XSystem, and thus in VIM-supplied code.

The only other VIM-specific method is update. The update method is used when a change to the XAM Library fields needs to be communicated to the VIM. For instance, an application setting a logging property would cause the XAM Library to call update for each VIM, passing in the XAM Library instance. The VIM may then interrogate the XAM Library using FieldContainer methods to read the fields.

4.3.1 org.snia.xam.XAMLibrary

This interface contains all the methods and behavior specified by the [XAM-ARCH]. An application creates an instance of a XAM Library and will use it via the XAM interface. Details on obtaining the library instance are not specified by either the [XAM-ARCH] or this document.

4.3.2 org.snia.xam.XSystem

This interface contains all the methods and behavior specified by the [XAM-ARCH]. An application may only create instances of objects that are implementing this interface by calling XAMLibrary.connect.

4.3.3 org.snia.xam.XSet

This interface contains all the methods and behavior specified by the [XAM-ARCH]. An application may only create instances of objects that are implementing this interface by calling XSystem.createXSet, XSystem.openXSet, or XSystem.copyXSet.

4.3.4 org.snia.xam.FieldContainer

This interface is not specified by the [XAM-ARCH]. A FieldContainer is a common interface which is implemented by XAM, XSystem, and XSet objects. This interface defines field manipulation methods that are common to all three XAM objects. No FieldContainer objects exist apart from a XAM, XSystem, or XSet object.

4.3.5 org.snia.xam.XStream

This interface contains all the methods and behavior specified by the [XAM-ARCH]. An application may only create instances of objects that are implementing this interface by calling FieldContainer.openStream or FieldContainer.createStream. The methods defined by this interface match the POSIX-style stream methods defined in the [XAM-ARCH]. For interoperability with most of the Java I/O system, a wrapper class providing Java I/O stream capabilities should be provided by a XAM Java implementation.

4.3.6 org.snia.xam.XAsync

Pronunciation eks-’A-sink: The XAsync is an object used to access information about an asynchronous operation. These asynchronous operations allow applications to connect to XSystems and read and write XSets associated with the XSystem without blocking, or losing control of, the thread that invokes the method. This object is returned when an asynchronous method is called, which allows applications to poll the status of the operations. The object is also passed as a parameter to any callbacks associated with an asynchronous method.

4.3.7 org.snia.xam.XASyncListener

The XAsyncListener in an interface that an application may implement to provide notification when an XAsync operation has completed. When the operation completes, the XAM Library will call the method XAsyncListener.XAsyncCallback, passing in the associated XAsync object. The application may then perform its completion logic as appropriate.

An application may simply respond to completion of asynchronous operations, or it may perform more advanced operations. For instance, an application can limit the amount of time an operation is active by using a timer task to cancel the asynchronous operation, if the operation has been active for longer than an application-defined maximum time period.

4.3.8 org.snia.xam.XIterator

Applications have access to collections of fields from the XAM Library, XSystems, and XSets. The org.snia.xam.XIterator interface extends the Java interface java.util.Iterator. The methods next and hasNext are required to be implemented by the XIterator interface. Note that the interface defined by java.util.Iterator also specifies an optional remove method. XAM XIterator implementations shall provide the method to satisfy the interface, but it shall throw the exception java.lang.UnsupportedOperationException.

All open XIterators shall be closed before closing the FieldContainer from which the Iterator was opened. The XIterator interface specifies the close method and shall be consistent with the XAM Architecture Specification.

4.3.9 org.snia.xam.XAMException

Errors encountered during XAM API method calls will cause an exception to be thrown. All possible exceptions extend the base class XAMException. This API specification does not specify all possible exceptions which may be thrown. All implementations are required to extend this base class.

4.3.10 org.snia.xam.vim.VIM

All VIM implementations must implement the VIM interface. This requirement allows the XAM Library to load VIMs, connect to systems, and provide XAM capabilities to the application. After the XAM Library has connected to a XAM Storage System, the VIM provides an object that implements an XSystem interface. All further operations by the application are then directed to the VIM by using the polymorphism inherent in Java.

4.4 VIM implementation models

A Java version of the XAM Library still relies on VIMs to mediate between the XAM Library and a XAM Storage System. As required by the [XAM-ARCH], VIMs may be written in Java or other languages. Because non-Java code may not run on the computer architectures that a JVM would, use of a non-Java VIM may limit the number of computer architectures in which an application may be deployed.

VIMs generally should be considered as a collection of classes that implement all of the required XAM Java interfaces. These VIMs may or may not use additional code to accomplish its task. The use of this additional code is acceptable, but VIM authors are cautioned to ensure that this additional code is not used in a way which will limit VIM portability between XAM Library implementations.

VIM authors should note that the VIM interface for the Java libraries is very different than the VIM interfaces required by the C implementation of a XAM Library. This difference is because of basic differences between polymorphism implemented in C and Java. The Java implementation of the XAM Library strongly leverages the use of polymorphism. When the application connects to a XAM Storage System, the VIM object returns an object that implements the XSystem interface. This object is *not* part of

the XAM Library; it is completely storage-vendor defined. This vendor object contains the code appropriate to interact with the storage vendor's system. Likewise, XSet instances are *not* part of the XAM Library.

For purposes of discussion, a VIM supported by a Java XAM Library is a collection of classes implementing the required interfaces. These classes will utilize other functionality as required by the XAM Storage System vendor.

4.4.1 Java VIMs

For a Java-implemented VIM, most of the methods called by the application will be sent directly to VIM-supplied objects (e.g., XSystem instance, XSet instance). This happens because when some of the factory methods are called, the vendor code is invoked directly to generate an instance that satisfies the interface. For instance, the application will call XAMLibrary.connect, which then causes the library to locate the correct VIM, and then call connect on the VIM, which returns a VIM-created XSystem object. After these calls, when the application calls XSystem methods, these calls will be directed into VIM-supplied code. Figure 5, "Application interaction with pure Java VIM" illustrates this example. In this example, the application connects to an XAM Storage System and then uses the XSystem instance to create an XSet.

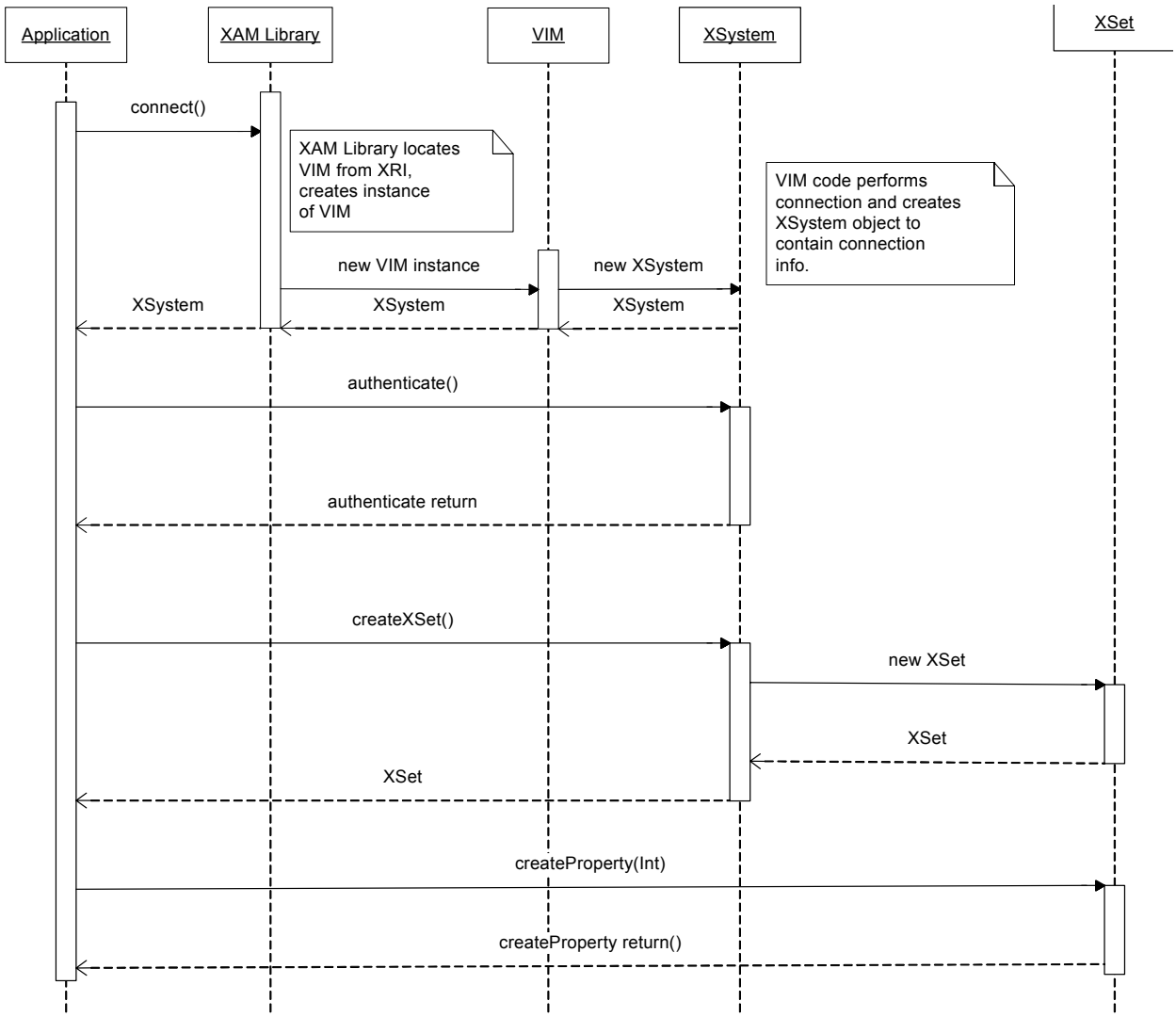

Figure 5 – Application interaction with pure Java VIM

4.4.2 Non-Java VIMs

Since most Java implementations allow for linking to non-Java code (JNI), non-Java VIMs can be used. Doing this with the most common JNI model is relatively straightforward. All C-based VIMs shall follow the standard VIM API as defined in the C API Specification [XAM-C-API].

4.4.3 VIM Initialization

As described in the XAM Architecture Specification [XAM-ARCH], the XAM Library copies XAM field information into the XSystem during initialization. As illustrated in Figure 6, “VIM Initialization”, the XAM Library asks the VIM to create a new, unconnected, unauthenticated XSystem instance. Before connecting to a system, the XAM Library copies all the XAM Library fields to the XSystem, using createProperty/ setProperty or XStream methods.

The XAM Library first creates the property, *.xsystem.initializing*, with the value of TRUE. Then the XAM Library copies all fields to the XSystem instance. Third, the XAM Library sets the value of *.xsystem.initializing* to FALSE. Finally, the unconnected and unauthenticated XSystem instance is returned to the application.

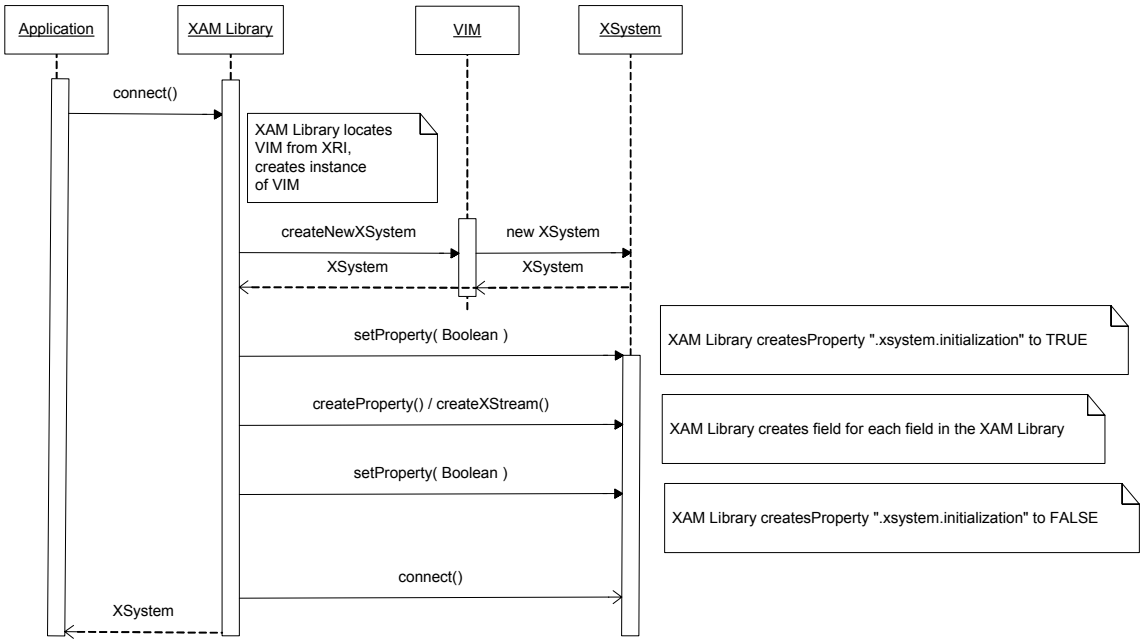

Figure 6 – VIM Initialization

4.5 Using the XAM API – abstract samples

The following samples show how to program a few common operations using the SNIA XAM Java API. Note that these examples are not intended to exhaustively show all possible methods or illustrate a strong and robust error-handling use case.

4.5.1 Write an XSet

This example creates an XSet, storing a xam_string property and a JPEG stream.

```
// Connect to the archive system
String xri = "snia-xam://ExampleVIM!lcom.example.xamstore"
XSystem sys = s_xam.connect(xri);

// NOTE: Authentication code not shown.

// Create an XSet to store the data in
XSet xset = sys.createXSet(XSet.MODE_UNRESTRICTED);
xset.createProperty( "com.example.name_of_subject", false, "John Smith" );
XStream str = xset.createStream( "com.example.picture", false, "image/jpeg" );

// Write the bytes of a JPEG image
str.close();

XUID new_id = xset.commit();
xset.close();
sys.close();
```

4.5.2 Read an XSet

This example opens the same XSystem and reads the data, using the XUID value from the previous example.

```
// Connect to the archive system
String xri = "snia-xam://ExampleVIM!lcom.example.xamstore"
XSystem sys = s_xam.connect(xri);

// NOTE: Authentication code not shown.

// Open the XSet from which to read data
XSet xset = sys.openXSet( new_id, XSet.MODE_READ_ONLY);
String value = xset.getString( "com.example.name_of_subject" );
XStream str = xset.openStream( "com.example.picture", XStream.MODE_READ_ONLY);

// Read the bytes of a JPEG image
str.close();

xset.close();
sys.close();
```

4.5.3 Query for data with the string literal

This method connects to the same XSystem (using the same connection string), runs a query job querying for XSets that have the “name of subject” fields set to “John Smith”, opens the XSets one at a time, and processes the contents.

Note: This example uses an XUID implementation, named XUIDImpl, which is presumed to follow the toolkit requirements detailed in Annex C, “(normative) Java-Specific Toolkit”.

```
// Connect to the archive system
String xri = "snia-xam://ExampleVIM!lcom.example.xamstore"
XSystem sys = s_xam.connect(xri);

// NOTE: Authentication code not shown.

XSet    query;

query = sys.createXSet( XSet.MODE_READ_ONLY );
query.createProperty( XSet.XAM_JOB_COMMAND, false, XSet.XAM_JOB_QUERY );
query.createProperty( XSet.XAM_JOB_QUERY_COMMAND, false,
       "select \".xset.xuid\" where \"com.example.name\" = \"John Smith\"" );

query.submitJob();

// Wait for the job to complete.

String queryStatus = query.getString( XSet.XAM_JOB_ERROR );
if( queryStatus.equals( XSet.XAM_JOB_ERRORHEALTH_OK ))
{
    XStream results = query.openXStream( XSet.XAM_JOB_QUERY_RESULTS,
                                         XStream.MODE_READ_ONLY );
    byte rawXUID[] = new byte[80];
    int  bytesRead = 0;
    while( (bytesRead = results.read(rawXUID)) >= 0 )
    {
        XUID localXUID = new XUIDImpl(rawXUID);
        XSet data = sys.openXSet( XSet.MODE_READ_ONLY );
        // Process results
        data.close();
    }
    results.close();
}
query.close();
```

5 Public Java API Reference

This chapter describes the public interfaces of the XAM Library. These interfaces are intended to be used by application programmers.

5.1 Design goals

Some simple design goals were kept in mind while defining the XAM Java API. These goals are for all methods to:

- Return output values by reference
- Support the XAM object model
- Be thread safe
- Support asynchronous operations for operations in the data path
- Present a minimum number of methods
- Favor compilation errors over runtime errors

5.2 Supporting data types

5.2.1 stypes

All XAM fields have type information that is described using MIME types. Complex fields require that the value of the field (the data associated with the field) be stored in an XStream. However, some predefined MIME types have also been defined for XAM fields. These MIME types (also known as simple MIME types or stypes) have data types associated with them, which allows the values to be checked at compile time.

The stypes and the data types are defined in the XAM interface and are also described below:

- **“application/vnd.snia.xam.boolean”:** This MIME type is associated with a standard Boolean type, xam_boolean. A xam field with this type will have a length of 1. A valid field of this type will contain a zero (0) when FALSE or a non-zero value when TRUE.
- **“application/vnd.snia.xam.int”**: This MIME type is associated with a 64-bit integer value on all platforms, xam_int. Note that this is not the same as a standard long type. The value stored in this field can be positive or negative. A xam field with this type will have a length of 8.
- **“application/vnd.snia.xam.double”**: This MIME type is associated with a standard double precision float, xam_double. A xam field with this type will have a length of 8.
- **“application/vnd.snia.xam.xuid”**: This MIME type is associated with an 80-element byte array, xam_xuid. A valid field of this type will have a value that is a canonical XUID. A xam field with this type will have a length of 80.
- **“application/vnd.snia.xam.string”**: This MIME type is associated with a XAM_MAX_STRING element byte array, xam_string. A valid field of this type will have XAM_MAX_STRING or fewer bytes which describe the string. The encoding of a string type is UTF-8. Note that xam_strings may not contain nulls; thus, null termination will be used in the Java API to mark the end of a string. A xam field with this type will have a length which matches the number of bytes that describes the actual string; nulls or other trailing bytes are not included in the length.

- **“application/vnd.snia.xam.datetime”**: This field is associated with a XAM_MAX_STRING element byte array, xam_datetime. It is an ISO 8601-compliant timestamp string, UTF-8 encoded, with 4 digit years, negative years allowed, no truncated years, no week dates, no ordinal dates, no 24:00 representation of midnight, time zone designators allowed, no duration or interval formats, and a millisecond resolution.

5.2.2 XAM status type

Most methods in the Java API have the ability to throw a XAMException. When the method has completed successfully, no exception is thrown. XAMExceptions may be generated in the XAM Java Library or in VIM-supplied code. Each XAMException contains the xam_status error code and descriptive text. The structure of the xam_status code is a 32-bit integer, as defined below:

The top bit is used as a flag, while the remaining 31 bits are used to hold the status payload. The topmost bit (bit 0) is set to zero when the payload contains a standard value, and 1 when the payload contains a non-standard (vendor-specific) value. The status format is illustrated in Figure 7, “XAM status type diagram”:

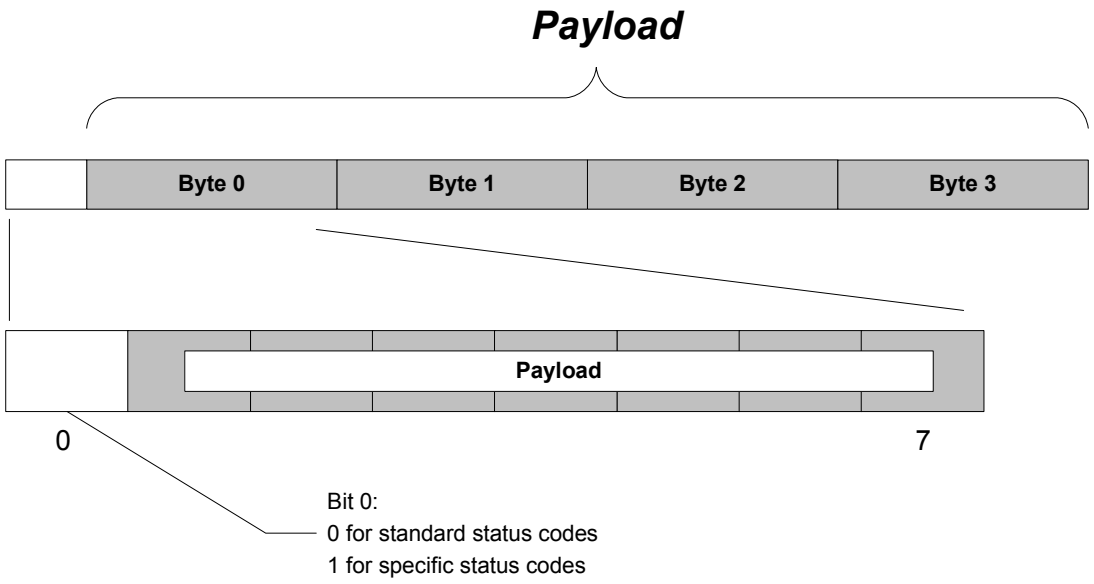

Figure 7 – XAM status type diagram

Note: Success is denoted with a status set to zero (bit 0 set to zero because it is a standard code, and the payload for success uses the standard code of 0).

The method XAMException getMessage returns the error token as a String. The string starts with a prefix (“xam” for standard errors or the reverse DNS of the vendor for non-standard errors) followed by a separator (“/”) and ends with a non-localized UTF-8 substring that briefly describes the error. For example, an “out of memory” error might generate the following error token:

```
"xam/out of memory"
```

This method requires an XSystem or a XAM Library object. If a XAM Library object is used, the method will not be able to generate vendor-specific error tokens. Such cases will result in the following error token:

```
"xam/unknown error"
```

XAMExceptions shall include both the status code and the error token, when printing the stack trace as part of the Java Exception printStackTrace method. The status code may be retrieved via the XAMException method getStatusCode.

5.2.3 XOPID

Every asynchronous method takes as an input argument a XAM asynchronous operation identifier (XOPID). It can be retrieved from either a pending or completed asynchronous operation. The XOPID type is as defined below:

```
long XOPID;
```

The XOPID is intended to provide a fast mechanism for the application to retrieve its state associated with the asynchronous operation. Because the 64-bit value is specified by the application and is opaque to the XAM Storage System, the application can attach any meaning to it that it wishes, including an index into an application's data structure, a pointer, or a bitfield.

5.2.4 Callbacks

Every asynchronous method takes a callback method as an optional input argument. The callback method will be called when the operation completes (either successfully or unsuccessfully). The XAM callback method is defined by the interface org.snia.xam.XAsyncListener.

```
public void XAsyncCallback( XAsync operation );
```

Within the callback routine, the XAM application should first retrieve the status of the operation. If the operation was successful, the XAM application can also retrieve the output arguments, using the appropriate methods. It can also retrieve the XOPID (see Section 5.2.3) to help retrieve the application state that is associated with the operation.

CAUTION: Since the callback method is called asynchronously from the XAM Library, the application must write the callback method in a thread-safe manner. The application-supplied listener shall not throw any exceptions.

5.3 Methods

5.3.1 XAM Library methods

public interface org.snia.xam.XAMLibrary implements org.snia.xam.FieldContainer

A XAM system represents the XAM Library running locally. An application wanting to use the XAM Library must create an instance of the XAM Library. This library object will manage and create instances of VIMs to connect to XAM Storage Systems. The XAM Library object may also contain toolkit functions for convenience.

5.3.1.1 connect

```
public org.snia.xam.XSystem connect(
     String XRI)
```

Connects to the specified XAM archive and returns an XSystem object, to allow the application to use the XAM API. XRI values are in the form of VIM names, and parameters are optional. Use of optional

parameters is specific to any XAM Storage System and its configuration. The XRI format is defined in section 7.2.1 of the XAM Architecture Specification [XAM-ARCH].

Note: A connected XSystem instance is not fully usable until authenticated.

- Parameters: XRI - The XAM Storage System Resource specification
- Returns: The XSystem object representing the connection
- Throws:
  - InvalidXRIException - The XRI is null or malformed.
  - ConnectException - A connection could not be made.
  - VIMLoadException - a VIM could not be found to handle this connection.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.2 XSystem methods

public interface org.snia.xam.XSystem implements org.snia.xam.FieldContainer

Represents a connection to a XAM Storage System. XSystem classes are implemented by storage system vendors and implement the XSystem interface. XSystem instances are created via the VIM.connect factory method. XSystem instances act as a factory for XSet instances and XStream instances.

5.3.2.1 XSystem connect

```
void connect(
     String xri)
```

Connects to the specified XAM Storage System. If connection was successful, a valid XSystem is returned. If connection fails, an exception will be thrown. NOTE: This method is NOT public, as it is not callable directly by the application. This method is used by the XAM Library to complete the XSystem initialization process.

- Parameters: xri - The application supplied XRI of the XAM Storage System
- Throws: XAMException - An error exists with the underlying XAM Storage System or VIM.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until complete.

5.3.2.2 authenticate

```
public byte[] authenticate(
     byte[] buffer )
```

This method will allow an application to authenticate an XSystem instance. It provides a generic interface to exchange data as part of the authentication process. The application must check the XSystem properties, prefixed constant value XAM_XSYSTEM_SASL_LIST, to determine which patterns of

authentication are available for use. After a pattern is selected, the appropriate sequence of data exchanges should be made (using this call) in order to authenticate. A failed authentication will make the XSystem instance unusable. The application cannot repeat failed authentications using the same XSystem instance.

- Parameters: buffer - Data being passed to the authentication system. The XSystem will use buffer.length to determine the significant bytes to be used by the authentication process. The format of the buffer differs, depending on the authentication mechanism chosen. Application authors are encouraged to refer to the appropriate SASL specifications [IANA-SASL].
- Returns: Output from the authentication method.
- Throws:
  - AuthenticationException - The authentication process encountered an error.
  - InvalidArgumentException - The buffer argument is null.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.2.3 close

```
public void close()
```

Closes this XSystem instance, preventing any further changes. Any resources associated with this XSystem instance will be released. This method blocks until complete.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - ObjectInUseException - There are open XStreams on the XSystem preventing the close method from completing.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.2.4 abandon

```
public void abandon()
```

Abandons all changes to this XSystem. No unsaved data is written, and all open XSets are abandoned, allowing close to be called.

CAUTION: Inappropriate use of this call will result in data loss. Applications are encouraged to track open XSets and close them properly instead of using this method.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.2.5 deleteXSet

```
public void deleteXSet(
     XUID xsetid)
```

Deletes the specified XSet without opening it.

- Parameters: xsetid - XUID of the XSet to be deleted.
- Throws:
  - AuthorizationException - The delete operation is not allowed.
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - XSetUnderRetentionException - The XSet may not be deleted because of retention.
  - XSetUnderHoldException - The XSet is under hold and cannot be deleted.
  - InvalidArgumentException - XUID is null.
  - XSetInaccessibleException - XSet does not exist or is not accessible; it cannot be deleted.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.2.6 isXSetRetained

```
public boolean isXSetRetained(
     XUID xsetid)
```

This method evaluates all of the retention settings on the specified XSet, to determine if the XSet is currently under retention protection.

- Parameters: xsetid - XUID of the XSet to be evaluated.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - XUID is null.
  - XSetInaccessibleException - XSet does not exist or cannot be accessed.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.2.7 holdXSet

```
public void holdXSet(
     XUID xsetid,
     String holdID)
```

Puts the specified XSet on hold. This method prevents applications from making changes to the XSet. An XSet may participate in multiple holds. The hold is specified by the application.

- Parameters:
  - xsetid - The XSet to hold.
  - holdID - The ID of the HOLD operation.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - AuthorizationException - The application is not authorized to place a hold.
  - InvalidFieldNameException - The field name for the hold is too long or malformed.
  - HoldIdException - The hold id is already in use
  - InvalidArgumentException - xsetid or holdid arguments are null.
  - XSetInaccessibleException - XSet does not exist or is not accessible.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.

- Blocking: This method blocks until completion.

5.3.2.8 releaseXSet

```
public void releaseXSet(
     XUID xsetid,
     String holdID)
```

Releases XSet from a specific hold, making it subject to normal retention processing.

- Parameters:
  - xsetid - The XSet to be released.
  - holdID - The ID of the corresponding HOLD operation.
- Throws:
  - AuthorizationException - The application is not authorized to release a hold.
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - HolidIdException - The hold id specified is not used as a hold on this XSet.
  - XSetInaccessibleException - XSet does not exist or is not accessible.
  - InvalidArgumentException - xsetid or hold id arguments are null.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.2.9 accessXSet

```
public boolean accessXSet(
     XUID xsetid,
     long mode)
```

Tests if a specified XSet is accessible, according to the modes specified.

- Parameters:
  - xsetid - The XSet to test.
  - mode - Bitset of modes (READ_OK, WRITE_OK, WRITE_SYS_OK, DELETE_OK, HOLD_OK, EVENT_OK)
- Returns: TRUE if the application can access the XSet, FALSE otherwise.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - xsetid is null or the mode specified is invalid.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.2.10 getXSetAccessTime

```
public java.util.Calendar getXSetAccessTime(
     XUID xsetid)
```

Retrieves the XSet's last access time, most recent of either time of last open or commit. This method will not alter the access time of the XSet.

- Parameters: xsetid - The XUID of the XSet.
- Returns: The last access time of the XSet.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - xsetid is null.
  - XSetInaccessibleException - XSet does not exist or is not accessible.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.2.11 createXSet

```
public org.snia.xam.XSet createXSet(
     String createMode)
```

Create a new, unsaved XSet, in this XSystem. The open XSet will initially be fully writeable but will be in the specified mode after the first commit.

- Parameters: createMode - The mode the XSet should be in after the first commit. One of:
  - XSet.MODE_RESTRICTED
  - XSet.MODE_UNRESTRICTED
- Returns: The new XSet object.
- Throws:
  - AuthorizationException - The application is not authorized to create an XSet.
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - createMode is either null or an invalid mode value.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.2.12 openXSet

```
public org.snia.xam.XSet openXSet(
     XUID xsetid,
     String openMode)
```

Open the specified XSet for access by the application.

- Parameters:
  - xsetid - The XUID of the XSet to open
  - openMode - One of the following:
    - XSet.MODE_READ_ONLY
    - XSet.MODE_RESTRICTED
    - XSet.MODE_UNRESTRICTED
- Returns: The XSet object to be opened.

- Throws
  - AuthorizationException - The application is not authorized to use the specified mode.
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - xsetid or openMode are null, or the mode is invalid.
  - InvalidXSetModeException - The specified mode is not recognized by the XSystem.
  - XSetInaccessibleException - XSet does not exist or is not accessible.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.2.13 copyXSet

```
public org.snia.xam.XSet copyXSet(
     XUID   xsetid,
     String copyMode)
```

Copy the specified XSet to a new XSet. The new XSet will be "open" in the mode specified.

- Parameters:
  - xsetid - The source XSet to copy.
  - copyMode - One of the following:
    -]XSET_MODE_RESTRICTED
    - XSET_MODE_UNRESTRICED
- Returns: The newly copied XSet.

- Throws:
  - AuthorizationException - The application is not authorized to perform the copy.
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - xsetid or copyMode are null, or copyMode is invalid.
  - InvalidXSetModeException - The specified mode is not recognized by the XSystem.
  - XSetInaccessibleException - The specified XSet does not exist or is not accessible.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.2.14 asyncOpenXSet

```
public org.snia.xam.XAsync asyncOpenXSet(
     XUID            inXuid,
     String          mode,
     long            xopid,
     XAsyncListener  listener)
```

Begins the asynchronous opening of an XSet in the XSystem, ultimately returning a handle to an XSet instance associated with the XSystem. The specified callback will be invoked as part of the asynchronous opening. To monitor the status of this operation, the application can poll the Async instance that is generated by this method. A handle to an Async instance is also passed to any provided callback method, when that callback method is invoked.

- Parameters:
  - inXuid - The XUID of the XSet to be opened.
  - mode - The mode the XSet should be opened in. One of the following:
    - MODE_READ_ONLY
    - MODE_RESTRICTED
  - xopid - The application XOPID of this XAsync operation.
  - listener - An optional parameter, which may be null, of the XASyncListener object to be notified when the operation has completed.
- Returns: XAsync object representing this operation.

- Throws, or returns via the XAsync object:
  - AuthorizationException - The application is not authorized to open XSets.
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - inXuid or mode are null, or the Xopid may already be in use.
  - XSetInaccessibleException - The specified XSet does not exist or is not accessible.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe. The XAsyncListener must be coded in a thread-safe manner.
- Blocking: This method will return immediately.

5.3.2.15 asyncCopyXSet

```
public org.snia.xam.XAsync asyncCopyXSet(
     XUID          inXuid,
     String        mode,
     long          xopid,
     XAsyncListener listener)
```

Copies the specified XSet to a new XSet. The new XSet will be "open" in the mode specified.

- Parameters:
  - inXuid - The XUID of the XSet to copy.
  - mode - The mode in which the XSet will be opened in. One of the following:
    - XSet.MODE_UNRESTRICTED
    - XSet.MODE_RESTRICTED
  - xopid - The application supplied XOPID of this XAsync operation.
  - listener - An optional parameter, which may be null, of the XASyncListener object to be notified when the operation has completed.
- Returns: XAsync object representing this operation.

- Throws, or returns via the XAsync object:
  - AuthorizationException - The application is not authorized to perform this operation.
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - inXuid or mode are null, or the Xopid id may already be in use.
  - XSetInaccessibleException - XSet does not exist or is not accessible.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe. The XAsyncListener must be coded in a thread-safe manner.
- Blocking: This method will return immediately.

5.3.3 XSet methods

public interface org.snia.xam.XSet implements org.snia.xam.FieldContainer

An XSet is the basic unit of storage for the XAM Storage System. XSet classes are implemented by storage system vendors and implement the XSystem interface. XSystem instances are created via the XSystem factory methods of openXSet, copyXSet, and createXSet.

Note: Changing the BINDING information on any persistently stored XSet field will result in a new XSet being created on commit. This is only TRUE when the XSet has been opened in UNRESTRICTED mode. Changes may alter the value of a BOUND field or change the BINDING attribute of a field.

5.3.3.1 applyAccessPolicy

```
public void applyAccessPolicy(
     boolean binding,
     String   policyName )
```

Creates or modifies a property field with the name of *.xset.access.policy* and a type set to ‘application/vnd.snia.xam.string’ on the XSet. Its value and binding attributes will be set according to the user-provided parameters. This field will be used by the XAM Storage System to determine the policies to use when accessing this XSet.

Note: If an access policy has not been applied to an XSet at the time of the initial commit, then the property will be created and set as the default access policy of the XSystem (i.e., the first string in *.xsystem.access.policy.list.*).

- Parameters:
  - binding - TRUE if this value is to be BOUND.
  - policyName - Name of the policy.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - MaximumFieldException - Too many fields in the XSet.
  - PolicyNameException - The policy name is null or invalid for this operation.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.2 resetAccessFields

```
public void resetAccessFields()
```

This method will remove all access fields from the XSet.

Note: If an access policy has not been applied to an XSet at the time of the initial commit, then the property will be created and set as the default access policy of the XSystem (i.e., the first string in *.xsystem.access.policy.list.*

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - ObjectInUseException -The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.3 applyManagementPolicy

```
public void applyManagementPolicy(
     boolean binding,
      String policyName)
```

Creates and sets the XAM string property *.xam.management.policy*. This field is used by the XAM Storage System to determine the default policies when managing the XSet. If the management policy has not been set at the time of first commit, then the property will be created and set as the default management policy of the XSystem. The default policy is the first string in the XSystem management policy list.

- Parameters:
  - binding - TRUE if this value is to be BOUND.
  - policyName - Name of the policy.
- Throws:
  - MaximumFieldException - Too many fields in the XSet.
  - PolicyNameException - The policy name is null or invalid for this operation.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.4 resetManagementFields

```
public void resetManagementFields()
```

Removes all management fields from the XSet. This includes *.xset.minimum.retention.starttime*. Because this is a binding field, a new XSet will be created on commit, if the XSet was opened in the unrestricted mode.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.5 createRetention

```
public void createRetention( boolean binding,
                             String retentionID )
```

Creates a scope for storing and evaluating retention criteria. It creates a field with the type of “application/vnd.snia.xam.string” and sets the value to the retention ID. The field name is formed by appending the retention id to the prefix *.xset.retention.list.*. The final form of the field name is *.xset.retention.list.<retentionID>*. The binding attribute of this field is set according to the parameter *binding.*

- Parameters:
  - binding - Sets the binding attribute of the field to TRUE or FALSE.
  - retentionID - A String containing the retention identifier of the retention being created.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - The parameter *retentionID* is null or malformed.
  - InvalidFieldNameException - The generated field name is too long.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.6 setRetentionEnabledFlag

```
public void setRetentionEnabledFlag( String retentionID,
                                     boolean binding,
                                     boolean enabled )
```

Enables or disables retention that is scoped by the specified retention id. This flag is stored in a field of type “application/vnd.snia.xam.boolean”. The name of the field is formed by inserting the retention id between a prefix (*.xset.retention.*) and a suffix (*.enabled*); thus, the final format of the name is *.xset.retention.<retention id>.enabled*. If the field does not exist, it will be created; otherwise, the value will be updated only if the value is changed from FALSE to TRUE; if the value is set to TRUE, it cannot be changed. It will have its binding attribute set according to the binding flag that is set by the application.

- Parameters:
  - retentionID - A String containing the retention identifier of the retention being enabled or disabled.
  - binding - Sets the binding attribute of the field to TRUE or FALSE.
  - enabled - Enables the retention if the parameter is TRUE or disables it unless it was already TRUE.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - The parameter *retentionID* is null or malformed.
  - InvalidFieldNameException - The generated field name is too long.
  - ObjectInUseException - The XSet has open import/export streams
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.7 applyRetentionEnabledPolicy

```
public void applyEventRetentionEnabledPolicy( String retentionID,
                                              boolean binding,
                                              String policyName )
```

Enables or disables retention that is scoped by the specified retention id, as determined by the named policy. The policy name of the policy holding the enabled flag is stored in a field of type “application/vnd.snia.xam.string”. The name of the field is formed by inserting the retention id between a prefix (*.xset.retention.*) and a suffix (*.enabled.policy*); thus, the final format of the name is *.xset.retention.<retention id>.enabled.policy*. If the field does not exist, it will be created; otherwise, the value will be updated only if the value is changed from FALSE to TRUE; if the value is set to TRUE, it cannot be changed. It will have its binding attribute set according to the binding flag that is set by the application.

Note: If the *.xset.retention.<retention id>.enabled* field is also present on the XSet, it will be used by the XAM Storage System in preference to this field.

- Parameters:
  - retentionID - A String containing the retention identifier of the retention being enabled or disabled.
  - binding - Sets the binding attribute of the field to TRUE or FALSE.
  - policyName - A String containing the name of the policy to be used.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - The parameter *retentionID* is null or malformed.
  - InvalidFieldNameException - The generated field name is too long.
  - PolicyNameException - The policy name is null or invalid for this operation.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.

  `Blocking: This method blocks until completion.`

5.3.3.8 setRetentionDuration

```
public void setRetentionDuration( String  retentionId,
                                  boolean binding,
                                  long    duration )
```

Sets the duration of retention that is scoped by the specified retention id. This flag is stored in a field of type “application/vnd.snia.xam.int”. The name of the field is formed by inserting the retention id between a prefix (*.xset.retention.*) and a suffix (*.duration*); thus, the final format of the name is *.xset.retention.<retention id>.duration*. If the field does not exist, it will be created; otherwise, the value will be updated only if the duration is increased. It will have its binding attribute set according to the binding flag that is set by the application.

- Parameters:
  - retentionID - A String containing the retention identifier of the retention being enabled or disabled.
  - binding - Sets the binding attribute of the field to TRUE or FALSE.
  - duration - a long value indicating the duration (in milliseconds) to be used for this retention. Zero indicates no retention, while negative one (-1) indicates infinite retention.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - The parameter *retentionID* is null or malformed.
  - InvalidFieldNameException - The generated field name is too long.
  - RetentionValueException - Thrown if the duration parameter specifies an illegal duration, such as a value which attempts to shorten the existing duration value.
  - ObjectInUseException- The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.9 applyRetentionDurationPolicy

```
public void applyRetentionDurationPolicy( String  retentionID,
                                          boolean binding,
                                          String  policyName )
```

Sets the duration of retention that is scoped by the specified retention id as specified by the named policy. This policy name is stored in a field of type “application/vnd.snia.xam.string”. The name of the field is formed by inserting the retention id between a prefix (*.xset.retention.*) and a suffix (*.duration.policy*); thus, the final format of the name is *.xset.retention.<retention id>.duration.policy*. If the field does not exist, it will be created; otherwise, the value will be updated only if the duration is increased. It will have its binding attribute set according to the binding flag that is set by the application.

Note: If the *.xset.retention.<retention id>.duration* field is also present on the XSet, it will be used by the XAM Storage System in preference to this field.

- Parameters:
  - retentionID - A String containing the retention identifier of the retention being enabled or disabled.
  - binding - Sets the binding attribute of the field to TRUE or FALSE.
  - policyName - A String containing the name of the policy to be used.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - The parameter *retentionID* is null or malformed.
  - InvalidFieldNameException - The generated field name is too long.
  - PolicyNameException - The policy name is null or invalid for this operation.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.10 setRetentionStarttime

```
public void setRetentionStarttime( String  retentionID,
                                   boolean binding )
```

Sets the start time of retention that is scoped by the specified retention id. The current time of the XSystem is stored in a field of type “application/vnd.snia.xam.datetime”. The name of the field is formed by inserting the retention id between a prefix (*.xset.retention.*) and a suffix (*.starttime*); thus, the final format of the name is *.xset.retention.<retention id>.starttime*. If the field does not exist, it will be created. If the field does exist, an exception will be thrown, since the retention start time is not allowed to be changed, once set. The field will have its binding attribute set according to the binding flag that is set by the application.

The start time for any specific retention ID may only be set once.

- Parameters:
  - retentionID - A String containing the retention identifier of the retention being enabled or disabled.
  - binding - Sets the binding attribute of the field to TRUE or FALSE.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - FieldExistsException -The property *.xset.retention.<retentionID>.starttime* exists, meaning that the start time has already been set for this retention ID.
  - InvalidArgumentException - The parameter *retentionID* is null or malformed.
  - InvalidFieldNameException - The generated field name is too long.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.11 setBaseRetention

```
public void setBaseRetention( boolean binding,
                              long    duration )
```

If this XSet does not already contain the field *.xset.retention.list.base*, this method will create the field with a type of “application/vnd.snia.xam.string” and set the value to “base”. It will also create the “application/vnd.snia.xam.boolean” field *.xset.retention.base.enabled* and set the value to TRUE. The duration will be stored in a field named *.xset.retention.base.duration*. This field is of type “application/vnd.snia.xam.int”. If the field already exists, its value will be changed to match the passed in duration only if the duration of the retention is not reduced; the method will generate an error if the duration is reduced. If the field does not already exist, it will be created with the specified duration as the value. These fields will have their binding attributes set according to the binding flag that is set by the application.

These fields will be used by the XAM Storage System to determine the base retention duration to use when managing this XSet.

Note: When an XSet instance containing the field *.xset.retention.list.base* is first committed, the field *.xset.retention.base.starttime* will be created and have its value set to *.xset.xuidtime*.

- Parameters:
  - binding - Sets the binding attribute of the field to TRUE or FALSE.
  - duration - a long value indicating the duration to be used for this retention. Zero indicates no retention, while negative one (-1) indicates infinite retention.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - RetentionValueException - Thrown if the duration parameter specifies an illegal duration, such as a value which attempts to shorten the existing duration value.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.12 applyBaseRetentionPolicy

```
public void applyBaseRetentionPolicy( boolean binding,
                                      String  policyName )
```

If this XSet does not already contain the field *.xset.retention.list.base*, this method will create the field with a type of “application/vnd.snia.xam.string” and set the value to “base”. It will also create the “application/vnd.snia.xam.boolean” field *.xset.retention.base.enabled* and set the value to TRUE. The duration policy will be stored in a field named *.xset.retention.base.duration.policy*. This field is of type “application/vnd.snia.xam.string”. If the field already exists, its value will be changed to match the passed-in policy only if the policy would not reduce the duration of the retention; the method will generate an error if the policy reduces the duration. If the field does not already exist, it will be created with the specified policy name as the value. These fields will have their binding attributes set according to the binding flag that is set by the application.

These fields will be used by the XAM Storage System to determine the base retention duration to use when managing this XSet.

Note: If the *.xset.retention.base.duration* field is also present on the XSet, it will be used by the XAM Storage System in preference to this policy field.

Note: When an XSet instance containing the field *.xset.retention.list.base* is first committed, the field *.xset.retention.base.starttime* will be created and have its value set to *.xset.xuidtime*.

- Parameters:
  - binding - Sets the binding attribute of the field to TRUE or FALSE.
  - policyName - A String containing the name of the policy to be used.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - The parameter *retentionID* is null or malformed.
  - InvalidFieldNameException - The generated field name is too long.
  - PolicyNameException - The policy name is null or invalid for this operation.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.13 setAutoDelete

```
public void setAutoDelete(
     boolean binding,
     boolean autoDelete)
```

If this XSet does not have auto delete set on it, this method will create a property field on the XSet with the name of *.xset.autodelete* with the type “application/vnd.snia.xam.boolean”. Its value and binding attributes will be set according to the user-provided parameters. If the field already exists on the XSet, then its value will be updated with the specified value. This field will be used by the XAM Storage System to determine if the XSet should be automatically deleted when retention expires.

- Parameters:
  - binding - TRUE if the property is BOUND.
  - autoDelete - TRUE if the system should auto-delete the object on expiration, FALSE otherwise.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - MaximumFieldException - Too many fields in the XSet.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.

- Blocking: This method blocks until completion.

5.3.3.14 applyAutoDeletePolicy

```
public void applyAutoDeletePolicy(
     boolean binding,
     String policyName)
```

If this XSet does not have an auto delete policy applied to it, this method will create a property field on the specified XSet with the name of *.xset.autodelete.policy* with the type set to “application/vnd.snia.xam.string.” Its value and binding attributes will be set according to the user-provided parameters. If the field already exists on the XSet, then its value will be updated with the specified value. This field will be used by the XAM Storage System to determine if the XSet should be automatically deleted when retention expires.

Note: If the explicit *autodelete* field is present on the XSet, it will be used by the XAM Storage System in preference to this field.

- Parameters:
  - binding - TRUE if the property is BOUND.
  - policyName - Name of the policy.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - PolicyNameException - The policy name is null or invalid for this operation.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.15 setShred

```
public void setShred(
     boolean binding,
     boolean shred)
```

If this XSet does not have the shred property set, this method creates a property field on the XSet with the name *.xset.shred* with type “application/vnd.snia.xam.boolean”. Its value and binding attributes will be set according to the user-provided parameters. If the field already exists on the XSet, then its value will be updated with the specified value. This field will be used by the XAM Storage System to determine if the XSet should be shredded after deletion.

- Parameters:
  - binding - TRUE if the property is BOUND.
  - shred - TRUE if the XSet is to be shredded on delete, FALSE otherwise.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - MaximumFieldException - Too many fields in the XSet.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.16 applyShredPolicy

```
public void applyShredPolicy(
     boolean binding,
     String policyName)
```

If this XSet does not have an auto shred policy applied to it, this method will create a property field on the specified XSet with the name of *.xset.shred.policy* set to type “application/vnd.snia.xam.string”. Its value and binding attributes will be set according to the user-provided parameters. If the field already exists on the XSet, then its value will be updated with the specified value. This field will be used by the XAM Storage System to determine if the XSet should be shredded after XSet deletion. If *.xset.shred* is also present on the XSet, it will be used by the XAM Storage System in preference to this field.

- Parameters:
  - binding - TRUE if the property is BOUND.
  - policyName - Name of the policy.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - MaximumFieldException - Too many fields in the XSet.
  - PolicyNameException - The policy name is null or invalid for this operation.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.17 applyStoragePolicy

```
public void applyStoragePolicy(
     boolean binding,
     String policyName)
```

If this XSet does not have a storage policy applied to it, this method will create a property field on the XSet with the name of *.xset.storage.policy* with a type of “application/vnd.snia.xam.string”. Its value and binding attributes will be set according to the user-provided parameters. If the field already exists on the XSet, then its value will be updated with the specified value. This field will be used by the XAM Storage System to determine the storage policy of the XSet.

- Parameters:
  - binding - TRUE if the property is BOUND.
  - policyName - Name of the policy.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - MaximumFieldException - Too many fields in the XSet.
  - PolicyNameException - The policy name is invalid for this operation.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.18 getActualRetentionDuration

```
public long getActualRetentionDuration( String retentionID )
```

Evaluates all management settings for this XSet to return the effective retention duration.

- Parameters: retentionID - A String containing the name of the retention being interrogated.
- Returns: The effective event retention duration for this XSet.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - The retention ID is null or malformed.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.19 getActualRetentionEnabled

```
public boolean getActualRetentionEnabled( String retentionID )
```

Evaluates all management settings for this XSet to return the effective event retention.

- Parameters: retentionID - A String containing the name of the retention being interrogated.
- Returns: The effective enabled state for the retention specified on this XSet.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - The retention ID is null or malformed.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.20 getActualAutoDelete

```
public boolean getActualAutoDelete()
```

Evaluates all management settings for this XSet to return the effective auto delete settings for this XSet.

- Returns: Effective auto delete settings.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.21 getActualShred

```
public boolean getActualShred()
```

Evaluates all management settings for this XSet to return the effective auto shred settings for this XSet.

- Returns: Effective auto shred settings.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.22 commit

```
public org.snia.xam.XUID commit()
```

Stores an XSet in the XSystem. Note that this does not close the XSet, which can still be modified as allowed by the authorization of the XSystem and/or the mode with which the XSet was created. A XUID will be assigned by the XAM Storage System, and this XUID will be returned.

Open XStreams will not cause the commit to fail. Only the data that was successfully written to such XSteams will be committed.

If this is a modified XSet (e.g., an existing XSet was opened and changed), then a new XUID may or may not be assigned by the commit, according to the following rules:

- If only nonbinding fields are edited (created, deleted, or changed), then the XAM Storage System may not assign a new XUID.
- If any binding fields are edited (created, deleted, or changed), then the XAM Storage System must assign a new XUID.

In any case, an application should be coded to handle cases where the XUID changes when a modified XSet is committed.

If a management policy has not been applied to the XSet before commit, a default management policy will be applied to the XSet at the time of commit.

- Returns: The XUID of the stored XSet.
- Throws:
  - AuthorizationException - The application is not authorized to perform this operation.
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - JobRunningException - This XSet is running a job and the XSystem does not support saving running jobs.
  - XSetUnderHoldException - The XSet is being held and may not be changed.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.23 close

```
public void close()
```

Closes the XSet, making it unavailable for further use. Any resources used by the open XSet will be released. After calling this method, the XSet instance may not be reused.

Note: This call will fail if there are any open XStreams associated with this XSet.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.24 abandon

```
public void abandon()
```

Allows an XSet to be closed after it has been corrupted. This method will allow the XSet to be closed without regard to any open XStreams. After calling this method, the only usable method is XSet.close; all other XSet operations will fail unsaved XSet changes.

CAUTION: This method should only be used on a corrupt XSet. Calling it at other inappropriate times can result in data loss. Applications are encouraged to track open XSets and close them appropriately.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.25 submitJob

```
public void submitJob()
```

Submits a job request to the XAM Storage System. Fields in the XSet are evaluated as input to the job according to the XAM job control subsystem.

- Throws:
  - AuthorizationException - The application is not authorized to perform this operation.
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - JobCommandException - One or more of the required job fields are not present in the XSet.
  - JobUnsupportedException - The specified job is not supported by the XSystem.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.26 haltJob

```
public void haltJob()
```

If this XSet is a running job, halts the job and fills in the appropriate fields for the completion of the job.

- Throws:
  - AuthorizationException - The application is not authorized to perform this operation.
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.27 openExportXStream

```
public org.snia.xam.XStream openExportXStream()
```

Opening a new XStream contains the normalized export format for the specified XSet. The XSet must have been committed and the instance must have been unchanged since opening. The application may read the XStream using the normal XStream methods. The application must close this XStream so that the XSet can be used for other XSet operations.

Note: The XSet cannot be modified while the export XStream is open.

- Returns: The XStream containing the XSet export format.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.28 openImportXStream

```
public org.snia.xam.XStream openImportXStream()
```

Opens an import stream that the application will use to write the normalized import format for the XSet. The XSet must have been created using either the RESTRICTED or UNRESTRICTED mode. When the import stream has been completely written and closed, the XAM Storage System will validate the data from the import stream. If the data has been found to be invalid or improperly formatted, the XSet will enter the corrupt state, and only the abandon method will succeed. After a successful import, the XSet may be committed via the normal commit semantics, and the returned XUID is the value described in the normalized import stream. Any modification to binding fields or binding attributes in the XSet will result in a new XSet and new XUID value.

- Returns: The XStream the application will use to write the canonical export format data.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.29 asyncCommit

```
public org.snia.xam.XAsync asyncCommit(
    long          xopid,
    XAsyncListener listener)
```

This method is an asynchronous version of the XSet.commit method.

- Parameters:
  - xopid - The application XOPID of this XAsync operation.
  - listener - An optional parameter, which may be null, of the XASyncListener object to be notified when the operation has completed.
- Returns: XAsync object representing this operation.
- Throws, or returns via the XAsync object:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - Xopid id may already be in use.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe. The XAsyncListener must be coded in a thread-safe manner.
- Blocking: This method will return immediately.

5.3.4 Field container methods

public interface org.snia.xam.FieldContainer

Details methods common to all first class XAM objects that have a property field associated with them.

5.3.4.1 openFieldIterator

```
public java.util.Iterator openFieldIterator( String prefix )
```

Iterates over a set of fields belonging to the primary XAM object. These iterators yield String objects, the name of the field selected. The filter pattern is a simple, regular expression that is limited to a prefix of character literals. To find all fields beginning with "com.example" make the call:

```
openFieldIterator( "com.example" );
```

- Parameters: prefix - A prefix string describing the fields to be returned by the iterator. The expression is simple: a series of byte values which are treated as a "prefix" to match. For example "com.example" will match all fields starting with "com.example".

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldNameException - prefix is null or malformed.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.2 containsField

```
public boolean containsField( String fieldName )
```

Returns a boolean value of TRUE, indicating if the named field is contained within the referenced FieldContainer object (XAM Library, XSystem, or XSet); returns FALSE in all other cases/

- Parameters: fieldName - The name of the field to test.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldName - fieldName is null or malformed.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.3 createProperty - xam_boolean

```
public void createProperty(
     String fieldName,
    boolean binding,
     boolean value)
```

Create a XAM Boolean valued property. XAM Boolean is Boolean in Java.

- Parameters:
  - fieldName - The name of this property.
  - binding - TRUE if this property is to be BOUND.
  - value - The Boolean value for this property.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldName - fieldName is null or malformed.
  - FieldExistsException - The specified field already exists.
  - InvalidOperationException - An attempt to create a bound field on a XAMLibrary or XSystem.
  - MaximumFieldException - Too many fields in the XSet.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.4 createProperty - xam_int

```
public void createProperty(
     String fieldName,
     boolean binding,
     long value)
```

Creates a XAM Int valued property. XAM Int is long in Java.

- Parameters:
  - fieldName - The name of this property.
  - binding - TRUE if this property is to be BOUND.
  - value - The long value for this property.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldNameException - fieldName is null or malformed.
  - FieldExistsException - The specified field already exists.
  - InvalidOperationException - An attempt to create a bound field on a XAMLibrary or XSystem.
  - MaximumFieldException - Too many fields in the XSet.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.5 createProperty - xam_double

```
public void createProperty(
     String fieldName,
     boolean binding,
     double value)
```

Create a XAM Float valued property. XAM Float is double in Java.

- Parameters:
  - fieldName - The name of this property.
  - binding - TRUE if this property is to be BOUND.
  - value - The double value for this property.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldNameException - fieldName is null or malformed.
  - FieldExistsException - The specified field already exists.
  - InvalidOperationException - An attempt to create a bound field on a XAMLibrary or XSystem.
  - MaximumFieldException - Too many fields in the XSet.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.6 createProperty - xam_xuid

```
public void createProperty(
     String fieldName,
     boolean binding,
     XUID value)
```

Creates a XUID valued property.

- Parameters:
  - fieldName - The name of the property.
  - binding - TRUE if the property is to be BOUND
  - value - The XUID value of the property.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - value is null or malformed.
  - InvalidFieldNameException - fieldName or malformed.
  - InvalidOperationException - An attempt to create a bound field on a XAMLibrary or XSystem.
  - FieldExistsException - The specified field already exists.
  - MaximumFieldException - Too many fields in the XSet.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.7 createProperty - xam_string

```
public void createProperty(
     String fieldName,
     boolean binding,
     String value)
```

Creates a XAM string valued property. Note that Java strings may be much longer than XAM strings. If the Java string is longer than the XAM MAX String, an exception will be thrown.

- Parameters:
  - fieldName - The name of the property.
  - binding - TRUE if the property is to be BOUND.
  - value - The string value of the property.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - value is null or an invalid UTF-8 string.
  - InvalidFieldNameException - fieldName is null or malformed.
  - InvalidOperationException - An attempt to create a bound field on a XAMLibrary or XSystem.
  - FieldExistsException - The specified field already exists.
  - MaximumFieldException - Too many fields in the XSet.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.8 createProperty - xam_datetime

```
public void createProperty(
     String fieldName,
     boolean binding,
     Calendar value)
```

Creates a XAM Date valued property. XAM Date is java.util.Calendar in Java.

- Parameters:
  - fieldName - The name of the property.
  - binding - TRUE if the property is to be BOUND.
  - value - The date and time value of the property.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - value is null or a non valid ISO-8601 date.
  - InvalidFieldNameException - fieldName is null or malformed.
  - InvalidOperationException - An attempt to create a bound field on a XAM Library or XSystem.
  - FieldExistsException - The specified field already exists.
  - MaximumFieldException - Too many fields in the XSet.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.9 setProperty - xam_boolean

```
public void setProperty(
     String  fieldName,
     boolean value)
```

Sets the named field to the specified Boolean value.

- Parameters:
  - fieldName - The name of the field to set.
  - value - The Boolean value to set

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldNameException - fieldName is null or malformed.
  - InvalidFieldTypeException - The field is not a xam_boolean stype.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.10 setProperty - xam_datetime

```
public void setProperty(
     String fieldName,
     Calendar value)
```

Sets the named field to the specified Date/Time.

- Parameters:
  - fieldName - The name of the field to set.
  - value - The Date/Time value to set.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - value is null.
  - InvalidFieldNameException - fieldName is null or malformed.
  - InvalidFieldTypeException - The field is not a xam_datetime stype.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.11 setProperty - xam_double

```
public void setProperty(
     String fieldName,
     double value)
```

Sets the named field to the specified XAM Float.

- Parameters: fieldName - The name of the field to set.
- Throws:
  - InvalidFieldNameException - fieldName is null or malformed.
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldTypeException - The field is not a xam_double stype.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.

- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.12 setProperty - xam_int

```
public void setProperty(
     String fieldName,
     long value)
```

Sets the named field to the specified XAM Int.

- Parameters:
  - fieldName - The name of the field to set.
  - value - The XAM Int value to set.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldNameException - fieldName is null or malformed.
  - InvalidFieldTypeException - The field is not a xam_int stype.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.13 setProperty - xam_string

```
public void setProperty(
     String fieldName,
     String value)
```

Sets the named field to the specified String.

- Parameters:
  - fieldName - The name of the field to set.
  - value - The string value to set. String length must not exceed the maximum XAM string limit.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - value is null or a non-valid UTF-8 string.
  - InvalidFieldNameException - fieldName is null or malformed.
  - InvalidFieldTypeException - The field is not a xam_string stype.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.14 setProperty - xam_xuid

```
public void setProperty(
     String fieldName,
     XUID value)
```

Sets the named field to the specified XUID.

- Parameters:
  - fieldName - The name of the field to set.
  - value - The XUID value to set.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - value is null.
  - InvalidFieldNameException - fieldName is null or malformed.
  - InvalidFieldTypeException - The field is not a xam_xuid stype.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.15 getBoolean - xam_boolean

```
public boolean getBoolean(
     String fieldName)
```

Gets the Boolean value of the named field.

- Parameters: fieldName - The name of the property to get the value.
- Returns: The Boolean value of the named field.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldNameException - fieldName is null or malformed.
  - InvalidFieldTypeException - The field is not a xam_boolean stype.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.16 getDatetime - xam_datetime

```
public java.util.Calendar getDateTime(
     String fieldName)
```

Gets the DateTime value of the named field.

- Parameters: fieldName - The name of the property to get the value.
- Returns: The Calendar instance of the named field.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldNameException - fieldName is null or malformed.
  - InvalidFieldTypeException - The field is not a xam_datetime stype.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.

- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.17 getDouble - xam_double

```
public double getDouble(
     String fieldName)
```

Gets the XAM Float value of the named field.

- Parameters: fieldName - The name of the property to get the value.
- Returns: The double value of the named field.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldNameException - fieldName is null or malformed.
  - InvalidFieldTypeException - The field is not a xam_double stype.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.18 getLong - xam_int

```
public long getLong(
     String fieldName)
```

Gets the XAM Int value of the named field.

- Parameters: fieldName - The name of the property to get the value.
- Returns: The Java long value of the named field.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldNameException - fieldName is null or malformed.
  - InvalidFieldTypeException - The field is not a xam_int stype.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.19 getString - xam_string

```
public java.lang.String getString(
     String fieldName)
```

Gets the XAM string value of the named field.

- Parameters: fieldName - The name of the property to get the value.
- Returns: The Java string value of the named field.
- Throws:
  - InvalidFieldNameException - fieldName is null or malformed.
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldTypeException - The field is not a xam_string stype.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.

- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.20 getXUID - xam_xuid

```
public org.snia.xam.XUID getXUID(
     String fieldName)
```

Gets the XUID value of the named field.

- Parameters: fieldName - The name of the property to get the value.
- Returns: The XUID value of the named field.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldNameException - fieldName is null or malformed.
  - InvalidFieldTypeException - The field is not a xam_xuid stype.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.21 createXStream

```
public org.snia.xam.XStream createXStream(
     String fieldName,
     boolean binding,
     String mimeType)
```

Creates a new XStream-valued field in the XSet. The new stream will be opened in WRITE-Truncate mode, with a zero length value.

- Parameters:
  - fieldName - The name of the new XStream.
  - binding - TRUE if this field is to be BOUND.
  - mimeType - The mime type of the new XStream.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - mimeType is null or malformed.
  - InvalidFieldNameException - fieldName is null or malformed.
  - InvalidOperationException - An attempt to create a bound field on this object is not supported.
  - FieldExistsException - The specified field already exists.
  - MaximumFieldException - Too many fields in the XSet.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.22 openXStream

```
public org.snia.xam.XStream openXStream(
     String fieldName,
     String mode)
```

Opens the named XStream for read or write. If the XStream has not been opened, the data manipulation methods will throw an exception.

- Parameters:
  - fieldName - The name of the XStream to be opened.
  - mode - A mode value indicating if the XStream should be opened MODE_READ_ONLY, MODE_WRITE_TRUNCATE, MODE_WRITE_APPEND.
- Returns: The XStream containing the data.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldNameException - fieldName is null or malformed.
  - FieldDoesNotExistException - The specified field does not exist.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.23 getFieldType

```
public java.lang.String getFieldType(
     String fieldName)
```

Returns the MIME type of the specified field.

- Parameters: fieldName - Name of the field from which to get the type.
- Returns The string value of the field's type.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldNameException - fieldName is null or malformed.
  - FieldDoesNotExistException - The specified field does not exist.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.

- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.24 getFieldLength

```
public long getFieldLength(
     String fieldName)
```

Returns the length of the field, in bytes.

- Parameters: fieldName - Name of the field from which to get the length.
- Returns: The number of bytes for the length of the field.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldNameException - fieldName is null or malformed.
  - FieldDoesNotExistException - The specified field does not exist.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.25 getFieldBinding

```
public boolean getFieldBinding(
     String fieldName)
```

Returns TRUE if the field is bound, FALSE otherwise.

- Parameters: fieldName - Name of the field from which to get the binding info.
- Returns
  - The Boolean value of the binding mode of the field.
  - The XStream containing the data.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldNameException - fieldName is null or malformed.
  - FieldDoesNotExistException - The specified field does not exist.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.26 getFieldReadOnly

```
public boolean getFieldReadOnly(String fieldName)
```

Returns TRUE if the readonly attribute of the field is TRUE; returns FALSE otherwise.

- Parameters: fieldName - Name of the field from which to get the readonly info.
- Returns: The Boolean value of the readonly mode of the property.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldNameException - fieldName is null or malformed.
  - FieldDoesNotExistException - The specified field does not exist.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.

- Blocking: This method blocks until completion.

5.3.4.27 setFieldAsBinding

```
public void setFieldAsBinding(
     String fieldName)
```

Sets the field binding attribute to TRUE. This method will only work on fields contained in an XSet.

- Parameters: fieldName - Name of the field to set the binding value.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldNameException - fieldName is null or malformed.
  - InvalidOperationException - The XAMLibrary or XSystem FieldContainer do not support binding fields.
  - FieldDoesNotExistException - The specified field does not exist.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.28 setFieldAsNonbinding

```
public void setFieldAsNonbinding(
     String fieldName)
```

Sets the field binding attribute to FALSE. This method will only work on fields contained in an XSet.

- Parameters:
  - fieldName - Name of the field to set the binding value.
  - The XStream containing the data.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldNameException - fieldName is null or malformed.
  - FieldDoesNotExistException - The specified field does not exist.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.4.29 deleteField

```
public void deleteField(
     String fieldName)
```

Removes the named field from the field container. This method applies only to XSets.

- Parameters: fieldName - The name of the field to remove.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldNameException - fieldName is null or malformed.
  - FieldDoesNotExistException - The specified field does not exist.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.

- Blocking: This method blocks until completion.

5.3.4.30 asyncOpenXStream

```
public org.snia.xam.XAsync asyncOpenXStream(
     String         fieldName,
     String         mode,
     long           xopid,
     XAsyncListener listener)
```

This method will begin the asynchronous opening of XStream in either readonly, writeonly or appendonly mode, based on the mode argument. The specified callback will be invoked as part of the asynchronous opening. To monitor the status of this operation, the application can poll the Async instance that is generated by this method. A handle to an Async instance is also passed to any provided callback method when that callback method is invoked.

- Parameters:
  - fieldName - The name of the XStream to be opened.
  - mode - The mode the XSet should be opened in. One of the following:
    - XStream.MODE_READ_ONLY
    - XStream.MODE_WRITE_TRUNCATE
    - XStream.MODE_WRITE_APPEND
  - xopid - The application XOPID of this XAsync operation.
  - listener - An optional parameter, which may be null, of the XASyncListener object to be notified when the operation has completed.
- Returns: XAsync object representing this operation.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - mode is null, or xopid is already in use.
  - InvalidFieldNameException - fieldName is null or malformed.
  - FieldDoesNotExistException - The specified field does not exist.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe. The XAsyncListener must be coded in a thread-safe manner.

- Blocking: This method will return immediately.

5.3.5 XStream methods

public interface org.snia.xam.XStream

XStreams are the interface to the content portions of a XAM Storage System. XStream implementations are provided by a storage vendor in the vendor's packages. XStream instances perform the storage vendor-specific functionality to interact with that storage vendor's XAM Storage System.

5.3.5.1 tell

```
public long tell()
```

Returns the position of the position indicator in the XStream.

- Returns: The cursor position.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - XStreamAbandonException - The XStream is in the abandoned state and may only be closed.
  - XStreamCorruptException - The XStream is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion. public class org.snia.xam.

5.3.5.2 seek

```
public void seek(
     long offset,
     long whence)
```

Seeks to a specific location within the XStream.

- Parameters:
  - offset - The number bytes of offset from the cursor position (whence).
  - whence - One of SEEK_CUR, SEEK_SET, SEEK_END.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - whence is invalid.
  - XStreamAbandonException - The XStream is in the abandoned state and may only be closed.
  - XStreamCorruptException - The XStream is in the corrupt state and may only be abandoned.
  - XStreamException - Attempt to seek outside boundaries of the stream.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion. public class org.snia.xam.

5.3.5.3 write

```
public long write(
     byte[] buffer,
     long   offset,
     long    bytesToWrite)
```

Transfers byte values to the XStream. This method will not block if a partial write occurred. Multiple calls to this method may be required to completely transfer the buffer.

- Parameters:
  - buffer - The array of bytes to be transferred to the XStream.
  - offset - The byte offset in the buffer from which to start writing.
  - bytesToWrite - The number of bytes to write.
- Returns: The number of bytes written to the stream.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - buffer is null, or size mismatch between arguments.
  - XStreamAbandonException - The XStream is in the abandoned state and may only be closed.
  - XStreamCorruptException - The XStream is in the corrupt state and may only be abandoned.
  - XStreamException - Maximum stream size has been exceeded.
  - XSetUnderHoldException - The XSet is being held and may not be changed.
  - XStreamException - The stream in the wrong open mode, or in a corrupt, close, or abandoned state.
  - XAMException - Another error exists with the underlying XAM Storage System.

- Thread Safety: This method is thread safe.
- Blocking: This method blocks until all data is completely written, but it will indicate the amount of data that was written in each call. Subsequent calls to this method may be required to write the buffer.

5.3.5.4 write

```
public long write( byte[] buffer
                   long   bytesToWrite )
```

Transfers byte values to the XStream. This method will not block if a partial write occurred. Multiple calls to this method may be required to completely transfer the buffer.

- Parameters:
  - buffer - The array of bytes to be transferred to the XStream.
  - bytesToWrite - The number of bytes, beginning at the start of the buffer, to transfer to the XStream.
- Returns: The number of bytes written to the stream.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - buffer is null, or size mismatch between arguments.
  - XStreamAbandonException - The XStream is in the abandoned state and may only be closed.
  - XStreamCorruptException - The XStream is in the corrupt state and may only be abandoned.
  - XStreamException - Maximum stream size has been exceeded.
  - XSetUnderHoldException - The XSet is being held and may not be changed.
  - XStreamException - The stream in the wrong open mode, or in a corrupt, close, or abandoned state.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until all data is completely written, but it will indicate the amount of data that was written in each call. Subsequent calls to this method may be required to write the buffer.

5.3.5.5 write

```
public long write( byte[] buffer )
```

Transfers byte values to the XStream. This method will not block if a partial write occurred. Multiple calls to this method may be required to completely transfer the buffer.

- Parameters: buffer - The array of bytes to be transferred to the XStream.

- Returns: The number of bytes written to the stream.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - buffer is null, or size mismatch between arguments.
  - XStreamAbandonException - The XStream is in the abandoned state and may only be closed.
  - XStreamCorruptException - The XStream is in the corrupt state and may only be abandoned.
  - XStreamException - Maximum stream size has been exceeded.
  - XSetUnderHoldException - The XSet is being held and may not be changed.
  - XStreamException - The stream in the wrong open mode, or in a corrupt, close, or abandoned state.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method does block until all data is completely written, but it will indicate the amount of data that was written in each call. Subsequent calls to this method may be required to write the buffer.

5.3.5.6 read

```
public long read(
     byte[] buffer,
     long offset,
     long bytesToRead)
```

Transfers byte values from the XStream to buffer. If there is not enough data left in the stream to fill the buffer, that data is copied to the buffer, leaving the remaining bytes unchanged. This method does not block if there are insufficient bytes to fill the buffer. Subsequent calls may be required to read the remaining data in the XStream.

- Parameters:
  - buffer - An array of byte values to read.
  - offset - The byte offset in the buffer to start placing bytes read from the stream.
  - bytesToRead - The maximum number of bytes to read.
- Returns: The actual number of bytes read.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - buffer is null, or size mismatch between arguments.
  - XStreamAbandonException - The XStream is in the abandoned state and may only be closed.
  - XStreamCorruptException - The XStream is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until all data is completely read, but it will indicate the amount of data that was read in each call. Subsequent calls to this method may be required to read the remainder of the data.

5.3.5.7 read

```
public long read(
     byte[] buffer,
     long bytesToRead)
```

Transfers byte values from the XStream to buffer. If there is not enough data left in the stream to fill the buffer, that data is copied to the buffer, leaving the remaining bytes unchanged. This method does not block if there are insufficient bytes to fill the buffer. Subsequent calls may be required to read the remaining data in the XStream.

- Parameters:
  - buffer - An array of byte values to read.
  - bytesToRead - The maximum number of bytes to read.
- Returns: The actual number of bytes read.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - buffer is null, or size mismatch between arguments.
  - XStreamAbandonException - The XStream is in the abandoned state and may only be closed.
  - XStreamCorruptException - The XStream is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe
- Blocking: This method does block until all data is completely read, but it will indicate the amount of data that was read in each call. Subsequent calls to this method may be required to read the remainder of the data.

5.3.5.8 read

```
public long read( byte[] buffer )
```

Transfers byte values from the XStream to buffer. If there is not enough data left in the stream to fill the buffer, that data is copied to the buffer, leaving the remaining bytes unchanged. This method does not block if there are insufficient bytes to fill the buffer. Subsequent calls may be required to read the remaining data in the XStream.

- Parameters: buffer - An array of byte values to read.
- Returns: The actual number of bytes read.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - buffer is null, or there is a size mismatch between arguments.
  - XStreamAbandonException - The XStream is in the abandoned state and may only be closed.
  - XStreamCorruptException - The XStream is in the corrupt state and may only be abandoned.
  - XStreamException - The stream is in a corrupt, close, or abandoned state.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method does block until all data is completely read, but it will indicate the amount of data that was read in each call. Subsequent calls to this method may be required to read the remainder of the data.

5.3.5.9 close

```
public void close()
```

Closes the XStream, making this XStream instance unavailable for further use. The XStream needs to be reopened to access this stream data.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - XSetUnderHoldException - The XSet is being held and may not be changed.
  - XStreamCorruptException - The XStream is in the corrupt state and may only be abandoned.
  - XStreamException - The stream was not open.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.5.10 abandon

```
public void abandon()
```

Abandons (loses) all pending work on this XStream, allowing it to be closed. This method is to be used when the XStream is in the corrupt state and the application needs to perform error recovery and clean up.

CAUTION: Calling abandon may result in data loss if used inappropriately.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.5.11 asyncRead

```
public org.snia.xam.XAsync asyncRead(
     byte[]         buffer,
     long           bufferLength,
     long           xopid,
     XAsyncListener listener)
```

Begins the asynchronous transfer of data from the storage system into the target buffer, up to the number of bytes requested. The specified callback will be invoked as part of the asynchronous transfer. To monitor the status of this operation, the application can poll the Async instance that is generated by this method. A handle to an Async instance is also passed to any provided callback method, when that callback method is invoked.

- Parameters:
  - buffer - The byte buffer into which the date is read.
  - bufferLength - The number of bytes to read. This must be less than or equal to the buffer length.
  - xopid - The application XOPID of this XAsync operation.
  - listener - An optional parameter, which may be null, of the XASyncListener object to be notified when the operation has completed.
- Returns: XAsync object representing this operation.

- Throws, or returns via the XAsync object:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - buffer is null, or xopid is already in use.
  - XStreamAbandonException - The XStream is in the abandoned state and may only be closed.
  - XStreamCorruptException - The XStream is in the corrupt state and may only be abandoned.
  - XStreamException - The stream is open in the wrong mode (write only).
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe. The XAsyncListener must be coded in a thread-safe manner.
- Blocking: This method will return immediately.

5.3.5.12 asyncWrite

```
public org.snia.xam.XAsync asyncWrite(
     byte[]         buffer,
     long           bufferLength,
     long           xopid,
     XAsyncListener listener)
```

Begins the asynchronous transfer of data from the source buffer to the XAM Storage System, up to the number of bytes requested. The specified callback will be invoked as part of the asynchronous transfer. To monitor the status of this operation, the application can poll the Async instance that is generated by this method. A handle to an Async instance is also passed to any provided callback method, when that callback method is invoked.

Note: This method may fail with an error if the maximum number of bytes supported in an XStream is reached. To determine the actual maximum number of bytes allowed in an XStream, an application should evaluate *.xsystem.limits.maxSizeOfXStream* on the XSystem instance. For more information on this topic, please consult [XAM-ARCH].

- Parameters:
  - buffer - A byte array containing the data to be written to the stream.
  - bufferLength - The number of bytes to be written from the buffer. This value must be less than or equal to the actual buffer length.
  - xopid - The application XOPID of this XAsync operation.
  - listener - An optional parameter, which may be null, of the XASyncListener object to be notified when the operation has completed.
- Returns: XAsync object representing this operation.

- Throws, or returns via the XAsync object:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - buffer is null, or xopid is already in use.
  - XStreamAbandonException - The XStream is in the abandoned state and may only be closed.
  - XStreamCorruptException - The XStream is in the corrupt state and may only be abandoned.
  - XStreamException - The stream is open in the wrong mode (write only).
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe. The XAsyncListener must be coded in a thread-safe manner.
- Blocking: This method will return immediately.

5.3.5.13 asyncClose

```
public XAsync asyncClose(
     long          xopid,
     XAsyncListener listener)
```

Begins the asynchronous closing of a previously opened XStream. Any resources that were allocated can be released at this point. The specified callback will be invoked as part of the asynchronous close. To monitor the status of this operation, the application can poll the Async instance that is generated by this method. A handle to an Async instance is also passed to any provided callback method when that callback method is invoked.

- Parameters:
  - xopid - The application XOPID of this XAsync operation.
  - listener - An optional parameter, which may be null, of the XASyncListener object to be notified when the operation has completed.
- Returns: XAsync object representing this operation.
- Throws, or returns via the XAsync object:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - xopid is already in use.
  - XStreamCorruptException - The XStream is in the corrupt state and may only be abandoned.
  - XStreamException - The stream is not open.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe. The XAsyncListener must be coded in a thread-safe manner.
- Blocking: This method will return immediately.

5.3.6 XAsync methods

5.3.6.1 halt

```
public void halt()
```

Stops the execution of the operations associated with the XAsync instance. This method may be used at any time.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.6.2 isComplete

```
public boolean isComplete()
```

Asynchronous XAM methods take, as an optional argument, an object which implements XAsyncListener. The method XAsyncCallback will be called when the operation is completed (either successfully or unsuccessfully).

- Returns: TRUE if the operation has completed; otherwise, it will return FALSE.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.6.3 getXOPID

```
public void getXOPID()
```

Retrieves the XOPID of the operations associated with the XAsync instance. This method may be used at any time.

- Returns: The long XOPID value of the XAsync.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - XAMException - Another error exists with the underlying XAM Storage System.

- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.6.4 getStatus

```
public void getStatus()
```

Retrieves the XAM status of the operation associated with the XAsync instance. This method may be used after the operation has transitioned to the completed state.

- Returns: Nothing. If an error occurred due to the XAsync operation, that exception will be thrown; otherwise, the method simply returns.
- Throws: Always throws the exception generated by the asynchronous operation. If no exception was generated by the operation, this method simply returns.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.6.5 getXSet

```
public XSet getXSet()
```

Retrieves the XSet reference associated with this XAsync instance. This is only valid if the operation was not an “asyncOpenXSet” or “asyncCopyXSet”.

- Returns: The XSet reference from the XAsync.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.6.6 getXStream

```
public XStream getXStream()
```

Retrieves the XStream reference associated with this XAsync instance. This is only valid if the operation was not an “asyncOpenXStream”.

- Returns: The XStream reference from the XAsync.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - XAMException - Another error exists with the underlying XAM Storage System.

- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.6.7 getXUID

```
public XUID getXUID()
```

Retrieves the XUID reference (if any) of the XSet associated with this XAsync instance. This method may be used at any time.

- Returns: The XUID reference.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.6.8 getBytesWritten

```
public long getBytesWritten()
```

Retrieves the number of bytes written by the operation associated with this XAsync instance. Not all operations write bytes, and for those operations, it will always be set to zero. This method may be used at any time.

- Returns: The number of bytes written.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.6.9 getBytesRead

```
public long getBytesWritten()
```

Retrieves the number of bytes read by the operation associated with the XAsync instance. Not all operations read bytes, and for those operations, it will always be set to zero. This method may be used at any time.

- Returns: The number of bytes read.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.6.10 close

```
public void close()
```

Releases the resources of the operation associated with the XAsync instance and the resources of XAsync operation itself. This method may be used after the operation has transitioned to the completed state.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.7 XAsyncListener methods

5.3.7.1 XAsyncCallback

```
public void XAsyncCallback( XAsync asyncState )
```

Asynchronous XAM methods take, as an optional argument, an object which implements XAsyncListener. The method XAsyncCallback will be called when the operation is completed (either successfully or unsuccessfully).

- Throws: This method shall not throw any exceptions.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.8 XUID methods

public interface org.snia.xam.XUID

These methods represent the methods performed on a XAM XUID.

5.3.8.1 toBytes

```
public byte[] toBytes()
```

Gets the raw array of bytes representing this XUID.

- Returns: The byte array.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.8.2 toString

```
public java.lang.String toString()
```

Converts this XUID to a base64-encoded String.

- Returns: The base64 version of this XUID.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.8.3 equals

```
public boolean equals(
     Object x)
```

Compares two XUID values.

- Returns: TRUE if the two values are the same.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.9 XIterator methods

public interface org.snia.xam.XIterator

These methods represent the methods performed on a XAM XIterator.

5.3.9.1 next

```
public String next()
```

Retrieves the next field name (a string) from the XIterator.

- Returns: The name of the field selected by the XIterator.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.9.2 hasNext

```
public boolean hasNext()
```

Indicates if the XIterator has more elements to retrieve.

- Returns: TRUE if more elements remain in the XIterator.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.9.3 remove

```
public void remove()
```

This method is required by the java.util.Iterator interface. XAM does not support the removal of items in the XIterator.

- Returns: Never. This method will always throw an exception.
- Throws: java.lang.UnsupportedOperationException, always.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.9.4 close

```
public boolean close()
```

Closes the XIterator and releases any resource associated with the XIterator.

- Returns: TRUE if the two values are the same.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.10 XAM exceptions

All XAMExceptions shall subclass from org.snia.xam.XAMException. Exceptions are thrown when any erroneous condition occurs during execution of a XAM method. Applications may catch or pass exceptions according to the normal Java exception processing. Applications are encouraged to handle exceptions rather than passing them, as some exceptions may result in unusable XAM object instances (e.g., XSystems or XSets in a corrupt state).

Constructors

XAM Exceptions follow all of the standard rules for Java exceptions. The exception to this rule is to integrate the XAM status code into the exception (see the exception method *getStatusCode*). To facilitate integration with C-based VIMs, all XAM exceptions shall provide a status code and shall provide a constructor to set the XAM Status codes. Generic constructors *may* be used in non-specific circumstances, but the error code is not detailed enough to provide useful information to programmers. XAM implementations and VIMs are encouraged to supply all exceptions with specific and unique error codes.

The following examples show the recommended constructors.

```
public <ExceptionName>(String message, long status)

public <ExceptionName>(String message, Throwable cause, long status)
```

Methods

5.3.10.1 XAMException - Constructor

```
public XAMException()
```

This basic constructor is provided for compatibility with the Exception superclass. The XAMException generated with this constructor shall be a nonspecific error and shall supply a generic status code.

- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.10.2 XAMException - Constructor

```
public XAMException(String arg0)
```

Provides the ability to specify the message (XAM error token). The XAMException generated with this constructor shall provide a generic error code and the specific error token.

- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.10.3 XAMException - Constructor

```
public XAMException(String arg0)
```

Provides the ability to specify the message (XAM error token). The XAMException generated with this constructor shall provide a generic error code and the specific error token.

- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.10.4 XAMException - Constructor

```
public XAMException(String arg0, long status)
```

Provides the ability to specify the message (XAM error token). The XAMException generated with this constructor shall provide a the specific error token and the specified status code.

- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.10.5 XAMException - Constructor

```
public XAMException(Throwable arg0)
```

Provides compatibility with the Java Exception superclass. The resulting XAMException represents a non-specific XAM error which wraps another Java Throwable. The exception shall contain a nonspecific error code.

- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.10.6 XAMException - Constructor

```
public XAMException(String arg0, Throwable arg1, long status)
```

Provides compatibility with the Java Exception superclass. The resulting XAMException represents a XAMException with a specific error token and specific error code.

- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.10.7 getStatusCode

```
public long getStatusCode();
```

Allows an application to retrieve the status code of the XAMException.

- Returns: The status code of the exception.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.10.8 getMessage

```
public String getMessage()
```

Allows an application to retrieve the error token for the XAMException.

- Returns: The error token message of the exception.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.11 XAM Specific Exception Classes

All of the exceptions described in Table 3 extend org.snia.xam.XAMException. Each exception subclass shall provide two constructors that allow VIMs to create these exceptions with and without underlying Throwable causes. Java XAM implementations are encouraged to coordinate error token and status code values with other implementations.

All XAMExceptions and subclasses shall be in the *org.snia.xam* package. This section specifies all of the standard exceptions supported by Java XAM implementations. Vendor-specific exceptions, thrown from the VIM, shall extend XAMException or extend one of the standard XAM exceptions. VIM vendors are encouraged to follow the exception object model specified in this section.

Table 3 lists all of the defined XAMException subclasses. Exceptions that subclass from XAMException may be superclasses for other exception collections or generalized exception conditions that may occur in a wide variety of situations.

Table 3 – Exceptions that extend XAMException

This exception...	Is thrown when...
AsyncHaltedException	an asynchronous operation has been programmatically halted. All XAsynch methods, except getStatus, shall throw this exception when an XAsync has been halted.
AsyncPendingException	an asynchronous operation is pending on this object.

Table 3 – Exceptions that extend XAMException

This exception...	Is thrown when...
AuthenticationException	an error occurred during the authentication process with the XSystem.
AuthenticationExpiredException	the current authentication has expired and the application needs to reauthenticate in order to continue using the XSystem.
AuthorizationException	the application is not authorized for an operation.
FieldContainerException	an unspecified error occurred for the FieldContainer operation. It is the superclass of all FieldContainer exceptions.
InsufficientResourcesException	the XSystem does not have enough resources to complete the operation.
InvalidArgumentException	arguments to a method are null or violate a constraint of the executed method.
InvalidOperationException	when an application attempts to create a bound field on a XAMLibrary or XSystem instance.
InvalidXUIDException	an improperly formed XUID or an XUID failing the CRC check is used.
JobException	an unspecified error occurred for the submitted job. It is the superclass of all Job exceptions.
ObjectInUseException	the operation failed because the object is in use elsewhere.
XSetException	an unspecified error occurred for the XSet operation. It is the superclass of all XSet exceptions.
XStreamException	an unspecified error occurred for the XStream operation. It is the superclass of all XStream exceptions.
XSystemException	an unspecified error occurred for the XSystem operation. It is the superclass of all XSystem exceptions.

Table 4 lists all of the exceptions associated with FieldContainer-specific methods. All of the exceptions listed in Table 4 shall extend the class *org.snia.xam.FieldContainerException*. Exception conditions not listed in this table may be encoded as a generic FieldContainerException. Applications must check the exceptions status code and message for more details.

Table 4 – Exceptions that extend FieldContainerException

This exception...	Is thrown when...
FieldDoesNotExistException	attempting to access a field which does not exist in a field container.
FieldExistsException	trying to create a field which already exists in the field container. To create a new field in the container, delete the field and then create the new field.
FieldInUseException	trying to modify a field that is currently in use elsewhere.
FieldReadOnlyException	attempting to modify a readonly field.
InvalidFieldNameException	a field name being used is not valid.
InvalidFieldTypeException	an improperly formed or illegal MIME type is specified.
MaximumFieldException	attempting to create more fields than are currently supported by the XSystem.

Table 5 lists all exceptions associated with job-specific methods. All of the exceptions listed in Table 5 shall extend the class *org.snia.xam.JobException*. Exception conditions not listed in this table may be encoded as a generic JobException. Applications must check the exceptions status code and message for more details.

Table 5 – Exceptions that extend JobException

This exception...	Is thrown when...
JobCommandException	some of the required job fields are not created before executing XSet.submit.
JobPermissionsException	the authorized application does not have sufficient permission to submit a job.
JobResourceException	the XSystem does not have sufficient resources to execute the job.
JobRunningException	the job specified by the submitted XSet is already running.
JobUnsupportedException	the submitted XSet specifies a job not supported on the XSystem.
QueryException	an error occurred while preparing a query job.

Table 6 lists all exceptions associated with XSet-specific methods. All of the exceptions listed in Table 6 shall extend the class *org.snia.xam.XSetException*. Exception conditions not listed in this table may be encoded as a generic XSetException. Applications must check the exceptions status code and message for more details.

Table 6 – Exceptions that extend XSetException

This exception...	Is thrown when...
HoldIdException	a malformed hold id is used, or a hold id has previously been used on the XSet.
InvalidXSetModeException	the mode specified in the operation is not recognized by the XSystem.
PolicyNameException	the policy name used in the operation is either not allowed or is invalid in this context.
RetentionValueException	applying an illegal retention duration to an XSet. This is typically thrown when the effective retention duration would be shortened using the specified value.
XSetInaccessibleException	the XSet specified in the operation does not exist, or the authenticated application is not allowed to access the XSet.
XSetUnderHoldException	attempting to modify an XSet which has one or more holds applied.
XSetUnderRetentionException	attempting to delete an XSet which is still being protected by retention.
XSetAbandonException	the XSet is in the abandoned state and can only be closed.
XSetCorruptException	the XSet is in the corrupt state and cannot be used.

Table 7 lists all exceptions associated with XStream-specific methods. All of the exceptions listed in Table 7 shall extend the class *org.snia.xam.XStreamException*. Exception conditions not listed in this table

may be encoded as a generic XStreamException. Applications must check the exceptions status code and message for more details.

Table 7 – Exceptions that extend XStreamException

This exception...	Is thrown when...
EndOfStreamException	attempting to read past the end of available data in an XStream.
InvalidXStreamModeException	the mode specified to open an XStream is invalid.
XStreamAbandonException	the XStream is in the abandoned state and may only be closed.
XStreamCorruptException	the XStream is in a corrupt state and cannot be used.

Table 8 lists all exceptions associated with XSystem-specific methods. All of the exceptions listed in Table 8 shall extend the class *org.snia.xam.XSystemException*. Exception conditions not listed in this table may be encoded as a generic XSystemException. Applications must check the exceptions status code and message for more details.

Table 8 – Exceptions that extend XSystemException

This exception...	Is thrown when...
ConnectException	an error occurred during an attempt to connect to an XSystem.
InvalidXRIException	the XRI used to connect is improperly formed.
VIMLoadException	an error occurs when loading a VIM. This exception may be thrown during connect but could also be thrown at other times.
XSystemCorruptException	the XSystem is in a corrupt state and cannot be used. The application should abandon the XSystem and then close it.
XSystemAbandonException	the XSystem is in the abandoned state and may only be closed.

5.4 Interface constant fields

The following constants are defined as fields in the specified interfaces. These constants should be used by application programmers whenever possible to increase application portability.

5.4.1 org.snia.xam.XAMLibrary Fields

Table 9 describes the programming constants defined in the XAMLibrary interface.

Table 9 – XAMLibrary Constants

Field Name	Description
public static final TEXT_PLAIN_MIME_TYPE	This is the ‘text/plain’ MIME for general .TXT XStreams. Applications may need to add a character set to the MIME type before stream creation.
public static final XUID_LIST_MIME_TYPE	The MIME type of the query result stream.
public static final STYPE_BOOLEAN_MIME_TYPE	The MIME type of XAM_BOOLEAN stypes.
public static final STYPE_INT_MIME_TYPE	The MIME type of XAM_INT stypes.

Table 9 – XAMLibrary Constants

Field Name	Description
public static final STYPE_DOUBLE_MIME_TYPE	The MIME type of xam_DOUBLE stypes.
public static final STYPE_STRING_MIME_TYPE	The MIME type of XAM_STRING stypes.
public static final STYPE_DATETIME_MIME_TYPE	The MIME type of XAM_DATETIME stypes.
public static final STYPE_XUID_MIME_TYPE	The MIME type of XAM_XUID stypes
public static final XAM_PROTOCOL	XAM XRI strings must begin with this protocol designator.
public static final XAM_VIM_LIBRARY_TOKEN	This character separates the VIM name from the XAM Storage System name.
public static final DEFAULT_MIME_TYPE	Applications may use this MIME type for data not defined by any other appropriate MIME type.
public static final MIME_XUID_LIST_TYPE	The MIME type of a list of fixed 80-byte records containing XUID values.
public static final XAM_IDENTITY	This XAM_STRING property indicates the origin of the XAM Library. It is intended for debugging, and applications should not code specific behavior with respect to this value.
public static final XAM_LOG_LEVEL	This XAM_STRING property is used to indicate the current level of normal library logging. The higher the value, the more detail is logged. Applications may set this value to control the log.
public static final XAM_LOG_PATH	This XAM_STRING property is used to indicate the path used by the XAMLibrary for logging.
public static final XAM_LOG_VERBOSITY	This XAM_STRING property is used to indicate the current level of debug library logging. The higher the value, the more detail is logged. Applications may set this value to control the log.
public static int XAM_LOG_ALL	This integer value is used in setting XAM_LOG_LEVEL to indicate that ALL messages are to be logged.
public static int XAM_LOG_INFO	This integer value is used in setting XAM_LOG_LEVEL to indicate that INFO or more severe messages are to be logged.
public static int XAM_LOG_WARN	This integer value is used in setting XAM_LOG_LEVEL to indicate that WARN or more severe messages are to be logged.
public static int XAM_LOG_ERROR	This integer value is used in setting XAM_LOG_LEVEL to indicate that ERROR or more severe messages are to be logged.
public static int XAM_LOG_FATAL	This integer value is used in setting XAM_LOG_LEVEL to indicate that FATAL or more severe messages are to be logged.
public static int XAM_LOG_NONE	This integer value is used in setting XAM_LOG_LEVEL to indicate that NO messages are to be logged.

Table 9 – XAMLibrary Constants

Field Name	Description
public static final XAM_API_LEVEL	This XAM_STRING property is used to indicate which version of the XAM API is implemented.
public static final XAM_VIM_LIST	This prefix is for the properties, in a XAM Library, listing the names of the available VIMs.
public static final XAM_MAX_STRING	The maximum number of characters allowed in a XAM_STRING field.

5.4.2 org.snia.xam.XSystem Fields

Table 10 describes the programming constants defined in the XSystem interface.

Table 10 – XSystem constants

Field Name	Description
public static final READ_OK	Read access is OK. Used in accessXSet.
public static final WRITE_OK	Write access is OK. Used in accessXSet.
public static final WRITE_SYS_OK	Write access to system fields is OK. Used in accessXSet.
public static final DELETE_OK	Delete access is OK. Used in accessXSet.
public static final HOLD_OK	Hold/release access is OK. Used in accessXSet.
public static final EVENT_OK	Event retention access is OK. Used in accessXSet.
public static final XAM_XSYSTEM_ACCESS_POLICY_LIST	This string is a prefix of the list of access policy names available to the application. This string will be passed to the openXIterator method.
public static final XAM_XSYSTEM_IDENTITY	This XAM string property has the vendor identity of the connected XSystem instance.
public static final XAM_XSYSTEM_TIME	This XAM date property has the current time of the connected XSystem instance. This value is advisory for application.
public static final XAM_XSYSTEM_LIMITS_MAX_FIELDS	This XAM int property (Java long) indicates the maximum number of fields that may be created in an XSet.
public static final XAM_XSYSTEM_MAX_STREAM_SIZE	This XAM int property (Java long) indicates the maximum size of an XStream.
public static final XAM_XSYSTEM_AUTH_SASL_LIST	This property name prefix may be used to get the list of SASL supported authentication mechanisms that are supported by the connected XSystem.
public static final XAM_XSYSTEM_AUTH_SASL_DEFAULT	This XAM string property has the name of default SASL mechanism for the connected XSystem.
public static final XAM_XSYSTEM_AUTH_GRANULE_LIST	This prefix of the auth granule list is to be used in an XIterator pattern.

Table 10 – XSystem constants

Field Name	Description
public static final XAM_XSYSTEM_AUTH_IDENTITY_AUTHENTICATION	This XAM string property contains the value of the SASL authentication within the scope of the current XAM session. See the Security chapter of [XAM-ARCH].
public static final XAM_XSYSTEM_AUTH_IDENTITY_AUTHORIZATION	This XAM string property contains the value of the SASL authorization within the scope of the current XAM session. See the Security chapter of [XAM-ARCH].
public static final XAM_XSYSTEM_AUTH_EXPIRATION	This XAM int property indicates the number of seconds remaining before the XSystem will require a reauthentication. This value is estimated and should be treated as a hint. A value of -1 (negative one) shall mean that the expiration estimate is infinite, or that no re-authentication is required.
public static final XAM_XSYSTEM_ACCESS	This XAM Boolean property indicates if the XSystem will support the XSet access control policy.
public static final XAM_XSYSTEM_ACCESS_POLICY_LIST	Prefix of the access policy list, to be used in an XIterator pattern.
public static final XAM_XSYSTEM_JOB_COMMIT_SUPPORTED	This XAM Boolean property indicates if the XAM Storage System is able to commit running XAM jobs.
public static final XAM_XSYSTEM_JOB_LIST	This prefix of the job list is to be used in an XIterator pattern.
public static final XAM_XSYSTEM_JOB_LIST_QUERY	This XAM string property contains the job code for the XAM query job, ‘xam.job.query’.
public static final XAM_XSYSTEM_JOB_QUERY_CONTINUANCE_SUPPORTED	This XAM Boolean property indicates if the XAM Storage System will continue to run committed running jobs.
public static final XAM_XSYSTEM_JOB_QUERY_LEVEL1_SUPPORTED	This XAM Boolean property indicates if the XAM Storage System supports XAM Query Level 1 syntax and functionality. This value is TRUE on all XAM-compliant systems.
public static final XAM_XSYSTEM_JOB_QUERY_LEVEL2_SUPPORTED	This XAM Boolean property indicates if the XAM Storage System supports XAM Query Level 2 syntax and functionality. This functionality is optional for XAM Storage Systems.
public static final XAM_RETENTION_DURATION_POLICY_LIST	This prefix of the retention policy list is to be used in an XIterator pattern.
public static final XAM_RETENTION_ENABLED_POLICY_LIST	This prefix of the retention enabled policy list is to be used in an XIterator pattern.
public static final XAM_DELETION_AUTODELETE	This XAM Boolean property indicates if the XAM Storage System supports setting the XSet expiration autodelete functionality.
public static final XAM_DELETION_AUTODELETE_POLICY_LIST	This prefix of a list of autodelete policy names is to be used in an XIterator pattern.
public static final XAM_DELETION_SHRED	This XAM Boolean property indicates if the XAM Storage System supports setting the XSet shred functionality.

Table 10 – XSystem constants

Field Name	Description
public static final XAM_DELETION_SHRED_POLICY_LIST	This prefix of a list of shred policy names is to be used in an XIterator pattern.
public static final XAM_STORAGE_POLICY_LIST	This prefix of the storage policy list is to be used in an XIterator pattern.
public static final XAM_MANAGEMENT_POLICY_LIST	This prefix of the management policy list is to be used in an XIterator pattern.
public static final XAM_MANAGEMENT_POLICY_DEFAULT	This XAM string policy contains the default XSet management policy name.

5.4.3 org.snia.xam.XSet Fields

Table 11 describes the programming constants defined in the XSet interface.

Table 11 – XSet Constants

Field Name	Description
public static final MODE_READ_ONLY	Opens an XSet in READONLY mode. Only reading of field operations is permitted on an XSet opened in this mode. All attempts to commit will fail. Modification attempts should fail as early as possible.
public static final MODE_RESTRICTED	Opens an XSet in RESTRICTED mode. Reading of all fields is allowed and modifications of UNBOUND fields is permitted. Commiting an XSet with a BINDING modification will fail, and attempts to modify a BOUND field or change the BINDING attribute of a field should fail as soon as possible.
public static final MODE_UNRESTRICTED	Opens an XSet in UNRESTRICTED mode. This mode allows reading and modification of all fields. Changing the BINDING attribute or changing the value of a BOUND field will result in a the creation of a new XSet with a new XUID value on commit.
public static final XAM_TIME_CREATION	The xam_datetime property indicating when the XSet was created. This time is earlier than the commit time.
public static final XAM_TIME_COMMIT	The xam_datetime property indicating when the XSet was persistently stored in the XSystem.
public static final XAM_TIME_XUID	The xam_datetime property indicating when the XUID was assigned to the XSet.
public static final XAM_TIME_MODIFICATION	The xam_datetime property indicating when the XSet was last modified.
public static final XAM_TIME_ACCESS	The xam_datetime property indicating when the XSet was last accessed.
public static final XAM_TIME_RESIDENCY	The xam_datetime property indicating when the XSet was first stored in this XAM Storage System.
public static final XAM_XUID	The xam_xuid property containing the XUID for the specified XSet.

Table 11 – XSet Constants

Field Name	Description
public static final XAM_DIRTY	The xam_boolean property indicating if the specified XSet has been modified and not committed.
public static final XAM_MANAGEMENT_POLICY	The policy property indicating which policy is to be used for retention, deletion, and storage behavior in the absence of both value and policy management properties.
public final static XAM_RETENTION_LIST	Prefix of the retention list. Used by XIterators to obtain the list of retention IDs applied to this XSet.
public final static XAM_RETENTION_LIST_BASE	Prefix of the retention properties for the **base** retention ID. Used by XIterators to obtain a list of the retention settings.
public final static XAM_RETENTION_LIST_EVENT	Prefix of the retention properties for the **event** retention ID. Used by XIterators to obtain a list of the retention settings.
public final static XAM_RETENTION_BASE_STARTTIME	The xam_datetime property containing the start time of the **base** retention ID.
public final static XAM_RETENTION_BASE_DURATION	The xam_int property containing the duration, in milliseconds, of the **base** retention ID.
public final static XAM_RETENTION_BASE_DURATION_POLICY	The xam_string property containing the name of the duration policy, if applied, for the **base** retention ID.
public final static XAM_RETENTION_EVENT_STARTTIME	The xam_datetime property containing the start time of the **event** retention ID.
public final static XAM_RETENTION_EVENT_DURATION	The xam_int property containing the duration, in milliseconds, of the **base** retention ID.
public final static XAM_RETENTION_EVENT_DURATION_POLICY	The xam_string property containing the name of the duration policy, if applied, for the **event** retention ID.
public final static XAM_RETENTION_EVENT_ENABLED	The xam_boolean property containing the enabled status of the **event** retention ID.
public final static XAM_RETENTION_EVENT_ENABLED_POLICY	The xam_string property containing the name of the event policy, if applied, for the **event** retention ID.
public final static XAM_HOLD	The xam_boolean property indicating if this XSet is currently under hold.
public static final XAM_AUTODELETE_POLICY	The policy property indicating if autodelete is to be enabled for this XSet in the absence of the autodelete property.
public static final XAM_DELETION_AUTODELETE	The xam_boolean property indicating if the XSet is to be deleted automatically when the document expires.
public static final XAM_SHRED_POLICY	The policy property indicating if shred is to be enabled for this XSet in the absence of the shred property.
public static final XAM_SHRED	The xam_boolean property indicating if the XSet is to be shredded once it is deleted.

Table 11 – XSet Constants

Field Name	Description
public static final XAM_STORAGE_POLICY	The policy property indicating how the XSet is to managed with respect to storage management capabilities.
public static final XAM_JOB_STATUS	The xam_string property indicating the runtime status of a job.
public static final XAM_JOB_STATUS_NEW	The string value for the property XAM_JOB_STATUS indicating that the job is submitted but not yet started by the system.
public static final XAM_JOB_STATUS_STARTING	The string value for the property XAM_JOB_STATUS indicating that the job is being initialized and is not yet running.
public static final XAM_JOB_STATUS_RUNNING	The string value for the property XAM_JOB_STATUS indicating that the job is running.
public static final XAM_JOB_STATUS_SHUTTING_DOWN	The string value for the property XAM_JOB_STATUS indicating that the job is in the process of stopping for some reason.
public static final XAM_JOB_STATUS_COMPLETE	The string value for the property XAM_JOB_STATUS indicating that the job is no longer running and has run to the end.
public static final XAM_JOB_STATUS_SUSPENDED	The string value for the property XAM_JOB_STATUS indicating that the job has temporarily ceased processing, but it will resume.
public static final XAM_JOB_STATUS_HALTED	The string value for the property XAM_JOB_STATUS indicating that the job has been stopped by an application action.
public static final XAM_JOB_STATUS_KILLED	The string value for the property XAM_JOB_STATUS indicating that the job has been stopped by a system action.
public static final XAM_JOB_ERROR_LEVEL_NOT_SUPPORTED	The string value for the property XAM_JOB_STATUS indicating that the query level specified was not supported by this XSystem.
public static final XAM_JOB_ERROR_INVALID_COMMAND_SYNTAX	The string value for the property XAM_JOB_STATUS indicating that the job command has a syntax error which prevents it from executing.
public static final XAM_JOB_ERROR_INSUFFICIENT_PERMISSIONS	The string value for the property XAM_JOB_STATUS indicating that the authenticated application is not authorized to execute the job.
public static final XAM_JOB_ERROR_INSUFFICIENT_RESOURCES	The string value for the property XAM_JOB_STATUS indicating that the XSystem does not have enough resources to complete the job.
public static final XAM_JOB_ERRORHEALTH	The xam_string property which indicates if the job has encountered an error.
public static final XAM_JOB_ERRORHEALTH_OK	The string value for the property XAM_ERRORHEALTH which indicates that the job is “OK”.

Table 11 – XSet Constants

Field Name	Description
public static final XAM_JOB_ERRORHEALTH_ERROR	The string value for the property XAM_ERRORHEALTH which indicates that the job has encountered an error.
public static final XAM_JOB_ERROR	The xam_string property that indicates which error the job has encountered.
public static final XAM_JOB_QUERY	The string constant to be placed in the XAM_JOB_COMMAND property to create and run a XAM query job.
public static final XAM_JOB_COMMAND	The xam_string property indicating what type of job is to be executed.
public static final XAM_JOB_QUERY_COMMAND	The XStream field containing the actual query string for a XAM query job.
public static final XAM_JOB_QUERY_RESULTS	The XStream field containing the XUID results from a query job.
public static final XAM_JOB_QUERY_RESULTS_COUNT	The xam_int property containing the number of XUIDs contained in the XAM_JOB_QUERY_RESULTS stream.
public static final XAM_JOB_QUERY_LEVEL	The xam_string property indicating the level of the query, as determined by the XAM Storage System.

5.4.4 org.snia.xam.XStream Constants

Figure 12 lists the programming constants defined in the XStream interface.

Table 12 – XStream Constants

Field Name	Description
public static final SEEK_SET	Seek to offset from the beginning of the stream. Used in seek.
public static final SEEK_CUR	Seek to offset from the current cursor. Used in seek.
public static final SEEK_END	Seek to the offset from the end of the stream. Used in seek.
public static final MODE_READ_ONLY	Open mode readonly. No changes to the stream contents are allowed. Used in openXStream.
public static final MODE_WRITE_TRUNCATE	Open mode writeonly, truncate contents. This mode will erase all stream contents on openXStream. Used in openXStream.
public static final MODE_WRITE_APPEND	Open mode writeonly, append to contents. This mode preserves contents of the stream and the current write position is placed at the end of the stream. Used in openXStream.

6 Private (VIM) Java API Reference

6.1 VIM methods

```
public interface org.snia.xam.vim.VIM
```

This interface provides a series of methods allowing the XAM Library to communicate to VIMs. These methods aren't available to application programs, as VIM objects are not available. A VIM object implements the vendor-specific logic to connect to a particular XAM Storage System. The createXSystem method is used by the XAM Library to cause the VIM to perform vendor-specific initialization logic and provide the XSystem. The provided XSystem instance is implemented in the vendor's package space but implements the org.snia.xam.XSystem interface.

VIM authors should note that the VIM interface for the Java libraries is very different than the VIM interfaces required by the C implementation of a XAM Library. The reason for this is because of basic differences between polymorphism implemented in C and Java. The Java implementation of the XAM Library strongly leverages the use of polymorphism. When the application connects to a XAM Storage System, the VIM object returns an object that implements the XSystem interface. This object is *not* part of the XAM Library; it is completely storage vendor defined. This vendor object contains the code appropriate to interact with the storage vendor's system. Likewise, XSet instances are *not* part of the XAM Library. For more information, see Section 4.4, "VIM implementation models".

6.1.1 XSystem createXSystem

```
public org.snia.xam.XSystem createXSystem()
```

This factory method is used by the XAM Library to obtain a new, unconnected and unauthenticated XSystem. This XSystem instance will be initialized according to the XSystem initialization sequence detailed in the XAM Architecture Specification. For more information, see [XAM-ARCH].

- Throws: XAMException, if the VIM is unable to provide an XSystem instance.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until complete.

Annex A
(normative)
Public Interfaces

The following interfaces contain the Java code for implementing the methods described in Chapter 5, “Public Java API Reference”. Comments and Javadoc comments have been removed for brevity.

A.1 XAMLibrary.java

```
package org.snia.xam;

/**
 * A XAM System represents the XAM Library running locally. An application
 * wanting to use the XAM Library must create an instance of the XAM Library.
 * This library object will manage and create instances of VIMs in order to
 * connect to XAM Storage Systems. The XAM Library object may also contain
 * toolkit functions for convenience.
 */
public interface XAMLibrary extends FieldContainer
{
    public final static String STYPE_BOOLEAN_MIME_TYPE =
              "application/vnd.snia.xam.boolean";

    public final static String STYPE_INT_MIME_TYPE =
              "application/vnd.snia.xam.int";

    public final static String STYPE_DOUBLE_MIME_TYPE =
              "application/vnd.snia.xam.double";

    public final static String STYPE_STRING_MIME_TYPE =
              "application/vnd.snia.xam.string";

    public final static String STYPE_DATETIME_MIME_TYPE =
              "application/vnd.snia.xam.datetime";

    public final static String STYPE_XUID_MIME_TYPE =
              "application/vnd.snia.xam.xuid";

    public final static String TEXT_PLAIN_MIME_TYPE = "text/plain";

    public final static String XUID_LIST_MIME_TYPE =
              "application/vnd.snia.query.xuid_list";

    public final static String XAM_PROTOCOL = "snia-xam://";

    public final static String XAM_VIM_LIBRARY_TOKEN = "!";

    public static final String DEFAULT_MIME_TYPE = "application/octet-stream";

    // == XAM Library Properties ==
    public final static String XAM_IDENTITY = ".xam.identity";

    public final static String XAM_LOG_LEVEL = ".xam.log.level";
```

```
    public final static String XAM_LOG_VERBOSITY = ".xam.log.verbosity";

    public final static String XAM_LOG_PATH = ".xam.log.path";

    public final static String XAM_API_LEVEL = ".xam.apiLevel";

    public final static String XAM_VIM_LIST = ".xam.vim.list.";

    // Useful constants.
    public final static int XAM_MAX_STRING = 512;

    public final static long XAM_LOG_ALL   = 5;
    public final static long XAM_LOG_INFO  = 4;
    public final static long XAM_LOG_WARN  = 3;
    public final static long XAM_LOG_ERROR = 2;
    public final static long XAM_LOG_FATAL = 1;
    public final static long XAM_LOG_OFF   = 0;

    public XSystem connect(String XRI)
              throws InvalidXRIException, ConnectException,
                     VIMLoadException, XAMException;
}
```

A.2 XSystem.java

```
package org.snia.xam;

import java.util.Calendar;

/**
 * Represents a connection to a XAM Storage System. An XSystem instance has been
 * implemented in the storage vendor's packages. Storage vendor-specific XSystem
 * instances will perform their specified interactions with XAM Storage Systems
 * and act as factory sources for XSet and XStream objects (via
 * create/open XSet or XStream).
 */
public interface XSystem extends FieldContainer
{
    // The following bit fields are defined for accessXSet().
    public final static long ACCESS_READ_OK              = 0x80000000;
    public final static long ACCESS_WRITE_APPLICATION_OK = 0x40000000;
    public final static long ACCESS_WRITE_SYSTEM_OK      = 0x20000000;
    public final static long ACCESS_CREATE_OK            = 0x10000000;
    public final static long ACCESS_DELETE_OK            = 0x08000000;
    public final static long ACCESS_HOLD_OK              = 0x04000000;
    public final static long ACCESS_RETENTION_EVENT_OK   = 0x02000000;
    public final static long ACCESS_JOB_OK               = 0x01000000;
    public final static long ACCESS_JOB_COMMIT_OK        = 0x00800000;

    // == XSystem Properties ==
    public final String XAM_XSYSTEM_IDENTITY = ".xsystem.identity";

    public final String XAM_XSYSTEM_TIME = ".xsystem.time";

    public final String XAM_XSYSTEM_LIMITS_MAX_FIELDS =
```

```
            ".xsystem.limits.maxFieldsPerXSet";

    public final String XAM_XSYSTEM_LIMITS_MAX_STREAM_SIZE =
            ".xsystem.limits.maxSizeOfXStream";

    public final String XAM_XSYSTEM_AUTH_SASL_LIST =
            ".xsystem.auth.SASLmechanism.list.";

    public final String XAM_XSYSTEM_AUTH_SASL_DEFAULT =
            ".xsystem.auth.SASLmechanism.default";

    public final String XAM_XSYSTEM_AUTH_GRANULE_LIST =
            ".xsystem.auth.granule.list.";

    public final String XAM_XSYSTEM_AUTH_IDENTITY_AUTHENTICATION =
            ".xsystem.auth.identity.authentication";

    public final String XAM_XSYSTEM_AUTH_IDENTITY_AUTHORIZATION =
            ".xsystem.auth.identity.authorization";

    public final String XAM_XSYSTEM_AUTH_EXPIRATION = ".xsystem.auth.expiration";

    public final String XAM_SYSTEM_ACCESS = ".xsystem.access";

    public final String XAM_SYSTEM_ACCESS_POLICY_LIST =
           ".xsystem.access.policy.list.";

    public final String XAM_XSYSTEM_JOB_COMMIT_SUPPORTED =
            ".xsystem.job.commit.supported";

    public final String XAM_XSYSTEM_JOB_LIST = ".xsystem.job.list.";

    public final String XAM_XSYSTEM_JOB_LIST_QUERY =
          ".xsystem.job.list.xam.job.query";

    public final String XAM_XSYSTEM_JOB_QUERY_CONTINUANCE_SUPPORTED =
            ".xsystem.job.xam.job.query.continuance.supported";

    public final String XAM_XSYSTEM_JOB_QUERY_LEVEL2_SUPPORTED =
            ".xsystem.job.xam.job.query.level1.supported";

    public final String XAM_XSYSTEM_JOB_QUERY_LEVEL1_SUPPORTED =
            ".xsystem.job.xam.job.query.level2.supported";

    public final String XAM_RETENTION_DURATION_POLICY_LIST =
            ".xsystem.retention.duration.policy.list.";

    public final String XAM_RETENTION_ENABLED_POLICY_LIST =
            ".xsystem.retention.enabled.policy.list.";

    public final String XAM_DELETION_AUTODELETE = ".xsystem.deletion.autodelete";

    public final String XAM_AUTODELETE_POLICY_LIST =
            ".xsystem.deletion.autodelete.policy.list.";
```

```
public final String XAM_DELETION_SHRED = ".xsystem.deletion.shred";

public final String XAM_DELETION_SHRED_POLICY_LIST =
          ".xsystem.deletion.shred.policy.list.";

public final String XAM_STORAGE_POLICY_LIST =
           ".xsystem.storage.policy.list.";

public final String XAM_MANAGEMENT_POLICY_LIST =
           ".xsystem.management.policy.list.";

public final String XAM_MANAGEMENT_POLICY_DEFAULT =
           ".xsystem.management.policy.default";

// == XAM Storage System Management
void connect(String xri) throws XAMException;

public byte[] authenticate(byte[] buffer)
          throws AuthenticationException, InvalidArgumentException,
                 XAMException;

public void close()
          throws AuthenticationExpiredException, ObjectInUseException,
                 XAMException;

public void abandon() throws AuthenticationExpiredException, XAMException;

// == XSet management ==
public void deleteXSet(XUID xsetid)
          throws AuthorizationException, AuthenticationExpiredException,
                 XSetUnderRetentionException, XSetUnderHoldException,
                 InvalidArgumentException, XSetInaccessibleException,
                 XSystemAbandonException, XSystemCorruptException,
                 XAMException;

public boolean isXSetRetained( XUID xsetid )
          throws AuthenticationExpiredException, InvalidArgumentException,
                 XSetInaccessibleException, XSystemAbandonException,
                 XSystemCorruptException, XAMException;

public void holdXSet(XUID xsetid, String holdID)
          throws AuthenticationExpiredException, AuthorizationException,
                 InvalidFieldNameException, HoldIdException,
                 InvalidArgumentException, XSetInaccessibleException,
                 XSystemAbandonException, XSystemCorruptException,
                XAMException;

public void releaseXSet(XUID xsetid, String holdID)
          throws AuthenticationExpiredException,
                  AuthorizationException, HoldIdException,
                  InvalidArgumentException, XSetInaccessibleException,
                  XSystemAbandonException, XSystemCorruptException,
                 XAMException;

public boolean accessXSet(XUID xsetid, long mode)
```

```
            throws AuthenticationExpiredException, InvalidArgumentException,
                   XSystemAbandonException, XSystemCorruptException,
                   XAMException;

    public Calendar getXSetAccessTime(XUID xsetid)
            throws AuthenticationExpiredException, InvalidArgumentException,
                   XSetInaccessibleException, XSystemAbandonException,
                   XSetCorruptException, XAMException;

    public XSet createXSet(String createMode)
            throws AuthenticationExpiredException, AuthorizationException,
                   InvalidArgumentException, XSystemAbandonException,
                   XSystemCorruptException, XAMException;

    public XSet openXSet(XUID xsetid, String openMode) throws XAMException;

    public XSet copyXSet(XUID xsetid, String copyMode)
            throws AuthenticationExpiredException, AuthorizationException,
                   InvalidArgumentException, XSetInaccessibleException,
                   XSystemAbandonException, XSystemCorruptException
                   XAMException;

    // == Asynchronous Operations


    public XAsync asyncCopyXSet( XUID           inXuid,
                                 String         mode,
                                 long           xopid,
                                 XAsyncListener listener )
            throws AuthorizationException, AuthenticationExpirationException,
                   InvalidArgumentException, XSetInaccessibleException,
                   XSystemAbandonException, XSystemCorruptException,
                   XAMException;

    public XAsync asyncOpenXSet( XUID           inXuid,
                                 String         mode,
                                 long           xopid,
                                 XAsyncListener listener )
            throws AuthenticationExpiredException, AuthorizationException,
                   InvalidArgumentException, XSetInaccessibleException,
                   XSystemAbandonException, XSystemCorruptException,
                   XAMException;
}
```

A.3 XSet.java

```
package org.snia.xam;

/**
 * An XSet is the basic unit of storage for the XAM System. XSet instances are
 * implemented by storage vendors in the storage vendor's packages. These XSet
 * instances will perform storage vendor-specific interactions with the XAM
 * Storage System, as well as acting as a factory source for XStream objects
 * (via create/open methods).
 * NOTE: Changing the BINDING information on any persistently stored XSet field
```

```
 * will result in a new XSet being created on commit. This is only TRUE when
 * the XSet has been opened in UNRESTRICTED mode. Changes may be altering the
 * value of a BOUND field, or changing the BINDING attribute of a field.
 */
public interface XSet extends FieldContainer
{
    public final static String MODE_READ_ONLY = "readonly";

    public final static String MODE_RESTRICTED = "restricted";

    public final static String MODE_UNRESTRICTED = "unrestricted";

    // == Basic Built-in Fields ==
    public final static String XAM_TIME_CREATION = ".xset.time.creation";

    public final static String XAM_TIME_XUID = ".xset.time.xuid";

    public final static String XAM_TIME_COMMIT = ".xset.time.commit";

    public final static String XAM_TIME_ACCESS = ".xset.time.access";

    public final static String XAM_TIME_RESIDENCY = ".xset.time.residency";

    public final static String XAM_XUID = ".xset.xuid";

    public final static String XAM_DIRTY = ".xset.dirty";

    // == Management Fields ==/
    public final static String XAM_RETENTION_LIST = ".xset.retention.list.";

    public final static String XAM_RETENTION_LIST_BASE =
              ".xset.retention.list.base";

    public final static String XAM_RETENTION_LIST_EVENT =
              ".xset.retention.list.event";

    public final static String XAM_RETENTION_BASE_STARTTIME =
              ".xset.retention.base.starttime";

    public final static String XAM_RETENTION_BASE_ENABLED =
              ".xset.retention.base.enabled";

    public final static String XAM_RETENTION_BASE_DURATION =
              ".xset.retention.base.duration";

    public final static Strign XAM_RETENTION_BASE_ENABLED_POLICY =
              ".xset.retention.base.enabled.policy";

    public final static String XAM_RETENTION_BASE_DURATION_POLICY =
             ".xset.retention.base.duration.policy";

    public final static String XAM_RETENTION_EVENT_STARTTIME =
              ".xset.retention.event.starttime";

    public final static String XAM_RETENTION_EVENT_DURATION =
```

```
            ".xset.retention.event.duration";

public final static String XAM_RETENTION_EVENT_ENABLED =
            ".xset.retention.event.enabled";

public final static String XAM_RETENTION_EVENT_DURATION_POLICY =
            ".xset.retention.event.duration.policy";

public final static String XAM_RETENTION_EVENT_ENABLED_POLICY =
            ".xset.retention.event.enabled.policy";



public final static String XSET_HOLD = ".xset.hold";

public final static String XSET_HOLD_LIST = ".xset.hold.list.";

public final static String XAM_DELETION_AUTODELETE =
            ".xset.deletion.autodelete";

public final static String XAM_DELETION_AUTODELETE_POLICY =
             ".xset.deletion.autodelete.policy";

public final static String XAM_DELETION_SHRED_POLICY =
            ".xset.deletion.shred.policy";

public final static String XAM_DELETION_SHRED =
            ".xset.deletion.shred";

public final static String XAM_STORAGE_POLICY = ".xset.storage.policy";

public final static String XAM_MANAGEMENT_POLICY = ".xet.management.policy";

// == Job Properties/Values
public final static String XAM_JOB_STATUS = ".xam.job.status";

public final static String XAM_JOB_STATUS_NEW = "NEW";

public final static String XAM_JOB_STATUS_STARTING = "STARTING";

public final static String XAM_JOB_STATUS_RUNNING = "RUNNING";

public final static String XAM_JOB_STATUS_SHUTTING_DOWN = "SHUTTING DOWN";

public final static String XAM_JOB_STATUS_COMPLETE = "COMPLETE";

public final static String XAM_JOB_STATUS_SUSPENDED = "SUSPENDED";

public final static String XAM_JOB_STATUS_HALTED = "HALTED";

public final static String XAM_JOB_STATUS_KILLED = "KILLED";

public final static String XAM_JOB_ERRORHEALTH = ".xam.job.errorhealth";

public final static String XAM_JOB_ERRORHEALTH_OK = "OK";
```

```
public final static String XAM_JOB_ERRORHEALTH_ERROR = "ERROR";

public final static String XAM_JOB_ERROR = ".xam.job.error";

// == Query Job Fields ==
public final static String XAM_JOB_QUERY = "xam.job.query";

public final static String XAM_JOB_COMMAND = "org.snia.xam.job.command";

public final static String XAM_JOB_QUERY_COMMAND = "xam.job.query.command";

public final static String XAM_JOB_QUERY_RESULTS = "xam.job.query.results";

public final static String XAM_JOB_QUERY_RESULTS_COUNT =
          "xam.job.query.results.count";

public final static String XAM_JOB_QUERY_LEVEL = "xam.job.query.level";

public final static String XAM_JOB_ERROR_LEVEL_NOT_SUPPORTED =
    "xam.job.query::level_not_supported";

public static final String XAM_JOB_ERROR_INVALID_COMMAND_SYNTAX =
    "xam.job.query::invalid_command_syntax";

public static final String XAM_JOB_ERROR_INSUFFICIENT_PERMISSIONS =
    "xam.job.query::insufficient_permission";

public static final String XAM_JOB_ERROR_INSUFFICIENT_RESOURCES =
    "xam.job.query::insufficient_resources";

// == Data Management ==
public void applyAccessPolicy( boolean binding,
                               String  policyName )
        throws AuthenticationExpiredException, MaximumFieldException,
               PolicyNameException, ObjectInUseException,
               XSetAbandonException, XSetCorruptException, XAMException;

public void resetAccessFields()
          throws AuthenticationExpiredException, ObjectInUseException,
                 XSetAbandonException, XSetCorruptException, XAMException;

public void applyManagementPolicy(boolean binding, String policyName)
        throws MaximumFieldException, PolicyNameException,
               ObjectInUseException, XSetAbandonException,
               XSetCorruptException, XAMException;

public void resetManagementFields()
          throws AuthenticationExpiredException, ObjectInUseException,
                 XSetAbandonException, XSetCorruptException, XAMException;

public void createRetention( boolean binding,
                             String  retentionID )
        throws AuthenticationExpiredException, InvalidArgumentException,
               InvalidFieldNameException, ObjectInUseException,
```

```
                XSetAbandonException, XSetCorruptException, XAMException;

public void setRetentionEnabledFlag( String  retentionID,
                                     boolean binding,
                                     boolean enabled )
        throws AuthenticationExpiredException, InvalidArgumentException,
               InvalidFieldNameException, ObjectInUseException,
               XSetAbandonException, XSetCorruptException, XAMException;

public void applyRetentionEnabledPolicy( String  retentionID,
                                         boolean binding,
                                         String  policyName )
        throws AuthenticationExpiredException, InvalidArgumentException,
               InvalidFieldNameException, PolicyNameException,
               ObjectInUseException, XSetAbandonException,
               XSetCorruptException, XAMException;

public void setRetentionDuration( String  retentionID,
                                  boolean binding,
                                  long    duration )
        throws AuthenticationExpiredException, InvalidArgumentException,
               InvalidFieldNameException, RetentionValueException,
               ObjectInUseException, XSetAbandonException,
               XSetCorruptException, XAMException;

public void applyRetentionDurationPolicy( String  retentionID,
                                          boolean binding,
                                          String  policyName )
        throws AuthenticationExpiredException, InvalidArgumentException,
               InvalidFieldNameException, PolicyNameException,
               ObjectInUseException, XSetAbandonException,
               XSetCorruptException, XAMException;

public void setRetentionStarttime( String retentionID,
                                   boolean binding )
        throws AuthenticationExpiredException, FieldExistsException,
               InvalidArgumentException, InvalidFieldNameException,
               ObjectInUseException, XSetAbandonException,
               XSetCorruptException, XAMException;

public void setBaseRetention( boolean binding,
                              long    duration )
        throws AuthenticationExpiredException, RetentionValueException,
               ObjectInUseException, XSetAbandonException,
               XSetCorruptException, XAMException;

public void applyBaseRetentionPolicy( boolean binding,
                                      String  policyName )
        throws AuthenticationExpiredException, InvalidArgumentException,
               InvalidFieldNameException, PolicyNameException,
               ObjectInUseException, XSetABandonException,
               XSetCorruptException, XAMException;

public void setAutoDelete(boolean binding, boolean autoDelete)
          throws AuthenticationExpiredException, MaximumFieldException,
```

```
                ObjectInUseException, XSetAbandonException,
                XSetCorruptException, XAMException;

        public void applyAutoDeletePolicy(boolean binding, String policyName)
                throws AuthenticationExpiredException, PolicyNameException,
                        ObjectInUseException, XSetAbandonException,
                        XSetCorruptException, XAMException;

        public void setShred(boolean binding, boolean shred )
                throws AuthenticationExpiredException, MaximumFieldException,
                        ObjectInUseException, XSetAbandonException,
                        XSetCorruptException, XAMException;

        public void applyShredPolicy(boolean binding, String policyName0
                throws AuthenticationExpiredException, MaximumFieldException,
                        PolicyNameException, ObjectInUseException,
                        XSetAbandonException, XSetCorruptException, XAMException;

        public boolean getActualShred()
                throws AuthenticationExpiredException, ObjectInUseException,
                        XSetAbandonException, XSetCorruptException, XAMException;

        public void applyStoragePolicy(boolean binding, String policyName)
                throws AuthenticationExpiredException, MaximumFieldException,
                        PolicyNameException, ObjectInUseException,
                        XSetAbandonException, XSetCorruptException, XAMException;

        public long getActualRetentionDuration( String retentionID )
                throws AuthenticationExpiredException, InvalidArgumentException,
                        ObjectInUseException, XSetAbandonException,
                        XAMCorruptException, XAMException;

        public boolean getActualRetentionEnabled( String retentionID )
                throws AuthenticationExpiredException, InvalidArgumentException,
                        ObjectInUseException, XSetAbandonException,
                        XAMCorruptException, XAMException;

        public boolean getActualAutoDelete()
                throws AuthenticationExpiredException, ObjectInUseException,
                        XSetAbandonException, XSetCorruptException, XAMException;

        // == XSet Operations ==
        public XUID commit()
                throws AuthorizationExpiredException, AuthorizationException,
                        JobIsRunningException, XSetUnderHoldException,
                        ObjectInUseException, XSetAbandonException,
                        XSetCorruptException, XAMException;

        public void close()
                throws AuthenticationExpiredException, ObjectInUseException,
                        XSetCorruptException, XAMException;

        public void abandon()
                throws AuthenticationExpiredException, XAMException;
```

```
    // == Job Methods ==
    public void submitJob()
              throws AuthenticationExpiredException, AuthorizationException,
                     JobCommandException, JobUnsupportedException,
                     XSetAbandonException, XSetCorruptException, XAMException;

    public void haltJob()
              throws AuthorizationException, AuthenticationException,
                     ObjectInUseException, XSetAbandonException,
                     XSetCorruptException, XAMException;

    // == Import/Export ==
    public XStream openExportXStream()
              throws AuthenticationExpirationException, ObjectInUseException,
                     XSetAbandonException, XSetCorruptException, XAMException;

    public XStream openImportXStream()
              throws AuthenticationExpiredException, ObjectInUseException,
                     XSetAbandonException, XSetCorruptException, XAMException;

    // == Asynchronous Operations


    public XAsync asyncCommit( long xopid,
                               XAsyncListener listener )
              throws AuthenticationExpiredException, InvalidArgumentException,
                     ObjectInUseException, XSetAbandonException,
                     XSetCorruptException, XAMException;
}
```

A.4 FieldContainer.java

```
package org.snia.xam;

import java.util.Calendar;
import java.util.Iterator;

/**
 * Details methods common to all first class XAM Objects which have Property
 * Field associated with them.
 */
public interface FieldContainer
{
    // == XIterator ==

    XIterator openFieldIterator(String prefix)
              throws AuthenticationExpiredException, InvalidFieldNameException,
                     ObjectInUseException, XSetAbandonException,
                     XSetCorruptException, XSystemAbandonException,
                     XSystemCorruptException, XAMException;

    // == Field Interrogation ==
    public boolean containsField( String fieldName )
              throws AuthenticationExpiredException, InvalidFieldNameException,
                     ObjectInUseException, XSetAbandonException,
```

```
                    XSetCorruptException, XSystemAbandonException,
                    XSystemCorruptException, XAMException;

        // == Property creation ==
        public void createProperty(String fieldName, boolean binding, boolean value)
                  throws AuthenticationExpiredException, InvalidFieldNameException,
                         FieldExistsException, InvalidArgumentException,
                         MaximumFieldException, ObjectInUseException,
                         XSetAbandonException, XSetCorruptException,
                         XSystemAbandonException, XSystemCorruptException,
                         XAMException;

        public void createProperty(String fieldName, boolean binding, long value)
                  throws AuthenticationExpiredException, InvalidFieldNameException,
                         FieldExistsException, InvalidArgumentException,
                         MaximumFieldException, ObjectInUseException,
                         XSetAbandonException, XSetCorruptException,
                         XSystemAbandonException, XSystemCorruptException,
                         XAMException;

        public void createProperty(String fieldName, boolean binding, double value)
                  throws AuthenticationExpiredException, InvalidFieldNameException,
                         FieldExistsException, InvalidArgumentException,
                         MaximumFieldException, ObjectInUseException,
                         XSetAbandonException, XSetCorruptException,
                         XSystemAbandonException, XSystemCorruptException,
                         XAMException;

        public void createProperty(String fieldName, boolean binding, XUID value)
                  throws AuthenticationExpiredException, InvalidFieldNameException,
                         FieldExistsException, InvalidArgumentException,
                         MaximumFieldException, ObjectInUseException,
                         XSetAbandonException, XSetCorruptException,
                         XSystemAbandonException, XSystemCorruptException,
                         XAMException;

        public void createProperty(String fieldName, boolean binding, String value)
                  throws AuthenticationExpiredException, InvalidFieldNameException,
                         FieldExistsException, InvalidArgumentException,
                         MaximumFieldException, ObjectInUseException,
                         XSetAbandonException, XSetCorruptException,
                         XSystemAbandonException, XSystemCorruptException,
                         XAMException;

        public void createProperty(String fieldName, boolean binding, Calendar value)
                  throws AuthenticationExpiredException, InvalidFieldNameException,
                         FieldExistsException, InvalidArgumentException,
                         MaximumFieldException, ObjectInUseException,
                         XSetAbandonException, XSetCorruptException,
                         XSystemAbandonException, XSystemCorruptException,
                         XAMException;

        // == Property value mutators (setters) ==
        public void setProperty(String fieldName, boolean value)
                  throws AuthenticationExpiredException, InvalidFieldNameException,
```

```
                InvalidFieldTypeException, ObjectInUseException,
                XSetAbandonException, XSetCorruptException,
                XSystemAbandonException, XSystemCorruptException,
                XAMException;

    public void setProperty(String fieldName, Calendar value)
            throws AuthenticationExpiredException, InvalidArgumentException,
                   InvalidFieldNameException, InvalidFieldTypeException,
                   ObjectInUseException, XSetAbandonException,
                   XSetCorruptException, XSystemAbandonException,
                   XSystemCorruptException, XAMException;

    public void setProperty(String fieldName, double value)
            throws AuthenticationExpiredException, InvalidFieldNameException,
                   InvalidFieldTypeException, ObjectInUseException,
                   XSetAbandonException, XSetCorruptException,
                   XSystemAbandonException, XSystemCorruptException,
                   XAMException;

    public void setProperty(String fieldName, long value)
            throws AuthenticationExpiredException, InvalidFieldNameException,
                   InvalidFieldTypeException, ObjectInUseException,
                   XSetAbandonException, XSetCorruptException,
                   XSystemAbandonException, XSystemCorruptException,
                   XAMException;

    public void setProperty(String fieldName, String value)
            throws AuthenticationExpiredException, InvalidArgumentException,
                   InvalidFieldNameException, InvalidFieldTypeException,
                   ObjectInUseException, XSetAbandonException,
                   XSetCorruptException, XSystemAbandonException,
                   XSystemCorruptException, XAMException;

    public void setProperty(String fieldName, XUID value)
            throws AuthenticationExpiredException, InvalidArgumentException,
                   InvalidFieldNameException, InvalidFieldTypeException,
                   ObjectInUseException, XSetAbandonException,
                   XSetCorruptException, XSystemAbandonException,
                   XSystemCorruptException, XAMException;

    // == Property value accessors (getters) ==
    public boolean getBoolean(String fieldName)
            throws AuthenticationExpiredException, InvalidFieldNameException,
                   InvalidFieldTypeException, ObjectInUseException,
                   XSetAbandonException, XSetCorruptException,
                   XSystemAbandonException, XSetCorruptException,
                   XAMException;

    public Calendar getDateTime(String fieldName)
            throws AuthenticationExpiredException, InvalidFieldNameException,
                   InvalidFieldTypeException, ObjectInUseException,
                   XSetAbandonException, XSetCorruptException,
                   XSystemAbandonException, XSetCorruptException,
                   XAMException;
```

```
public double getDouble(String fieldName)
          throws AuthenticationExpiredException, InvalidFieldNameException,
                 InvalidFieldTypeException, ObjectInUseException,
                 XSetAbandonException, XSetCorruptException,
                 XSystemAbandonException, XSetCorruptException,
                 XAMException;

public long getLong(String fieldName)
          throws AuthenticationExpiredException, InvalidFieldNameException,
                 InvalidFieldTypeException, ObjectInUseException,
                 XSetAbandonException, XSetCorruptException,
                 XSystemAbandonException, XSetCorruptException,
                 XAMException;

public String getString(String fieldName)
          throws AuthenticationExpiredException, InvalidFieldNameException,
                 InvalidFieldTypeException, ObjectInUseException,
                 XSetAbandonException, XSetCorruptException,
                 XSystemAbandonException, XSetCorruptException,
                 XAMException;

public XUID getXUID(String fieldName)
          throws AuthenticationExpiredException, InvalidFieldNameException,
                 InvalidFieldTypeException, ObjectInUseException,
                 XSetAbandonException, XSetCorruptException,
                 XSystemAbandonException, XSetCorruptException,
                 XAMException;

// == XStream Creation ==
public XStream createXStream( String fieldName,
                              boolean binding,
                              String mimeType)
          throws AuthenticationExpiredException, InvalidArgumentException,
                 InvalidFieldNameException, InvalidOperationException,
                 FieldExistsException, MaximumFieldException,
                 ObjectInUseException, XSetAbandonException,
                 XSetCorruptException, XSystemAbandonException,
                 XSystemCorruptException, XAMException;

// == XStream Mutator/Accessor (setter/getter) ==
public XStream openXStream(String fieldName, String mode)
          throws AuthenticationExpiredException, InvalidFieldNameException,
                 FieldDoesNotExistException, ObjectInUseException,
                 XSetAbandonException, XSetCorruptException,
                 XSystemAbandonException, XSystemCorruptException,
                 XAMException;

// == Field Operations (XStreams or Properties) ==
public String getFieldType(String fieldName)
          throws AuthenticationExpiredException, InvalidFieldNameException,
                 FieldDoesNotExistException, ObjectInUseException,
                 XSetAbandonException, XSetCorruptException,
                 XSystemAbandonException, XSystemCorruptException,
                 XAMException;
```

```
    public long getFieldLength(String fieldName)
              throws AuthenticationExpiredException, InvalidFieldNameException,
                     FieldDoesNotExistException, ObjectInUseException,
                     XSetAbandonException, XSetCorruptException,
                     XSystemAbandonException, XSystemCorruptException,
                     XAMException;

    public boolean getFieldBinding(String fieldName)
              throws AuthenticationExpiredException, InvalidFieldNameException,
                     FieldDoesNotExistException, ObjectInUseException,
                     XSetAbandonException, XSetCorruptException,
                     XSystemAbandonException, XSystemCorruptException,
                     XAMException;

    public boolean getFieldReadOnly(String fieldName)
              throws AuthenticationExpiredException, InvalidFieldNameException,
                     FieldDoesNotExistException, ObjectInUseException,
                     XSetAbandonException, XSetCorruptException,
                     XSystemAbandonException, XSystemCorruptException,
                     XAMException;

    public void deleteField(String fieldName)
              throws AuthenticationExpiredException, InvalidFieldNameException,
                     FieldDoesNotExistException, ObjectInUseException,
                     XSetAbandonException, XSetCorruptException,
                     XSystemAbandonException, XSystemCorruptException,
                     XAMException;

    // == Binding Attribute for Fields ==
    public void setFieldAsBinding(String fieldName)
              throws AuthenticationExpiredException, InvalidFieldNameException,
                     InvalidOperationException, FieldDoesNotExistException,
                     ObjectInUseException, XSetAbandonException,
                     XSetCorruptException, XSystemAbandonException,
                     XSystemCorruptException, XAMException;

    public void setFieldAsNonbinding(String fieldName)
              throws AuthenticationExpiredException, InvalidFieldNameException,
                     FieldDoesNotExistException, ObjectInUseException,
                     XSetAbandonException, XSetCorruptException,
                     XSystemAbandonException, XSystemCorruptException,
                     XAMException;

    // == Open XStream ==
    public XAsync asyncOpenXStream( String         name,
                                    String         mode,
                                    long           xopid,
                                    XAsyncListener listener )
              throws AuthenticationExpiredException, InvalidArgumentException,
                     InvalidFieldNameException, FieldDoesNotExistException,
                     ObjectInUseException, XSetAbandonException,
                     XSetCorruptException, XSystemAbandonException,
                     XSystemCorruptException, XAMException;
}
```

A.5 XStream.java

```
package org.snia.xam;

/**
 * XStreams are the interface to the content portions of a XAM Storage System.
 * XStream implementations are provided by a storage vendor in the vendor's
 * packages. XStream instances perform the storage vendor-specific functionality
 * to interact with that storage vendor's XAM Storage System.
 */
public interface XStream
{
    public static final int SEEK_SET = 0;

    public static final int SEEK_CUR = 1;

    public static final int SEEK_END = 2;

    public final static String MODE_READ_ONLY = "readonly";

    public final static String MODE_WRITE_TRUNCATE = "writeonly";

    public final static String MODE_WRITE_APPEND = "appendonly";

    public long tell()
              throws AuthenticationExpiredException, XStreamAbandonException,
                     XStreamCorruptException, XAMException;

    public void seek(long offset, long whence)
              throws AuthenticationExpiredException, InvalidArgumentException,
                     XStreamAbandonException, XStreamCorruptException,
                     XStreamException, XAMException;

    public long write( byte buffer[] )
              throws AuthenticationExpiredException, InvalidArgumentException,
                     XStreamAbandonException, XStreamCorruptException,
                     XStreamException, XSetUnderHoldException, XAMException;

    public long write( byte buffer[], long bytesToWrite )
              throws AuthenticationExpiredException, InvalidArgumentException,
                     XStreamAbandonException, XStreamCorruptException,
                     XStreamException, XSetUnderHoldException, XAMException;

    public long write(byte buffer[], long offset, long bytesToWrite)
              throws AuthenticationExpiredException, InvalidArgumentException,
                     XStreamAbandonException, XStreamCorruptException,
                     XStreamException, XSetUnderHoldException, XAMException;

    public int read( byte buffer[] )
              throws AuthenticationExpiredException, InvalidArgumentException,
                     XStreamAbandonException, XStreamCorruptException,
                     XStreamException, XAMException;

    public int read( byte buffer[] long bytesToWrite )
              throws AuthenticationExpiredException, InvalidArgumentException,
                     XStreamAbandonException, XStreamCorruptException,
```

```
                    XAMException;

    public int read(byte buffer[], long offset, long bytesToRead)
              throws AuthenticationExpiredException, InvalidArgumentException,
                    XStreamAbandonException, XStreamCorruptException,
                    XAMException;

    public void close()
              throws AuthenticationExpiredException, XSetUnderHoldException,
                    XStreamCorruptException, XStreamException, XAMException;

    public void abandon()
              throws AuthenticationExpiredException, XAMException;

    public XAsync asyncRead( byte           buffer[],
                             long           bufferLength,
                             long           xopid,
                             XAsyncListener listener )
              throws AuthenticationExpiredException, InvalidArgumentException,
                    XStreamAbandonException, XStreamCorruptException,
                    XStreamException, XAMException;

    public XAsync asyncWrite( byte           buffer[],
                              long           bufferLength,
                              long           xopid,
                              XAsyncListener listener )
              throws AuthenticationExpiredException, InvalidArgumentException,
                    XStreamAbandonException, XStreamCorruptException,
                    XStreamException, XAMException;

    public XAsync asyncClose( long           xopid,
                              XAsyncListener listener )
              throws AuthenticationExpiredException, InvalidArgumentException,
                    XStreamCorruptException, XStreamException, XAMException;
}
```

A.6 XAsync.java

```
package org.snia.xam;

/** The XAsync object represents an asynchronous XAM operations. The XAsync object
 * contains the state information regarding the asynchronous operation.
 * Asynchronous operations are in one of two states: pending and completed.
 * When the operation is first initiated, it is in the pending state.
 * Because the operation has not completed, it is only possible to query
 * whether the operation has completed, retrieve the XOPID that was specified
 * when the operation was initiated, and to halt the operation.
 */
public interface XAsync
{
    public void halt()
              throws AuthenticationExpiredException, XAMException;

    public boolean isComplete()
              throws AuthenticationExpiredException, XAMException;
```

```
    public long getXOPID()
              throws AuthenticationExpiredException, XAMException;

    public long getStatus()
              throws AuthenticationExpiredException, XAMException;

    public XSet getXSet()
              throws AuthenticationExpiredException, XAMException;

    public XStream getXStream()
              throws AuthenticationExpiredException, XAMException;

    public XUID getXUID()
              throws AuthenticationExpiredException, XAMException;

    public long getBytesRead();

    public long getBytesWritten();

    public void close()
              throws AuthenticationExpiredException, XAMException;
}
```

A.7 XASyncListener.java

```
package org.snia.xam;

/**
 * Asynchronous XAM methods take, as an optional argument, an object which
 * implements XAsyncListener. The method XASyncCallback will be called
 * when the operation is completed (either successfully or unsuccessfully).
 */
public interface XAsyncListener
{
   public void XAsyncCallback( XAsync asyncState );
}
```

A.8 XUID.java

```
package org.snia.xam;

/**
 * This represents a XAM XUID.
 */
public interface XUID
{
    public byte[] toBytes();

    public String toString();

    public boolean equals(Object x);
}
```

A.9 XIterator.java

```
package org.snia.xam;

import java.util.Iterator;

/**
 * An XIterator is used to enumerate the field names of the fields on an XSet,
 * XSystem, or XAM object. It does not have fields itself; therefore, it is
 * not a primary XAM object.
 * The XIterator can be created with a prefix (in which case only those fields
 * that match the prefix are enumerated) or without one (in which case all fields
 * are listed). Methods also exist to retrieve the next field name (and advance
 * the cursor) and to release the resources associated with the handle. The specific
 * methods associated with an XIterator are listed with other methods.
 * An XIterator must be closed before the object FieldContainer from which it was
 * opened may be closed.
 */
public interface XIterator extends Iterator
{
    public void close() throws AuthenticationExpiredException, XAMException;
};
```

Annex B
(normative)
VIM Interface

```
package org.snia.xam.vim;

import org.snia.xam.FieldContainer;
import org.snia.xam.Logger;
import org.snia.xam.XAMException;
import org.snia.xam.XSystem;

/**
 * This interface provides a series of methods allowing the XAM Library to
 * communicate to VIMs. These methods aren't available to application programs,
 * as VIM objects are not available.
 *
 * A VIM object implements the vendor-specific logic to connect to a particular
 * XAM Storage System. The connect factory method is used by the XAM Library to
 * cause the VIM to perform vendor-specific connection logic, and provide the
 * XSystem. The provided XSystem instance is implemented in the vendor's package
 * space, but implements the org.snia.xam.XSystem interface.
 */
public interface VIM
{
    public XSystem createXSystem() throws XAMException;
}
```

Annex C
(normative)
Java-Specific Toolkit

This annex defines toolkit functions that will extend the XAM Java API to make it easier to use with Java run-time systems. While the XAM Java API does not require these methods, a XAM Library implementation may find it useful to include these utility classes. These toolkit functions shall be implemented in a way that makes no assumptions about any particular implementation of a XAM Library.

For all of the examples below, the bodies of implementations are missing, to keep the examples short.

C.1 Extended FieldContainer

Many Java packages deal with “boxed values”, which are object versions of the built-in primitives. The base XAM Java API does not deal with the boxed values from Java, namely, Long or Double. The class ExtendedFieldContainer supplies the methods to do the simple object manipulations of the primitives for create, set, and get of properties. To keep this example short, FieldContainer methods are not shown, and only the additional methods for the ExtendedFieldContainer are shown.

```
public class ExtendedFieldContainer implements FieldContainer
{
              /** Create a property by being passed an object, which is one of
               *  Long, Double, String, Calendar, XUID, Boolean. All other will
               *  throw an exception.
               */
              public void createProperty(String fieldName,
                                         boolean binding,
                                         Object value)
                 throws AuthenticationExpiredException,
                        InvalidFieldNameException,
                        FieldExistsException, InvalidArgumentException,
                        MaximumFieldException, ObjectInUseException,
                        XSetAbandonException, XSetCorruptException,
                        XSystemAbandonException, XSystemCorruptException,
                        XAMException;

              /** Set the value of property by being passed an object,
               *  which is one of Long, Double, String, Calendar, XUID,
               *  Boolean. All other will throw an exception.
               */
              public void setProperty(String fieldName,
                                      Object value)
                 throws AuthenticationExpiredException,
                        InvalidArgumentException,
                        InvalidFieldNameException, InvalidFieldTypeException,
                        ObjectInUseException, XSetAbandonException,
                        XSetCorruptException, XSystemAbandonException,
                        XSystemCorruptException, XAMException;

              /** Get the value of property returning an object,
               *  which is one of Long, Double, String, Calendar, XUID,
               *  Boolean. Applications must perform conversions if they wish
               *  wish to use Integer, Float, or Date classes.
               */
```

```
                public java.lang.Object getProperty( String fieldName )
                    throws AuthenticationExpiredException,
                           InvalidFieldNameException,
                           InvalidFieldTypeException, ObjectInUseException,
                           XSetAbandonException, XSetCorruptException,
                           XSystemAbandonException, XSetCorruptException,
                           XAMException;

                /** Convenience functions to determine if a field is a property. */
                public boolean fieldIsProperty( String fieldName )
                    throws AuthenticationExpiredException,
                           FieldDoesNotExistException,
                           XSetAbandonException, XSetCorruptException,
                           XSystemAbandonException, XSystemCorruptException,
                           XAMException;

                /** Convenience function to determine if a field is an XStream. */
                public boolean fieldIsStream( Sting fieldName )
                    throws AuthenticationExpiredException,
                           FieldDoesNotExistException,
                           XSetAbandonException, XSetCorruptException,
                           XSystemAbandonException, XSystemCorruptException,
                           XAMException;
}
```

1.0.1 createProperty

```
public void createProperty(
     String  fieldName,
     boolean binding,
     Object  value )
```

Creates a property value based on the Object version of an SType value. This method shall only accept the following values: Boolean, Calendar, Double, Long, String, or XUID.

- Parameters:
  - — fieldName - The name for the new property being created.
  - — binding - TRUE if this property is to be BOUND.
  - — value - The Object value for this property.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - value is null or an non-allowed object was passed as a value.
  - InvalidFieldNameException - fieldName is null or malformed.
  - FieldExistsException - The specified field already exists.
  - MaximumFieldException - Too many fields in the XSet
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

1.0.2 setProperty

```
public void createProperty(
     String  fieldName,
     boolean binding,
     Object  value )
```

Sets a property value based on the Object version of an SType value. This method shall only accept the following values: Boolean, Calendar, Double, Long, String, or XUID.

- Parameters:
  - fieldName - The name of the property being modified.
  - binding - TRUE if this property is to be BOUND.
  - value - The Object value for this property.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidArgumentException - value is null or an non-allowed object was passed as a value.
  - InvalidFieldTypeException - There is a type mismatch between the existing field and value argument for this method.
  - ObjectInUseException - The XSet has open import/export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

1.0.3 getProperty

```
public Object getProperty( String  fieldName )
```

Gets the value of a property as a Java Object. This method shall only return the following values: Boolean, Calendar, Double, Long, String, or XUID.

- Parameters: fieldName - The name for the new property being created.
- Returns: An object value of the appropriate type.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - InvalidFieldNameException - fieldName is null or malformed.
  - InvalidFieldTypeException - The specified field is not one of the XAM Stypes.
  - FieldDoesNotExistException - The named field does not exist in this FieldContainer.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.

- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

1.0.4 fieldIsProperty

```
public boolean fieldIsProperty(
     String  fieldName )
```

Provides a quick check to determine if a named field is a property.

- Parameters: fieldName - The name for the new property being checked.
- Returns: TRUE if field's type is one of the define XAM stypes, FALSE otherwise.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - FieldDoesNotExistException - The specified field already exists.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

1.0.5 fieldIsXStream

```
public boolean fieldIsXStream(
     String  fieldName )
```

Provides a quick check to determine if a named field is an XStream.

- Parameters: fieldName - The name for the new property being checked.
- Returns: TRUE if field's type is not one of the define XAM stypes, FALSE otherwise.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - FieldDoesNotExistException - The specified field already exists.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

C.2 XUID

Toolboxes should supply a public implementation of an object implementing the XUID interface. This tool is used to allow applications the ability to create XUIDs without interacting with the XAM Library. An example class is shown here.

```
public class XUID implements org.snia.xam.XUID
{
              public XUID( String base64String );

              public XUID( byte binary[] );

              public byte[] toBytes();

              public String toString()};

              public boolean equals(Object xuid);
}
```

The critical piece for this toolbox class is to have the two constructors public, so that the application may use these instances. All other methods specified as part of the org.snia.xam.XUID interface must be implemented as specified in Section C.4, “Java Output Stream”.

1.0.6 XUID Constructor

```
public XUID(
     String  base64String )
```

Creates a XUID from a base64-encoded value.

- Parameters: base64String - The base64-encoded value of the XUID.
- Throws: InvalidXUIDException - The base64String is not encoded correctly, or the XUID value is illegal.
- Thread Safety: This method is thread safe.

- Blocking: This method blocks until completion.

1.0.7 XUID Constructor

```
public XUID(
     byte binary[] )
```

Creates a XUID from a binary array of byte values.

- Parameters: binary - The binary bytes of the XUID.
- Throws: InvalidXUIDException - The binary array is the wrong length is not encoded correctly, or the XUID value is illegal.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

1.0.8 toBytes

```
public byte[] toBytes()
```

- Returns: An array of bytes representing the binary value of the XUID.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

1.0.9 toString

```
public toString()
```

- Returns: A base64-encoded representation of the XUID.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

1.0.10 equals

```
public equals(
     Object  xuid )
```

Compares this XUID value with another XUID for equality.

- Parameters: xuid - The other XUID to compare.
- Throws: InvalidArgumentException - The object passed in is not a XUID.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

C.3 Java Input Stream

The XAM XStream objects do not implement Java read or write streams, which limits their usefulness with the Java IO packages. Adaptor classes shall be supplied to provide Java IO stream interfaces to XAM

XStreams. These classes shall provide the required methods to satisfy the Java InputStream and OutputStream interfaces.

```
package org.snia.xam.util;

import java.io.IOException;
import java.io.InputStream;

import org.snia.xam.EndOfStreamException;
import org.snia.xam.FieldContainer;
import org.snia.xam.XAMException;
import org.snia.xam.XStream;

/**
 * An InputStream wrapper class for XStream objects.
 */
public class XStreamInputStream extends InputStream
              /**
               * Opens an existing XStream in read only mode and
               * wraps it in a InputStream interface.
               *
               * @param fc The FieldContainer that contains the XStream
               * @param name The name of the XStream
               * @throws XAMException If the XStream does not exist,
               *         or an error occurs trying to open it.
               */
              public XStreamInputStream( FieldContainer fc,
                                         String         name)
                  throws AuthenticationExpiredException,
                         InvalidFieldNameException, FieldDoesNotExistException,
                         ObjectInUseException,
                         XSetAbandonException, XSetCorruptException,
                         XSystemAbandonException, XSystemCorruptException,
                         XAMException;

              /**
               * Wraps an open XStream in an InputStream interface.
               * The XStream must have been opened in read only mode.
               *
               * @param stream An open read only XStream
               */
              public XStreamInputStream(XStream stream)
                  throws AuthenticationExpiredException,
                         XSetAbandonException, XSetCorruptException,
                         XSystemAbandonException, XSystemCorruptException,
                         XAMException;

              /**
               * Closes the underlying XStream
               */
              public void close() throws IOException;

              public boolean markSupported();

              public int available() throws IOException;
```

```
        public int read() throws IOException;

        public int read(byte[] b) throws IOException;

        public int read(byte[] b, int offset, int length)
            throws IOException;
}
```

1.0.11 XStreamInputStream

```
public XStreamInputStream(
     FieldContainer fc,
     String         name  )
```

Creates an XStreamInputStream from a named field in the specified FieldContainer.

- Parameters:
  - fc - A FieldContainer object containing the XStream.
  - name - The name of the XStream field.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - FieldDoesNotExistException - The named XStream does not exist in the FieldContainer.
  - ObjectInUseException - The XSet has open import or export streams.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Some other exception occurred opening the XStream.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

1.0.12 XStreamInputStream

```
public XStreamInputStream(
     XStream stream )
```

Creates an XStreamInputStream from a named field in the specified FieldContainer.

- Parameters: stream - XStream providing data to the XStreamInputStream.

- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Some other exception occurred opening the XStream.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

1.0.13 close

```
public void close()
```

Closes this input stream and the underlying XStream, and releases any resource associated with this input stream.

- Throws: IOException - This base java exception is thrown when any error occurs and shall wrap lower level exceptions generated by the XStream or XSystem.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

1.0.14 markSupported

```
public boolean markSupported()
```

Tests if this input stream supports the mark and reset methods. This functionality is optional.

- Returns: TRUE if this stream implements mark and reset functionality, FALSE otherwise.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

1.0.15 available

```
public int available()
```

Returns an estimate of the number of bytes that can be read (or skipped over) from this input stream without blocking by the next invocation of a method for this input stream. The next invocation might be the same thread or another thread. A single read or skip of this many bytes will not block, but may read or skip fewer bytes.

Note that while some implementations of InputStream will return the total number of bytes in the stream, many will not. It is never correct to use the return value of this method to allocate a buffer intended to hold all data in this stream.

- Returns: The approximate number of bytes which can be read without blocking.
- Throws: IOException - This base java exception is thrown when any error occurs and shall wrap lower level exceptions generated by the XStream or XSystem.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

1.0.16 read

```
public int read()
```

Reads the next byte from the input stream. The value byte is returned as an int in the range 0 to 255. If no byte is available because the end of the stream has been reached, the value -1 is returned. This method blocks until input data is available, the end of the stream is detected, or an exception is thrown. If an application wishes to avoid blocking behavior, it should be written using the XAsync mechanisms.

- Returns: Byte value read from the input stream, or -1 if end the stream is reached.
- Throws: IOException - This base java exception is thrown when any error occurs and shall wrap lower level exceptions generated by the XStream or XSystem.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

1.0.17 read

```
public int read( byte[] b )
```

Reads some number of bytes from the input stream and stores them into the buffer array b. The number of bytes actually read is returned as an integer. This method blocks until input data is available, end of file is detected, or an exception is thrown.

If the length of b is zero, then no bytes are read and 0 is returned; otherwise, there is an attempt to read at least one byte. If no byte is available because the stream is at the end of the file, the value -1 is returned; otherwise, at least one byte is read and stored into b.

The first byte read is stored into element b[0], the next one into b[1], and so on. The number of bytes read is, at most, equal to the length of b. Let k be the number of bytes actually read; these bytes will be stored in elements b[0] through b[k-1], leaving elements b[k] through b[b.length-1] unaffected.

The read(b) method for class InputStream has the same effect as:

```
read(b, 0, b.length)
```

- Parameters: b - The buffer into which the data is read.
- Returns: The total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream was reached.
- Throws: IOException - This base java exception is thrown when any error occurs and shall wrap lower level exceptions generated by the XStream or XSystem.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

1.0.18 read

```
public int read(
            byte[] b
            int off
            int len )
```

Reads up to len bytes of data from the input stream into an array of bytes. An attempt is made to read as many as len bytes, but a smaller number may be read. The number of bytes actually read is returned as an integer.

This method blocks until input data is available, end of file is detected, or an exception is thrown.

If len is zero, then no bytes are read and 0 is returned; otherwise, there is an attempt to read at least one byte. If no byte is available because the stream is at end of file, the value -1 is returned; otherwise, at least one byte is read and stored into b.

The first byte read is stored into element b[off], the next one into b[off+1], and so on. The number of bytes read is, at most, equal to len. Let k be the number of bytes actually read; these bytes will be stored in elements b[off] through b[off+k-1], leaving elements b[off+k] through b[off+len-1] unaffected.

In every case, elements b[0] through b[off] and elements b[off+len] through b[b.length-1] are unaffected.

The read(b, off, len) method for class InputStream simply calls the method read repeatedly. If the first such call results in an IOException, that exception is returned from the call to the read(b, off, len) method. If any subsequent call to read results in a IOException, the exception is caught and treated as if it were end of file; the bytes read up to that point are stored into b and the number of bytes read before the exception occurred is returned. The default implementation of this method blocks until the requested amount of input data len has been read, end of file is detected, or an exception is thrown. Subclasses are encouraged to provide a more efficient implementation of this method.

- Parameters:
  - b - The buffer into which the data is read.
  - off - The start offset in the array b at which the data is written.
  - len - The maximum number of bytes to read.
- Returns: The total number of bytes read into the buffer or -1 if end the stream is reached.
- Throws: IOException - This base java exception is thrown when any error occurs and shall wrap lower level exceptions generated by the XStream or XSystem.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

C.4 Java Output Stream

```
package org.snia.xam.util;

import java.io.IOException;
import java.io.OutputStream;

import org.snia.xam.FieldContainer;
import org.snia.xam.XAMException;
```

```
import org.snia.xam.XStream;

/**
 * An OutputStream wrapper class for XStream objects.
 */
public class XStreamOutputStream extends OutputStream
{
              /**
               * Opens an existing XStream and either truncates it or appends to
               * it based on the value of the append parameter. The opened XStream
               * is wrapped in an InputStream interface.
               *
               * @param ct The FieldContainer containing the XStream
               * @param name The name of the XStream
               * @param append True to append new data, false to overwrite
               *              existing data
               * @throws XAMException If an error occurs trying to open
               *         the XStream
               */
              public XStreamOutputStream( FieldContainer ct,
                                          String         name,
                                          boolean        append)
                 throws AuthenticationExpiredException,
                        InvalidFieldNameException, FieldReadOnlyException,
                        ObjectInUseException, XSetUnderHoldException,
                        XSetAbandonException, XSetCorruptException,
                        XSystemAbandonException, XSystemCorruptException,
                        XAMException;

              /**
               * Creates a new XStream and wraps it in an InputStream interface.
               *
               * @param ct The FieldContainer to create the XStream in
               * @param name The name of the XStream
               * @param binding True if the XStream is binding
               * @param mimeType The mime type of the XStream
               * @throws XAMException If an error occurs while trying to
               * create the XStream
               */
              public XStreamOutputStream( FieldContainer ct,
                                          String         name,
                                          boolean        binding,
                                          String         mimeType)
                 throws AuthenticationExpiredException,
                        XSetAbandonException, XSetCorruptException,
                        XSystemAbandonException, XSystemCorruptException,
                        XAMException;

              /**
               * Wraps an open XStream in an InputStream interface. The XStream
               * must have been opened in either write append or write truncate
               * mode.
               *
               * @param stream An open XStream
               */
```

```
                public XStreamOutputStream( XStream stream);

                /**
                 * Flushes all write operations and closes the underlying XStream.
                 */
                public void close() throws IOException;

                public void flush() throws IOException;

                public void write( int b ) throws IOException;

                public void write( byte[] b ) throws IOException;

                public void write( byte[] b, int offset, int length)
                      throws IOException;
}
```

1.0.19 XStreamOutputStream

```
 public XStreamInputStream(
      FieldContainer fc,
      String         name  )
```

Creates an XStreamOutputStream to a named field in the specified FieldContainer.

- Parameters:
  - fc - A FieldContainer object containing the XStream.
  - name - The name of the XStream field.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - FieldReadOnlyException - The named XStream exists in the FieldContainer and is Read Only.
  - InvalidFieldNameException - The field name is null or malformed.
  - ObjectInUseException - The XSet has open import or export streams.
  - XSetUnderHoldException - The XSet is under hold and may not be modified.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Some other exception occurred opening the XStream.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

1.0.20 XStreamInputStream

```
public XStreamOutputStream(
     XStream stream )
```

Creates an XStreamOutputStream to a named field in the specified FieldContainer.

- Parameters: stream - XStream providing data to the XStreamInputStream.
- Throws:
  - AuthenticationExpiredException - The latest authentication has expired and the application must reauthenticate.
  - XSetAbandonException - The XSet is in the abandoned state and may only be closed.
  - XSetCorruptException - The XSet is in the corrupt state and may only be abandoned.
  - XSystemAbandonException - The XSystem is in the abandoned state and may only be closed.
  - XSystemCorruptException - The XSystem is in the corrupt state and may only be abandoned.
  - XAMException - Some other exception occurred opening the XStream.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

1.0.21 close

```
public void close()
```

Closes this input stream and the underlying XStream, and releases any resource associated with this input stream.

- Throws: IOException - This base java exception is thrown when any error occurs and shall wrap lower level exceptions generated by the XStream or XSystem.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

1.0.22 flush

```
public void flush()
```

Flushes this output stream and forces any buffered output bytes to be written out.

- Throws: IOException - This base java exception is thrown when any error occurs and shall wrap lower level exceptions generated by the XStream or XSystem.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

1.0.23 write

```
public void read( int b )
```

Writes the specified byte to this output stream. The general contract for write is that one byte is written to the output stream. The byte to be written is the eight low-order bits of the argument b. The 24 high-order bits of b are ignored.

- Parameters: b - The byte value to write.
- Throws: IOException - This base java exception is thrown when any error occurs and shall wrap lower level exceptions generated by the XStream or XSystem.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

1.0.24 write

```
public void write( byte[] b )
```

Writes b length bytes from the specified byte array to this output stream. The general contract for write(b) is that it should have exactly the same effect as the call write(b, 0, b.length).

- Throws: IOException - This base java exception is thrown when any error occurs and shall wrap lower level exceptions generated by the XStream or XSystem.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

1.0.25 write

```
public void write(
             byte[] b
             int off
             int len )
```

Writes len bytes from the specified byte array starting at offset off to this output stream. The general contract for write(b, off, len) is that some of the bytes in the array b are written to the output stream in order; element b[off] is the first byte written and b[off+len-1] is the last byte written by this operation.

The write method of OutputStream calls the write method of one argument on each of the bytes to be written out. Subclasses are encouraged to override this method and provide a more efficient implementation.

If off is negative, or len is negative, or off+len is greater than the length of the array b, then an IndexOutOfBoundsException is thrown.

- Parameters:
  - b - The buffer into which the data is read.
  - off - The start offset in the array b at which the data is written.
  - len - The maximum number of bytes to read.

- Throws: IOException - This base java exception is thrown when any error occurs and shall wrap lower level exceptions generated by the XStream or XSystem.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

Annex D
(informative)
Java API Method Mapping

Table D.1, “Java Method Name Mapping to XAM Architecture Specification” lists the methods in [XAM-ARCH] and the corresponding method name for the Java binding.

Table D.1 – Java Method Name Mapping to XAM Architecture Specification

Type	Methods in Arch Spec	Methods in Java API Spec
XAM Library	XAMLibrary.connect	connect
XSystem	N/A[1]	XSystem connect
	XSystem.authenticate	authenticate
	XSystem.close	close
	XSystem.abandon	abandon
	XSystem.deleteXSet	deleteXSet
	XSystem.isXSetRetained	isXSetRetained
	XSystem.holdXSet	holdXSet
	XSystem.releaseXSet	releaseXSet
	XSystem.accessXSet	accessXSet
	XSystem.getXSetAccessTime	getXSetAccessTime
	XSystem.createXSet	createXSet
	XSystem.openXSet	openXSet
	XSystem.copyXSet	copyxset
	XSystem.asyncOpenXSet	asyncOpenXSet
	XSystem.asyncCopyXSet	asyncCopyXSet
XSet	XSet.applyAccessPolicy	applyAccessPolicy
	XSet.resetAccessFields	resetAccessFields
	XSet.applyManagementPolicy	applyManagementPolicy
	XSet.resetManagementFields	resetManagementFields
	XSet.createRetention	createRetention
	XSet.setRetentionEnabledFlag	setRetentionEnabledFlag
	XSet.applyRetentionEnabledPolicy	applyRetentionEnabledPolicy
	XSet.setRetentionDuration	setRetentionDuration
	XSet.applyRetentionDurationPolicy	applyRetentionDurationPolicy
	XSet.setRetentionStarttime	setRetentionStarttime
	XSet.setBaseRetention	setBaseRetention

Table D.1 – Java Method Name Mapping to XAM Architecture Specification

Type	Methods in Arch Spec	Methods in Java API Spec
XSet (cont.)	XSet.applyBaseRetentionPolicy	applyBaseRetentionPolicy
	XSet.setAutoDelete	setAutoDelete
	XSet.applyAutoDeletePolicy	applyAutoDeletePolicy
	XSet.setShred	setShred
	XSet.applyShredPolicy	applyShredPolicy
	XSet.getActualRetentionDuration	getActualRetentionDuration
	XSet.getActualRetentionEnabled	getActualRetentionEnabled
	XSet.getActualAutoDelete	getActualAutoDelete
	XSet.getActualShred	getActualShred
	XSet.commit	commit
	XSet.close	close
	XSet.abandon	abandon
	XSet.submitJob	submitJob
	XSet.haltJob	haltJob
	XSet.openExportXStream	openExportXStream
	XSet.openImportXStream	openImportXStream
	XSet.asyncCommit	asyncCommit
	XSet.applyStoragePolicy	applyStoragePolicy
Field Container	<XAMHandle>.openFieldIterator	openFieldIterator
	<XAMHandle>.containsField	containsField
	<XAMHandle>.createBoolean	createProperty - xam_boolean
	<XAMHandle>.createInt	createProperty - xam_int
	<XAMHandle>.createDouble	createProperty - xam_double
	<XAMHandle>.createXUID	createProperty - xam_xuid
	<XAMHandle>.createString	createProperty - xam_string
	<XAMHandle>.createDatetime	createProperty - xam_datetime
	<XAMHandle>.setBoolean	setProperty - xam_boolean
	<XAMHandle>.setDatetime	setProperty - xam_datetime
	<XAMHandle>.setDouble	setProperty - xam_double
	<XAMHandle>.setInt	setProperty - xam_int
	<XAMHandle>.setString	setProperty - xam_string
	<XAMHandle>.setXUID	setProperty - xam_xuid
	<XAMHandle>.getBoolean	getBoolean - xam_boolean

Table D.1 – Java Method Name Mapping to XAM Architecture Specification

Type	Methods in Arch Spec	Methods in Java API Spec
Field Container (cont.)	<XAMHandle>.getDatetime	getDatetime - xam_datetime
	<XAMHandle>.getDouble	getDouble - xam_double
	<XAMHandle>.getInt	getLong - xam_int
	<XAMHandle>.getString	getString - xam_string
	<XAMHandle>.getXUID	getXUID - xam_xuid
	<XAMHandle>.createXStream	createXStream
	<XAMHandle>.openXStream	openXStream
	<XAMHandle>.getFieldType	getFieldType
	<XAMHandle>.getFieldLength	getFieldLength
	<XAMHandle>.getFieldBinding	getFieldBinding
	<XAMHandle>.getFieldReadOnly	getFieldReadOnly
	<XAMHandle>.setFieldAsBinding	setFieldAsBinding
	<XAMHandle>.setFieldAsNonBinding	setFieldAsNonBinding
	<XAMHandle>.deleteField	deleteField
	<XAMHandle>.asyncOpenXStream	asyncOpenXStream
XStream	XStream.tell	tell
	XStream.seek	seek
	XStream.write	write (3 methods)
	XStream.read	read (3 methods)
	XStream.close	close
	XStream.abandon	abandon
	XStream.asyncRead	asyncRead
	XStream.asyncWrite	asyncWrite
	XStream.asyncClose	asyncClose
XAsync	XAsync.halt	halt
	XAsync.isComplete	isComplete
	XAsync.getXOPID	getXOPID
	XAsync.getStatus	getStatus
	XAsync.getXSet	getXSet
	XAsync.getXStream	getXStream
	XAsync.getXUID	getXUID
	XAsync.getBytesRead	getBytesRead
	XAsync.getBytesWritten	getBytesWritten

Table D.1 – Java Method Name Mapping to XAM Architecture Specification

Type	Methods in Arch Spec	Methods in Java API Spec
XAsync (cont.)	XAsync.close	close
	N/A[2]	XAsyncCallback
XUID	N/A[3]	toBytes
	XUIDToString	toString
	N/A[4]	equals
XIterator	XIterator.next	next
	Xiterator.hasNext	hasNext
	XIterator.close	close
	N/A[5]	remove

1. Java-specific.
2. Java-specific.
3. Java-specific.
4. Java-specific.
5. XAM XIterator implementations shall provide the remove method to satisfy the interface but throws the exception java.lang.UnsupportedOperationException.