



---

# **eXtensible Access Method (XAM™) – SDK Developer's Guide**

**Version 1.01**

"Publication of this Working Draft for review and comment has been approved by the SDK TWG. This draft represents a "best effort" attempt by the SDK 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." Suggestion for revision should be directed to the SNIA Technical Council Managing Director at [tcmd@snia.org](mailto:tcmd@snia.org)."

***WORKING DRAFT***

**August 31, 2009**

## Revision History

Version	Date	Originator	Sections	Comments
0.8	4-9-09	M. McMinn	All	Added new section placeholders; reorganized doc and moved existing information.
	4-21-09	M. McMinn	Ch 3 & 4	Added text to “Installing Compilers and Dependent Libraries.”
	4-28-09	M. McMinn	All	Added info from updated readme’s (Java XAM Library and Java Reference VIM) and converted each chapter to MS Word for SME edits. Posted to Subversion.
0.9	7-16-09	M. McMinn	All	Incorporated SME edits and information from updated readme files. Added Error Codes and XAM Configuration chapters; added Appendix A - Reference VIM Architecture.
1.01	8-31-09	M. McMinn	All	<ul style="list-style-type: none"> <li>- Recreated all graphics in Visio</li> <li>- Updated Ch 4,5,6, &amp; 7</li> <li>- Added Appendix B - HTTP VIM Architecture &amp; Protocol</li> <li>- Added trademark to XAM</li> </ul>

The SNIA hereby grants permission for individuals to use this document for personal use only, and for corporations and other business entities to use this document for internal use only (including internal copying, distribution, and display) provided that:

- Any text, diagram, chart, table or definition reproduced shall be reproduced in its entirety with no alteration, and,
- Any document, printed or electronic, in which material from this document (or any portion hereof) is reproduced shall acknowledge the SNIA copyright on that material, and shall credit the SNIA for granting permission for its reuse.

Other than as explicitly provided above, you may not make any commercial use of this document, sell any excerpt or this entire document, or distribute this document to third parties. All rights not explicitly granted are expressly reserved to SNIA.

Permission to use this document for purposes other than those enumerated above may be requested by e-mailing [tcmd@snia.org](mailto:tcmd@snia.org) please include the identity of the requesting individual and/or company and a brief description of the purpose, nature, and scope of the requested use.

Copyright © 2009 Storage Networking Industry Association.  
XAM is a trademark of the Storage Networking Industry Association.

# Contents

<b>1 About this Guide</b>	<b>1</b>
Purpose and Audience	1
Contents	1
References	2
Additional Information	3
Conventions	3
SNIA Welcomes Your Comments	3
<b>2 Introduction to the SDK Developer's Guide</b>	<b>4</b>
SDK Terms of Use	4
SDK Components	4
Supported Operating Systems	5
Software Requirements	5
SDK Installation Instructions	5
<b>3 XAM™ Library Configuration</b>	<b>6</b>
Introduction	6
Configuration File Format	7
Configuration File Discovery	7
XAM™ Library Configuration Behavior	8
Stackable VIM Support	8
XAM™ Library Configuration Namespaces	8
Configuration File Example	9
Aliasing VIM Implementations	10
Leveraging Lookup Order for VIM Discovery	11
Preloading VIM Libraries	11
Defining VIM Stacks	11
Using Application-Defined Configurations	12
Configuring VIM-Specific Controls	12
Specifying XAM™ Library Logging Controls	12
<b>4 Java Library</b>	<b>13</b>
Understanding What Is Provided	13
Unpacking Your ZIP file	14
Installing Compilers and Dependent Libraries	15
Installing JUnit	15
Windows	15
Unix (bash)	15
Installing libcurl	16
Building the Code for Your Platform	16
Supported Platforms	16
Installing the Binaries for Your Platform	17
Installing a VIM	17
Running Tests	17
Using the Java Library	17
Obtaining an Instance of the XAM™ Library	17
Configuring the Library Initialization	18

Logging Parameters .....	19
Java Logging Use .....	19
Using Classes from <b>XAMLib.jar</b> and <b>XAMToolkit.jar</b> .....	<b>20</b>
<b>5 C Library .....</b>	<b>21</b>
Understanding What Is Provided .....	21
Unpacking Your ZIP file .....	22
Installing Compilers and Dependent Libraries .....	23
Installing JUnit .....	23
Windows .....	23
Unix (bash) .....	23
Installing libcurl .....	24
Choosing a C compiler .....	24
Building the Code for Your Platform .....	24
Building the XAM™ SDK with Visual Studio on Windows .....	25
Building the XAM™ SDK Using ant .....	25
Installing the Binaries for Your Platform .....	25
Configuring the XAM™ Library .....	25
Configuration Options .....	26
Supported Configuration Namespaces .....	26
Configuration File Syntax .....	27
Example Configuration File .....	28
Installing a VIM .....	28
Configuring the HTTP VIM .....	29
HTTP VIM Parameters .....	29
HTTP VIM Configuration File .....	29
Supported Configuration Namespaces .....	29
Configuration File Syntax .....	29
Example Configuration File: .....	30
<b>6 Java Reference VIM .....</b>	<b>31</b>
Overview .....	31
Summary of Features .....	32
XAM™ API Features .....	32
API Methods Supported .....	33
Unpacking your ZIP file .....	34
Installing Compilers and Dependent Libraries .....	34
Installing JUnit .....	34
Windows .....	34
Unix (bash) .....	35
Building the Code for Your Platform .....	35
Supported Platforms .....	35
Software Requirements .....	35
Ant .....	35
Derby .....	35
JUnit .....	35
Java .....	36
JavaCC .....	36
JavaMail .....	36
JavaBeans Activation Framework (JAF) .....	36
Lucene .....	36
PATH Requirements .....	36

Building the XAM SDK using ant.....	36
Running the ant test .....	37
Installing the Binaries for Your Platform .....	37
Configuring and Operating the XAM™ Reference VIM and Library .....	37
Basic Configuration .....	38
User Configuration .....	38
Access Control Policies .....	38
Autodelete Configuration.....	39
Autodelete and Shred Policies .....	40
Retention Policies .....	40
Import Processing .....	41
Asynchronous Methods.....	41
Reference VIM Logging .....	42
Summary of Configuration Properties .....	43
Scalability .....	43
Building and Running the Client Example .....	43
Building and Running the Example Using ant.....	43
Building and Running the Example From the Command Line .....	44
Viewing Build Example Output - ant.....	44
Using Reference VIM Example Programs .....	45
Building and Running Tests and Examples .....	45
Configuring Unit Tests.....	45
Building and Running Tests Using ant .....	46
Running Your Application with XAM™ .....	46
Default Repository Location .....	46
Directories and Files Created.....	47
Files Created for a Persisted XSet .....	47
Temp Files Created .....	47
Database For Query Support.....	47
Specification of a Different Reference VIM Repository Location.....	48
Repository Maintenance .....	48
<b>7 HTTP Protocol VIM .....</b>	<b>50</b>
Description .....	50
Functionality.....	51
Server Configuration, Installation, Building, and Testing .....	51
Configuring the HTTP VIM Implementation Target of the HTTP VIM	
Client.....	52
Installing Required Runtime Libraries .....	52
Building the Server .....	53
Running Ant Tasks .....	53
Verifying the Server.....	53
Starting the Server .....	54
Protocol VIM Use .....	54
Java VIM Requirements .....	55
<b>8 Error Codes .....</b>	<b>56</b>
XAMException .....	56
FieldContainerException.....	57
JobException .....	57
XSetException .....	58
XStreamException .....	58

XSystemException.....	59
Non-Categorized C Errors .....	59
<b>Appendix A:Reference VIM Architecture .....</b>	<b>60</b>
Class Structure .....	60
XSystem.....	61
XSet .....	62
XStream .....	63
Persistence Manager .....	64
Default Repository Location.....	64
Directories and Files Created .....	65
Files Created for a Persisted XSet .....	65
Temp Files Created .....	65
Specification of a VIM Repository Location .....	65
Policy.....	66
Access Policies.....	67
Disposition Policies.....	67
Retention Policies .....	67
Jobs.....	68
DBManager .....	68
Operational Flow.....	71
Initializing an XSystem .....	71
Importing an XSet .....	73
Processing a Query.....	75
Future Ideas.....	81
<b>Appendix B:HTTP VIM Architecture .....</b>	<b>83</b>
Terms and Scope .....	83
Terms .....	83
Scope .....	84
Overview of HTTP VIM Design.....	85
Java HTTP VIM Client .....	88
Connect Processing .....	89
XAsyncCallback Management .....	89
Java HTTP VIM Server.....	90
VIM Class and Library Load Operations.....	92
VIM Wire Protocol.....	92
Organization.....	93
Method Access.....	93
Return Values .....	93
Value Encoding .....	93
Example Exchange .....	93
Operations.....	94
XAMCreateFieldIterator .....	94
XAsyncClose .....	94
XAsyncGetBytesRead .....	95
XAsyncGetBytesWritten .....	95
XAsyncGetStatus.....	96
XAsyncGetXOPID.....	97
XAsyncGetXSet .....	97
XAsyncGetXStream .....	98
XAsyncGetXUID .....	98
XAsyncHalt .....	99

XAsyncIsComplete .....	99
XAsync – POLL .....	100
XIteratorClose .....	101
XIteratorHasNext .....	101
XIteratorNext.....	102
XSetAbandon.....	102
XSetApplyAccessPolicy.....	103
XSetApplyAutoDeletePolicy.....	103
XSetApplyBaseRetention .....	104
XSetApplyManagementPolicy .....	104
XSetApplyRetentionDurationPolicy .....	105
XSetApplyRetentionEnabledPolicy.....	105
XSetApplyShredPolicy .....	106
XSetApplyStoragePolicy .....	106
XSetAsyncCommit.....	107
XSetAsyncOpenXStream .....	107
XSetClose.....	108
XSetCommit.....	108
XSetContainsField .....	109
XSetCreateProperty .....	109
XSetCreateRetention .....	110
XSetCreateXStream .....	111
XSetDeleteField .....	111
XSetGetActualAutoDelete .....	112
XSetGetActualRetentionDuration .....	112
XSetGetActualRetentionEnabled.....	113
XSetGetActualShred.....	113
XSetGetFieldBinding .....	114
XSetGetLength .....	114
XSetGetFieldReadOnly .....	115
XSetGetProperty.....	115
XSetGetPropertyType.....	116
XSetHaltJob .....	117
XSetOpenExportStream .....	117
XSetOpenImportStream .....	118
XSetOpenXStream .....	118
XSetResetAccessFields .....	119
XSetResetManagementFields .....	119
XSetSetAutoDelete.....	120
XSetSetBaseRetention .....	120
XSetSetFieldAsBinding.....	121
XSetSetFieldAsNonbinding .....	121
XSetSetProperty .....	122
XSetSetRetentionDuration.....	123
XSetSetRetentionEnabledFlag .....	123
XSetSetRetentionStarttime .....	124
XSetSetShred .....	124
XSetSubmitJob .....	124
XStreamAbandon .....	125
XStreamAsyncClose.....	125
XStreamAsyncRead .....	126
XStreamAsyncWrite.....	126
XStreamClose.....	127
XStreamRead .....	127
XStreamSeek.....	128

XStreamTell .....	128
XStreamWrite .....	129
XSystemAbandon .....	129
XSystemAccessXSet .....	130
XSystemAsyncCopyXSet .....	131
XSystemAsyncOpenXSet .....	131
SystemAsyncOpenXStream .....	132
XSystemAuthenticate .....	133
XSystemClose .....	133
XSystemConnect .....	134
XSystemContainsField .....	136
XSystemCopyXSet .....	136
XSystemCreateProperty .....	137
XSystemCreateXSet .....	137
XSystemCreateXStream .....	138
XSystemDeleteXSet .....	139
XSystemGetFieldBinding .....	139
XSystemGetFieldLength .....	140
XSystemGetProperty .....	140
XSystemGetPropertyType .....	141
XSystemGetXSetAccessTime .....	141
XSystemHoldXSet .....	142
XSystemIsXSetRetained .....	142
XSystemOpenXSet .....	143
XSystemOpenXStream .....	143
XSystemReleaseXSet .....	144
XSystemSetFieldAsBinding .....	145
XSystemSetFieldAsNonbinding .....	145
XSystemSetProperty .....	146
Known Issues .....	147



## Tables

Table 1 – Typographic Conventions .....	3
Table 2 – Logging Parameters .....	19
Table 3 – Access Policy Example .....	39
Table 4 – Autodelete and Shred Policies .....	40
Table 5 – Retention Policies .....	40
Table 6 – XAM™ and Java Logging Levels .....	42
Table 7 – Summary of Configuration Properties .....	43
Table 8 – XAMException Mapping .....	56
Table 9 – FieldContainerException Mapping .....	57
Table 10 – JobException Mapping .....	57
Table 11 – XSetException Mapping .....	58
Table 12 – XStreamException Mapping .....	58
Table 13 – XSystemException Mapping .....	59
Table 14 – Mapping for Non-Categorized C Errors .....	59
Table 15 – Manager Classes .....	61
Table 16 – Disposition Policies .....	67
Table 17 – Retention Policies .....	67
Table 18 – Table Columns and Value Property Names .....	79
Table 19 – Terms - HTTP VIM Design .....	83
Table 20 – CRC for XAM_API .....	86
Table 21 – CRC for VIM_API .....	86
Table 22 – CRC for XAM_Library .....	86
Table 23 – CRC for HTTP_VIM_Client .....	87
Table 24 – CRC for HTTP_VIM_Server .....	88

## Figures

Figure 1 – Manager Classes .....	55
Figure 2 – Reference XSet .....	56
Figure 3 – Reference XStream .....	57
Figure 4 – Policy System .....	60
Figure 5 – Job Manager .....	62
Figure 6 – DBManager .....	63
Figure 7 – XFieldValues Database .....	64
Figure 8 – XSystem Initialization .....	65
Figure 9 – Importing XSets .....	66
Figure 10 – Processing a Query .....	68
Figure 11 – Query Results .....	69
Figure 12 – Optimized Query .....	70

---

## Chapter 1: About this Guide

### *Purpose and Audience*

The *eXtensible Access Method (XAM™) – SDK Developer's Guide* is written for programmers and application developers who develop custom applications for XAM Storage Systems. This document, along with the following three documents, provides the information that you need to develop custom applications for a XAM Storage System. The three documents include the following:

- 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

### *Contents*

The contents of this document are described as follows:

- Chapter 1, “About this Guide” describes the audience and purpose, contents of this guide, additional references and web sites, and typographical conventions.
- Chapter 2, “Introduction to the SDK Developer's Guide” provides an introduction to the eXtensible Access Method (XAM™) – Software Development Kit (SDK) and its components.
- Chapter 3, “XAM™ Library Configuration” explains the conventions and syntax for setting generic properties for the XAM Libraries and the Reference VIM.
- Chapter 4, “Java Library” describes the Java library components for the XAM SDK.
- Chapter 5, “C Library” describes the C library components for the XAM SDK.

- Chapter 6, “Java Reference VIM” provides the deliverables, directory structure, and build and test instructions for the Java Reference VIM.
- Chapter 7, “HTTP Protocol VIM” provides the HTTP vendor interface modules for the Java and C language bindings.
- Chapter 8, “Error Codes” provides common error codes for the XAM SDK.
- Appendix A: “Reference VIM Architecture” documents the architecture of the SNIA XAM Reference VIM, which is written in and leverages the object-oriented capabilities of the Java language.
- Appendix B: “HTTP VIM Architecture” documents the architecture of the HTTP VIM and the wire protocol used between the “halves” of the HTTP VIM.

## 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.

- [XAM-ARCH] “XAM™ Architecture Specification” – Available from the SNIA Fixed Content Aware Storage TWG, Version 1.0.1
- [XAM-SDK-REQS] “XAM™ SDK Requirements” - Available from the XAM SDK TWG site, SNIA Working Draft
- [XAM-C-API] “XAM™ C API Specification” - Available from SNIA Fixed Content Aware Storage TWG, Version 1.0.1
- [XAM-Java-API] “XAM™ Java API Specification” - Available from SNIA Fixed Content Aware Storage TWG, Version 1.0.1
- [REST] “Representational State Transfer” - [http://www.ics.uci.edu/~fielding/pubs/dissertation/rest\\_arch\\_style.htm](http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm)
- [HTTP-RESPONSE] Hypertext Transfer Protocol – HTTP/1.1, Chapter 10 Status Code Definitions - <http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>
- [URL-Encoding] URL Safe character encoding <http://www.w3.org/TR/html40/appendix/notes.html#non-ascii-chars>. This mechanism is available for Java in the classes URLEncoder and URLDecoder.
- Related third-party web site references. This document references third-party URLs that provide additional, related information.

**Note:** SNIA is not responsible for the availability of third-party web sites mentioned in this document. SNIA does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. SNIA will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

## *Additional Information*

The SNIA web site provides additional information about the SNIA XAM™ initiative at <http://www.snia.org/xam>.

## *Conventions*

Table 1 describes the typographic conventions that are used in this document.

**Table 1 – Typographic Conventions**

Convention	Description
Fixed-width text	The names of commands, files and directories, and on-screen computer output
<b>Bold, fixed-width text</b>	What you type, contrasted with on-screen computer output
<i>Italicized text</i>	Variables, field names, and book titles
<b>Note:</b>	Additional or useful informative text.
<b>WARNING:</b>	Indicates that you should pay careful attention to the probable action, so that you may avoid system failure or harm.

## *SNIA Welcomes Your Comments*

SNIA is interested in improving its documentation and welcomes your comments and suggestions. You can submit your comments by sending an e-mail to [tcmd@snia.org](mailto:tcmd@snia.org).

---

## Chapter 2: Introduction to the SDK Developer's Guide

This chapter provides an introduction to the SNIA XAM™ Software Development Kit (SDK) and its components.

The following topics are discussed:

- SDK Terms of Use
- SDK Components
- Supported Operating Systems
- Software Requirements
- SDK Installation Instructions

### *SDK Terms of Use*

The SDK Developer's Guide is released to you under the following copyright notice:

"Copyright © 2009 Storage Networking Industry Association. Use is subject to license terms."

### *SDK Components*

The XAM™ SDK distribution includes the following:

- HTTP VIM
- Java Reference VIM
- Java Library
- C++ Library

### *Supported Operating Systems*

This release supports the following operating systems:

- AIX 5.2 and above
- HP-UX B.11.11
- HP-UX B.11.23
- Linux - GCC 3.3
- Linux - GCC 4.0
- Solaris 8.0 and above
- Win32 Dev 8
- Win64 Dev 8

### *Software Requirements*

For software requirements, refer to the following chapters:

- Chapter 4, "Java Library"
- Chapter 5, "C Library"
- Chapter 6, "Java Reference VIM"
- Chapter 7, "HTTP Protocol VIM"

### *SDK Installation Instructions*

The working draft VIM SDK is provided as a ZIP file of the SDK source code. Please unzip this file in the desired directory and follow the instructions included with each component. Or, check your distribution to see if packaging is available for your operating systems, i.e. Linux, OpenSolaris.

Requirements include the following:

- Java Library and Java Reference VIM: JDK 5 or later from Sun and JRE Java 5 or later
- HTTP C VIM - In addition to the standard C run-time libraries provided by Visual Studio, the HTTP C VIM also uses the free, open source libcurl (<http://curl.haxx.se/libcurl/>) and Boost (<http://www.boost.org>) libraries. See their respective web sites for licensing information. To compile the code:
  - Download required boost libraries from <http://www.boost.org/downloads> page (<http://www.boost.org/users/download/>).
  - Extract the libraries to HTTP\_C\_VIM directory of the trunk.
  - Rename the extracted boot library directory from boost\_version to "boost". For Example, If you download boost version 1.37, rename the extracted directory from boost\_1\_37\_0 to boost.

---

## Chapter 3: XAM™ Library Configuration

This chapter provides a standard set of options and the formatting requirements for configuring a XAM™ Library. Topics include:

- Introduction
- Configuration File Format
- Configuration File Discovery
- XAM™ Library Configuration Behavior
- Stackable VIM Support
- XAM™ Library Configuration Namespaces
- Configuration File Example

### *Introduction*

The configuration options and formatting help to further define the responsibilities of the XAM Library (especially with respect to awareness of stackable VIMs and how to pass information among them). Moreover, the standard configuration format enables the XAM Library to migrate more easily between implementations. The standard formatting required also helps the systems administrators more easily manage the storage systems.

The configuration file should provide enough information for the XAM Library (or a VIM that contains aspects of VIM Management) to:

- Determine a list of libraries that can be loaded to resolve an *xsystemname*
- Allow a system administrator to configure a stack of VIMs
- Allow a system administrator to configure VIM-specific options for multiple VIMs (without requiring application modifications)
- Use a contextual name that is abstracted from the implementation language's loading semantics. For example, rather than using `com.sun.xxxx` that uses Java language semantics, use an



HTTP-like context parameter that uniquely identifies the VIM, such as StorageTek5800.

- Define a VIM discovery path if an XRI does not contain a VIM name or alias
- Preload a specific list of VIMs on initialization, such that the `.xam.vim.list.*` field namespace is populated for applications immediately (as opposed to “on-demand”).
- Dynamically control logging facilities

## Configuration File Format

For simplicity, we will use a properties file format that can be loaded from either C or Java (though Java has direct support for the format via the `Properties` class). Within the property file, we will use namespaces to help organize the file.

XAM configuration files will have the following attributes:

- Property fields should be specified as a name/value pair delimited by '=' on a single line, e.g.,  
  
`“org.company.my_field_name=my_field_value”`
- Property names may not exceed XAM\_MAX\_STRING characters.
- White space and commented lines (prefixed with the '#' character) are to be ignored.
- All defined fields are of type `xam_string`, unless otherwise specified, and use a simple type prefix (i.e., `<stype>.name=value`), where `stype` may be either `xam_string`, `xam_int`, `xam_double`, or `xam_datetime`.

**Note:** `xam_xuid` types may only be supported if the XAM Library implementation provides string-to-XUID conversion services.

## Configuration File Discovery

By default, the XAM Library will check for and load a configuration file named `xam.properties` in the working directory. Optionally, an environment variable named `XAM_CONFIG_PATH` may be used to specify an alternate path to the configuration file.

In either case, a string property field named `.xam.config.path` will be synthesized by the XAM Library object, and the value will contain the specified path to the configuration file.

This field may also be set directly by applications. If an application creates or modifies this field, the properties defined will be loaded from any newly specified configuration file, which allows applications to refresh the configuration contents on demand. If an error occurs reading the specified configuration file, the field will not be synthesized and a non-fatal error will be returned.

### *XAM™ Library Configuration Behavior*

Note that the configuration file is extensible. If a property is found in the configuration file, a corresponding property field with the same name/value will be synthesized on the XAM Library object. Because all XSystem instances inherit all XAM Library fields on construction, this allows for VIM-specific XSystem configuration options to be automatically supported by the XAM Library configuration file.

### *Stackable VIM Support*

A stackable VIM is defined to be a VIM that can be inserted between the XAM Library and other VIMs or between two VIMs (with the VIM above the stackable VIM also being a stackable VIM). The bottom-most VIM in a stack of VIMs must represent a storage system. The stackable VIM receives XAM API requests and forwards those requests to other VIMs via the VIM API to the VIM on top of which it is stacked.

An application should be unaware of stackable VIMs and the chain of responsibility that the VIM stack represents. When an application uses an XRI to specify an *xsystemname*, the *xsystemname* should be recognized and appropriately handled by the stackable VIM that the XAM Library loaded directly.

A request by an application to the XAM Library and objects that make up the [XAM-ARCH] must be processed according to the specification, regardless of whether one or more stackable VIMs is in the chain of responsibility for processing the request. On the other hand, once a request to an object that is represented through a stackable VIM is made, that request may change in one or more ways en route to the VIMs that represent a storage system at the other end of the chain of responsibility.

XAM Library implementations are required to support the configuration of stackable VIMs. A stackable VIM maintains the relationships of the XAM Library that states that each XSystem instance has a relationship to one VIM.

### *XAM™ Library Configuration Namespaces*

The following property namespaces will be recognized by the XAM Library:

- *.xam.config.vim.alias*.{*aliasvalue*}={*vim path or aliasvalue*}

An alias for a VIM. Each *aliasvalue* must be unique within the *.xam.config.vim.alias* namespace. In a property file, the value of this property name is the name of (or full path to) the library or class that must be loaded for the VIM. Aliases may reference other aliases.

**Example:** *.xam.config.vim.alias.centera=C:\VIMS\centera\_vim.dll*

- *.xam.config.vim.alias.stack*.{*aliasvalue*}={*alias1*}:*{alias2}*:...

Defines an alias for a chain of VIMs. The *aliasvalue* must be unique within the *.xam.config.vim.alias* namespace. The property value is a chain of aliases (stackable or not) which are ":" delimited. All of the values in the chain of responsibility must be identified with alias values, or the chain of responsibility is invalid.

**Example:** *.xam.config.vim.alias.stack=http:encrypter:centera*

- `.xam.config.vim.alias.{aliasvalue}.param.{name}={value}`

Refers to a name/value pair that should be added (or appended) to the XRI name/value pair list by the XAM Library if a particular *aliasvalue* is used in the XRI during connect. May be used for any *aliasvalue* defined in the *vim.alias* or *vim.alias.stack* namespaces.

- `.xam.config.vim.lookuporder.{n}={aliasvalue}`

Refers to the order of the VIMs for locating one that can leverage a particular XRI if multiple VIMs could be used (in the absence of a specified VIM name). The value *n* must be a positive integer value that is unique to the entire lookup order namespace and indicates the ordinal position of the VIM to use for the lookup (starting with either index 0 or 1).

- `.xam.config.vim.preload.{n}={aliasvalue}`

Preloads a VIM library during XAM Library initialization (instead of on first use), where *n* is a positive integer (starting with either index 0 or 1) that is unique to the entire namespace. The preload order should define the lookup order in the absence of both *vim.lookuporder* properties and a specified VIM name in the XRI (as the first component of VIM discovery should be to attempt to connect with existing loaded VIMs).

- `.xam.log.{option}={value}`

Supports all defined logging field controls defined by the XAM Library implementation, including:

- `xam_int..xam.log.level`
- `xam_int..xam.log.verbosity`
- `xam_string..xam.log.path`

**Note:** These fields should be recognized by the XAM Library whether they were synthesized by reading the configuration file, or if the corresponding fields were created by applications directly.

### Configuration File Example

An example of this property file format in action is as follows:

```
# C VIM Aliases
xam.config.vim.alias.emc=C:\VIMS\centera_vim.dll
xam.config.vim.alias.http=C:\VIMS\http_vim.dll
xam.config.vim.alias.encrypter=C:\VIMS\encryption_vim.dll

# Java VIM Aliases
xam.config.vim.alias.hp=com.hp.xam.VIMImpl
xam.config.vim.alias.sun=com.sun.honeycomb.xam.XAMImpl

# VIM Stacks
xam.config.vim.alias.stack.isolated_centera=http:emc
xam.config.vim.alias.stack.isolated_encrypt_hp=http:encrypter:hp
```

```
.xam.config.vim.alias.stack.isolated_sun=http:sun

# VIM Stack name/value pairs (passed to the http VIM in
the XRI)
.xam.config.vim.alias.isolated_centera.param.xsystem=1
0.241.44.10
.xam.config.vim.alias.isolated_centera.param.alias=iso
lated_centera
.xam.config.vim.alias.isolated_encrypt_hp.param.xsyste
m=10.241.44.20
.xam.config.vim.alias.isolated_encrypt_hp.param.alias=
isolated_encrypt_hp
.xam.config.vim.alias.isolated_sun.param.xsystem=192.1
68.1.1
.xam.config.vim.alias.isolated_sun.param.alias=isolate
d_sun

# VIM Lookup order (used in the absence of a specified
VIM name in the XRI)
.xam.config.vim.lookuporder.1=isolated_centera
.xam.config.vim.lookuporder.2=isolated_encrypt_hp
.xam.config.vim.lookuporder.3=isolated_sun

# Centera VIM XSystem specific properties
xam_int.com.emc.centera.maxconnections=99

# Logging
xam_int..xam.log.level=5
xam_int..xam.log.verbosity=5
xam.log.path=C:\xam.log
```

There are a variety of ways this configuration information can be used. Some standard use cases are discussed in the following sections:

- Aliasing VIM Implementations
- Leveraging Lookup Order for VIM Discovery
- Preloading VIM Libraries
- Defining VIM Stacks
- Using Application-Defined Configurations
- Configuring VIM-Specific Controls
- Specifying XAM™ Library Logging Controls

### Aliasing VIM Implementations

The following XRI refers unequivocally to the aliased entry in the property file, but the VIM class *com.sun.honeycomb.xam.XAMImpl* is devoid of the additional context that the XAM Library can obtain from the property file:

```
snia-xam//sun!192.168.1.1
```

The XAM Library is tasked with converting these aliases appropriately before the connect:

```
snia-xam://com.sun.honeycomb.xam.XAMImpl!192.168.1.1
```

### Leveraging Lookup Order for VIM Discovery

A XAM Library's VIM Manager can leverage the lookup order in cases where an XRI is supplied that does not use the optional *vimname* parameter. (Recall that the VIM Name is actually an optional field on an XRI and is ambiguous without the use of aliases.)

### Preloading VIM Libraries

The XAM Library's *.xam.vim.list.\** namespace is populated with VIM names as they are discovered or specified by the application. Some applications may benefit by having some or all VIM libraries preloaded when the singleton XAM Library object is initialized, rather than on demand. The application may then identify the list of available VIMs directly from the XAM Library object's VIM list.

### Defining VIM Stacks

This format is also useful for stacking VIM libraries. Aliases may be used to denote a chain of stackable VIMs with a target VIM as the final item. This target VIM may be a stackable VIM with its own VIM Manager.

The application can use the following XRI to refer to the Sun VIM-managed xsystemname 192.168.1.1 (serviced via the VIM library C:\VIMS\http\_vim.dll at location 192.168.1.100):

```
snia-xam://isolated_sun!192.168.1.100
```

Information configured by the system administrator is used by the XAM Library to construct the internal XRI:

```
snia-xam://
C:\VIMS\http_vim.dll!192.168.1.100?xsystem=192.168.1.1
&alias=isolated_sun
```

Within this XRI, a few important conversions and contextual information passes have taken place:

- The local VIM Manager has a location to resolve the location of the HTTP VIM server: 192.168.1.100
- When connecting to the HTTP VIM, the XRI contains enough information to resolve the target VIM that lies beyond the HTTP VIM. The HTTP VIM will connect to the VIM underneath it using: *snia-xam://com.sun.honeycomb.xam.XAMImpl!192.168.1.1*
- The expansion is repeatable for additional stackable VIMs.

The following XRI defines a VIM stack with automatic encryption of written XStreams and decryption of read XStreams, serviced by a stackable encryption VIM via an HTTP VIM server:

```
snia-xam://isolated_encrypted_hp!192.168.1.110
```

The XAM Library would construct the following XRI for the HTTP VIM:

```
snia-xam://
C:\VIMS\http_vim.dll!192.168.1.110?xsystem=10.241.4
4.20&alias=isolated encrypted hp
```

The HTTP VIM, in turn, could service the request to the encryption VIM, which would service the request to the final hp VIM. All the information required to traverse the topology would pass through each VIM in the stack.

Note the all underlying VIMs will receive a copy of all XAM Library configuration options during XSystem.connect. Having the configuration options allows stackable VIM libraries to appropriately convert one XRI into another, based on all the information provided by the administrator.

The aspect of parameterization within the configuration file and on the XRI syntax is extremely important with the stackable VIMs but can also be used by storage system VIMs, similar to the Java Servlet context parameters. Parameterization should be used to hide from the application programmer the storage ecosystem details that may change. System administrators must be able to add environmentally relevant information to XRIs without impacting application XRIs.

## Using Application-Defined Configurations

Applications may use the predefined field names to configure these values directly on the XAM Library object without using a configuration file. The application may allow users or system administrators to manage the configuration from within the application, either via the GUI or other means.

## Configuring VIM-Specific Controls

Because all XAM Library fields are copied onto all XSystem instances, the application or system administrator may configure a number of VIM-specific options via the properties file. VIMs that do not understand these options may simply ignore them.

**Example:** # Set the maximum number of open connections for all  
Centra VIM-managed XSystem instances

```
xam int.com.emc.centera.maxconnections=99
```

## Specifying XAM™ Library Logging Controls

SNIA has defined a set of XAM Library logging controls, which may be specified by the configuration file in the absence of any application level logging control facilities.

**Example:** # Enable XAM Library Logging

```
xam_int..xam.log.level=5
xam_int..xam.log.verbosity=5
xam_string..xam.log.path=C:\Logs\xam.log
```

---

## Chapter 4: Java Library

This chapter provides information about the Java Library that is provided with the XAM™ Storage System SDK.

This chapter discusses the following topics:

- Understanding What Is Provided
- Unpacking Your ZIP file
- Installing Compilers and Dependent Libraries
- Building the Code for Your Platform
- Installing the Binaries for Your Platform
- Installing a VIM
- Using the Java Library

### *Understanding What Is Provided*

Version 1.0 (in progress) is an implementation of the XAM Library and optional Toolkit.

The Java Library provides the following:

- An implementation of the XAM Library
- The Java XAM Interfaces as part of the distribution
- XAM Library and Interface implementations are distributed in `xamlib.jar`.
- Required Java Toolkit functions (`org.snia.xam.util`) are distributed in `xamtoolkit.jar`.
- An internal implementation of an `AbstractFieldContainer`, which may be used by Java-based VIM authors, which is neither standardized nor required at this time
- Support for the SNIA XAM SDK configuration standard.

**Note:** This version does **not** provide support for using C (.dll/.so) VIMs.

- Default logging using the standardized logging control properties. The current implementation uses Java 5 logging.
- Unit tests that test the XAM Library implementation.

**Note:** These tests do not provide VIM validation tools. VIM vendors/implementers must do their own validation and testing.

## Unpacking Your ZIP file

The working draft VIM SDK is provided as a ZIP file of the SDK source code. Please unzip this file in the desired directory and follow the instructions included with each component.

The directory contents for the Java XAM Library are listed as follows:

<b>/Java_XAM_Library</b>	
/bin	Temporary directory created and used during building
/deliverables	Output directory for the final build components that are created during the build
/doc	Output directory of the Java Doc process
/src	Java source for the library code
/org/snia/xam/base	Required source for the Java XAM Library
/org/snia/xam/util	Optional source for the Java Toolkit functions
/test	Source for the Java XAM Library Unit Tests
/org/snia/xam/	
/org/snia/xam/base	
/org/snia/xam/testvim	
/org/snia/xam/util	
build.xml	The Ant build script
XAMImplementation.config	The XAM Library configuration file used for unit tests
xam.test.props	A Java properties file to configure the unit test program



The directory contents for the Java Interfaces are listed as follows:

<b>/Java_Interfaces</b>	
/bin	Temporary directory created and used during building
/deliverables	Output directory for the final build components that are created during the build
/doc	Output directory of the Java Doc process
/src	
/org/snia/xam/	Source for the Java XAM Interfaces

### *Installing Compilers and Dependent Libraries*

#### **Installing JUnit**

JUnit is a simple framework for writing and running automated tests.

*To install JUnit:*

- 1 Download the latest version of JUnit (`junit.zip`) from <http://download.sourceforge.net/junit/>.
- 2 Install JUnit on your platform of choice.

Windows To install JUnit on Windows, follow these steps:

- 1 Unzip the `junit.zip` distribution file to a directory referred to as `%JUNIT_HOME%`.
- 2 Add JUnit to the class path:  

```
set CLASSPATH=%CLASSPATH%;%JUNIT_HOME%\junit.jar
```

Unix (bash) To install JUnit on Unix, follow these steps:

- 1 Unzip the `junit.zip` distribution file to a directory referred to as `$JUNIT_HOME`.
- 2 Add JUnit to the class path:  

```
export CLASSPATH=$CLASSPATH:$JUNIT_HOME/junit.jar
```
- 3 (Optional) Unzip the `$JUNIT_HOME/src.jar` file.
- 4 Test the installation by using either the textual or graphical test runner to run the sample tests distributed with JUnit.

**Note:** The sample tests are not contained in the `junit.jar`, but in the installation directory directly. Therefore, make sure that the JUnit installation directory is in the class path.

— For the textual TestRunner, type:

```
java junit.textui.TestRunner
junit.samples.AllTests
```

- For the graphical TestRunner, type:

```
java junit.swingui.TestRunner
junit.samples.AllTests
```

All the tests should pass with an "OK" (textual) or a green bar (graphical). If the tests don't pass, verify that `junit.jar` is in the the class path.

### Installing libcurl

libcurl is the free, multiprotocol file transfer library.

*To install libcurl:*

- 1 For general information, go to <http://curl.haxx.se/libcurl/>.
- 2 Download the software from <http://curl.haxx.se/download.html>.

## Building the Code for Your Platform

### Supported Platforms

Currently supported platforms are Solaris 10 (tested on X86 and SPARC), Windows (WIN32), Fedora 10, Open SUSE 10, Ubuntu 9, and Apple OS X Windows.

The Java Library has been tested with Sun Java 5 and Java 6 on the following platforms:

- Windows XP and Vista (32 bit)
- Linux (32 bit, Fedora 10, Open Suse 10, Ubuntu 9)
- Open Solaris (x86)
- Apple Macintosh OS X (10.5)

The SNIA XAM SDK Technical Working Group would like to hear about incompatibilities or success stories on platforms other than those listed here.

*To build the code for your platform:*

- 1 Make sure you have the required software:
  - **Ant**: <http://ant.apache.org> (tested with 1.7.0)
  - **Java**: <http://java.sun.com> (or the platform JVM provider) (Version 1.5 required; tested with 1.5.0\_07)
  - **JUnit** - <http://junit.org> (developed with JUnit version 3.8.1)
- 2 Set the `JAVA_HOME` environment variable to point to the root java installation directory of the compiler, for each platform must exist in the `PATH`.
- 3 Make sure the `JAVA_HOME` and the java directories containing the java compiler (javac) and java are in the `PATH`.

*To build the XAM Library using ant:*

- 1 Enter the `Java_Interfaces` directory.
- 2 Run `ant deliverables`.
- 3 Enter the `Java_XAM_Library` directory.
- 4 Run `ant deliverables`.

The following files are generated at the following locations:

- `Java_XAM_Library/deliverables/xamlib.jar`
- `Java_XAM_Library/deliverables/testvim.jar`

**Note:** `testvim.jar` is part of the unit test process and is not usable by applications; therefore, it must not be distributed.

- `Java_XAM_Library/deliverables/api/<JavaDocs>`
- `Java_XAM_Library/deliverables/xamtoolkit.jar`

**Note:** You do not need to include the Java Interfaces `jar` file in your class path; they are merged into the XAM Library `jar` (`xamlib.jar`).

### *Installing the Binaries for Your Platform*

Make sure that you put `xamlib.jar` and `xamtoolkit.jar` into your class path.

### *Installing a VIM*

See “Configuration File Example” in Chapter 3, “XAM™ Library Configuration”.

### **Running Tests**

**Note:** The tests do NOT require a working VIM.

*To run tests:*

- 1 Enter the `Java_XAM_Library` directory.
- 2 Run `ant test`

All of the unit tests will run; there should be no errors.

### *Using the Java Library*

#### **Obtaining an Instance of the XAM™ Library**

To use this Java library, the application must locate and include the `xamlib.jar` file in the application's class path. If the application uses any of the classes from the `org.snia.xam.util` package, it must also locate and include `xamtoolkit.jar`.

To obtain an instance of the XAM Library, do one of the following:

- Use the *XAMLibraryFactory* class, which is included in the `jar` files (recommended). This factory will instantiate the library and isolate your application from a XAM Library implementation. This instantiation will let you reconfigure your application to use the C XAM Library (via the Java Native Interface (JNI)) without needing to recompile. To use the *XAMLibraryFactory* to obtain a XAM Library instance, enter the following:

```
XAMLibrary lib = XAMLibraryFactory.newXAMLibrary();
```

This causes the XAM Library to be created without reading a configuration file, which makes it difficult to configure the library to specify VIMs, etc. To allow specifying a configuration file, the application should call:

```
XAMLibraryFactory.newXAMLibrary(
    XAMLibraryFactory.DEFAULT_XAM_LIBRARY,
    configFilepath );
```

- If the application chooses not to use the *XAMLibraryFactory*, use the following to obtain an instance:

```
XAMLibrary xam = new XAMImplementation(
    "XAM.config" );
```

If the library is created without a configuration argument (no argument constructor or the *XAMLibraryFactory*.getLibrary() method), the Java XAM Library will read the value of the environment variable *XAM\_CONFIG\_PATH*. If this points to a file, this file will be used as the XAM Config. If there is no argument and the *XAM\_CONFIG\_PATH* environment variable does not exist, the default value *xam.properties* will be used.

### Configuring the Library Initialization

Each XAM Library instance will load configuration parameters from a configuration file. The file must contain information to allow the library to find the referenced VIM. Additionally, the file may contain parameters to control logging.

Make sure the configuration file contains a line similar to:

```
.xam.config.vim.alias.TestVIM=org.snia.xam.testvim.TestVim
```

This line tells the library that a VIM named "TestVIM" is available by creating an instance of the class "org.snia.xam.testvim.TestVim".

**Note:** Make sure the class for the VIM is in the application's class path.

Logging Parameters Table 2 describes the parameters that control logging:

**Table 2 – Logging Parameters**

Parameter	Description
<i>.xam.log.path</i>	The file path indicating where to place the log file. <b>Note:</b> Make sure the application has write permission to this path location.
<i>.xam.log.verbosity</i>	An integer value 0-5 indicating the logging verbosity. These values are specified in the XAM Architecture Specification (see [XAM-ARCH]).
<i>.xam.log.debug</i>	An integer value 0 to MAX_LONG (2**32-1) indicating the amount of debug logging to be used. Debug logging in the Java XAM Library is minimal, and this value is most useful with a particular VIM.
<i>.xam.log.max.size</i>	The maximum size of a log file, in kilobytes, before it is rolled over to a new file. To cap the log file to 100KB, the value of this property is set to 100.
<i>.xam.log.max.rollovers</i>	The maximum number of log files to keep after they rolled over.
<i>.xam.log.append</i>	A value, if TRUE, will append to the latest log file; otherwise, the library will create a new file when the library is created.

Java Logging Use The following information is provided for application and VIM authors wishing to integrate with Java XAM Library use of Java Logging.

The Java XAM Library uses Java native logging (`java.util.logging`). Java logging is very versatile but doesn't completely match the functionality of XAM logging. The Java XAM Library simplifies its use of Java logging by creating a single instance of a logger for each log file specified via the logging properties. To accomplish this, a toolkit class (`org.snia.xam.util.LogManager`) is provided to encapsulate all the functionality:

```
public static Logger GetLogger( String path,
                               int    logSize,
                               int    rolloverCount,
                               boolean append )
    throws XAMException
```

The LogManager will create a logger and FileHandler to service the log file. Whenever log settings have changed (via a `XAMLibrary.setProperty()` or `XSystem.setProperty()` method call), the `GetLogger` method is called to update the FileHandler on the logger. It is important to realize that because a specific log file is global, all settings are shared by the logger. If a VIM is running in the same JVM as the Java XAM Library, the VIM is encouraged to use the `org.snia.xam.util.LogManager` class.

The LogManager uses Java logging in the following ways:

- One logger per uniquely named log file (based on the path property). This is different than using the classname to identify loggers.
- A single `FileHandler` attached to the logger.
- Loggers do NOT inherit handlers from the parent logger (in this case, the root logger).

If the application or VIM is running in another JVM instance or on different hosts, the log files may not be integrated. It is beyond the scope of this document to describe logfile unification across multiple processes or hosts.

### Using Classes from `XAMLib.jar` and `XAMToolkit.jar`

The two `jar` files contain the entirety of the Java XAM Library. The `jar` file `XAMLib.jar` contains all the classes required to implement the Java XAM Library, while the `jar` file `XAMToolkit.jar` contains the required tool kit functions specified by the [XAM-JAVA-API]. The toolkit functions include:

- `ExtendedFieldContainer` (XAM required)
- `ISO8601Date`
- `LogManager`
- `QueryFactory` (XAM required)
- `SASLUtills`
- `XAMIOException` (needed for the streams)
- `XAMLibraryFactory` (XAM SDK Required)
- `XStreamInputStream` (XAM required)
- `XStreamOutputStream` (XAM required)
- `XUIDIterator` (XAM required)

The toolkit functions are programmed using only the publicly defined XAM interfaces. Thus, the toolkit is usable with either the Java XAM Library or the C XAM Library (using the JNI bindings).

The core of the Java XAM Library (`XAMLib.jar`, containing packages `org.snia.xam.base`) has all of the classes required to implement the Java XAM Library. Applications should not use classes from this package. VIM authors may find the implementation class from `org.snia.xam.base` to be of use when implementing a Java base VIM, but be aware that you are tying your implementation to the Java XAM Library code base. This, in itself, is not a problem, but VIM authors are strongly encouraged to consider the long-term consequences of this dependency.

---

## Chapter 5: C Library

This chapter provides information about the C++ Library that is provided with the XAM™ Storage System SDK.

The following topics are discussed:

- Understanding What Is Provided
- Unpacking Your ZIP file
- Installing Compilers and Dependent Libraries
- Choosing a C compiler
- Installing the Binaries for Your Platform
- Installing a VIM
- Configuring the HTTP VIM

### *Understanding What Is Provided*

The following deliverables are provided with the C++ Library:

- **XAM C API Public Headers** - Defines the XAM C API, which may be used by application developers to interact with multiple Vendor Implementation Module (VIM) libraries.
- **XAM C Library** - Implements the XAM C API, to which applications may link in order to interact with multiple VIM libraries.
- **VIM C API Header** - Defines the VIM C API, which may be used by vendors for developing VIM libraries (i.e., VIMs).
- **XAM Java API Wrapper for C Library** - Implements the XAM Java API interfaces, which may be used by application developers in order to interact with multiple VIM libraries. The Java interfaces are implemented in `XAMLibrary.jar`, which uses the XAM C Library via Java Native Interface (JNI).
- **XAM Toolkits** - Includes simple XAM toolkit examples for both C and Java, which include standard XAM-defined utility methods.

- **HTTP VIM** - A VIM which proxies all XAM operations to a HTTP server using GET and POST. This can be useful in conjunction with the HTTP Protocol VIM, as it allows the C versions of the API to use the Reference VIM, which is written in Java.

## Unpacking Your ZIP file

The working draft VIM SDK is provided as a ZIP file of the SDK source code. Please unzip this file in the desired directory and follow the instructions included with each component.

The directory contents are listed as follows::

<b>/C_XAM_Library</b>	
/build_script	Contains ant build.xml and miscellaneous project files
/ <code>&lt;platform&gt;</code>	Platform-specific build and project files
/lib	Place to put any needed libraries that are not installed in the native OS. The cURL libraries should be manually placed here if they are not otherwise installed.
/objects	Generated as part of the build process
/ <code>&lt;platform&gt;</code>	Contains platform-specific object files
/src	Source code tree
/c	C API source code
/api	Implementation of the XAM interfaces in xam.h
/classlib	Class definitions for all XAM objects
/doc	Contains doxygen documentation on the source code (generated as part of the build)
/http_vim	HTTP proxy VIM
/include	Public header files
/jni	Java Native Interface (supports java api)
/logger	Logging framework and components
/posix	POSIX compatibility interface
/toolkit	XAM toolkit library implementation
/utils	Various XAM utilities
/vim	Representation of a VIM library (includes the vim.h interface for VIM development)
/java	Java API source code



/deliverables	Generated as part of the build process
/ <code>&lt;platform&gt;</code>	Deliverables for a specific platform
/include	Public headers
/lib	Binary deliverables
/doc	Doxygen documentation for public header files
/Java_Interfaces	Contains the Java interfaces required to build the XAM Java API wrapper jar for interfacing with the C XAM Library

## Installing Compilers and Dependent Libraries

### Installing JUnit

JUnit is a simple framework for writing and running automated tests.

*To install JUnit:*

- 1 Download the latest version of JUnit (`junit.zip`) from <http://download.sourceforge.net/junit/>.
- 2 Install JUnit on your platform of choice.

Windows

*To install JUnit on Windows, follow these steps:*

- 1 Unzip the `junit.zip` distribution file to a directory referred to as `%JUNIT_HOME%`.
- 2 Add JUnit to the class path:

```
set CLASSPATH=%CLASSPATH%;%JUNIT_HOME%\junit.jar
```

Unix (bash)

*To install JUnit on Unix, follow these steps:*

- 1 Unzip the `junit.zip` distribution file to a directory referred to as `$JUNIT_HOME`.
  - 2 Add JUnit to the class path:
- ```
export CLASSPATH=$CLASSPATH:$JUNIT_HOME/junit.jar
```
- 3 (Optional) Unzip the `$JUNIT_HOME/src.jar` file.
  - 4 Test the installation by using either the textual or graphical test runner to run the sample tests distributed with JUnit.

**Note:** The sample tests are not contained in the `junit.jar`, but in the installation directory directly. Therefore, make sure that the JUnit installation directory is in the class path.

— For the textual TestRunner, type:

```
java junit.textui.TestRunner
junit.samples.AllTests
```

- For the graphical TestRunner, type:

```
java junit.swingui.TestRunner
junit.samples.AllTests
```

All the tests should pass with an OK (textual) or a green bar (graphical). If the tests don't pass, verify that `junit.jar` is in the class path.

### Installing libcurl

libcurl is the free, multiprotocol file transfer library.

*To install libcurl:*

- 1 For general information, go to <http://curl.haxx.se/libcurl/>.
- 2 Download the software from <http://curl.haxx.se/download.html>.

### Choosing a C compiler

#### *Building the Code for Your Platform*

Currently supported platforms are (32- and 64-bit where applicable):

- AIX 5.1
- HP-UX-B.11.11
- HP-UX-B.11.23
- Linux-GCC3.3
- Linux-GCC4
- SunOS-5.8
- Win32Dev 8
- Win64Dev 8

**Note:** See the `build_script` directory for the current list of platforms.

*To build the code for your platform:*

- 1 Make sure you have the required software:
  - **Ant** - <http://ant.apache.org> (tested with 1.6.5)
  - **Doxygen** - <http://www.doxygen.org> (required for doc generation)
  - **Java** - <http://java.sun.com> (or the platform JVM provider)
- 2 Make sure the compiler for each platform exists in the *PATH*.
- 3 Set the *JAVA\_HOME* environment variable to point to the root java installation directory (e.g. `C:\j2sdk1.4.2_12`).
- 4 For doc generation, make sure that `doxygen.exe` is in the *PATH*.

- 5 For Windows, make sure that `devenv.exe` is in the *PATH* so that the ant builds to work correctly.

**Note:** For Visual Studio Express Edition on Windows, you must perform the build from the IDE directly (i.e., you may not use ant).

#### Building the XAM™ SDK with Visual Studio on Windows

- 1 Load the `C_XAM_Library/build_script/Win32Dev8/XAM_SDK.sln` project file.
- 2 Select either the **Win32** or **x64** build configuration. (The 64-bit build will produce a Win64Dev8 deliverable.)
- 3 Select **Build > Build solution**.

Binaries, public headers, and documentation are delivered to:

- `C_XAM_Library/deliverables/Win32Dev8/lib`
- `C_XAM_Library/deliverables/Win32Dev8/include`
- `C_XAM_Library/deliverables/Win32Dev8/doc`

#### Building the XAM™ SDK Using ant

- 1 Enter the `C_XAM_Library/build_script` directory.
- 2 Enter the following:

```
ant -v -f build.xml -Dplatform=<platform>
```

where `<platform>` is one of the directory names under the `build_script` directory. For example:

```
ant -v -f build.xml -Dplatform=Linux-GCC3.3
```

Files are generated at the following locations:

- `C_XAM_Library/deliverables/<platform>/lib`
- `C_XAM_Library/deliverables/<platform>/include`
- `C_XAM_Library/deliverables/<platform>/doc`

#### Installing the Binaries for Your Platform

Depending on your platform, you may install the binaries in a central location or a location for your own development.

#### Configuring the XAM™ Library

To configure the XAM Library:

- 1 Decide which options you want to configure. See “Configuration Options.”
- 2 Locate the `xam.properties` file in the working directory.
- 3 Use the supported namespaces and configuration file syntax to edit the `xam.properties` file. See “Supported Configuration Namespaces” and “Configuration File Syntax, respectively.”

- 4 For examples, refer to “Example Configuration File.”

### Configuration Options

You may use a properties file to configure any of the following XAM Library options:

- VIM-specific XSystem configuration fields
- Logging controls
- VIM name aliasing
- VIM stack definitions
- VIM lookup order priority
- Automatic VIM preloading
- Automatic generation of XRI name/value pairs for a given VIM alias

By default, the XAM Library checks for a configuration file named `xam.properties` in the working directory. To specify an alternate path to the configuration file, use the `XAM_CONFIG_PATH` env var. In either case, the XAM Library object synthesizes a string property field named `.xam.config.path`, with the value containing the specified path to the configuration file.

Applications may also set this field directly. If an application modifies or deletes this field, the newly-specified configuration file will reload the properties, which allows applications to refresh the configuration contents on demand.

**Note:** The configuration file is extensible. If a property is found in the configuration file, a corresponding property field with the same name/value is synthesized on the XAM Library object. Because all XSystem instances inherit all XAM Library fields on construction, VIM-specific XSystem configuration options are automatically supported by the XAM Library configuration file.

### Supported Configuration Namespaces

The C XAM Library recognizes the following string property namespaces:

- `.xam.config.vim.alias.{aliasvalue}={vim path}`

A VIM alias. The *aliasvalue* string may be specified in an XRI to reference the given *vim path*, which may be either the full path to a VIM library or a VIM name. Aliases may also reference other aliases.

Sample Alias Usage (using alias in place of the vim name):

```
snia-xam://aliasvalue!<connection string>
```

- `.xam.config.vim.alias.stack.{aliasvalue}={aliasvalue1}:{aliasvalue2}:...`

Identifies an alias for a chain of VIMs. The value of this property is a chain of vim names or aliases (stacks or that individual VIMs) are ":" delimited.

Sample Stack Alias Usage (using alias in place of vimname):

```
snia-xam://aliasvalue!<connection string>
```

- `.xam.config.vim.alias.{aliasvalue}.param.{name}={value}`

Refers to a name/value pair that should be added to the XRI name/value pair list by the XAM Library if a particular *aliasvalue* is used in the XRI during connect. May be used for any *aliasvalue* defined in the *vim.alias* or *vim.alias.stack* namespaces.

Sample Result (name/value pairs appended by the XAM Library):

```
snia-xam://aliasvalue!<connection
string>?name=value"
```

Multiple name/value pairs may be specified resulting in:

```
[?name1=value1[&name2=value2[&...]]]"
```

- `.xam.config.vim.lookuporder.{n}={aliasvalue}`

Refers to the order of the VIMs for locating one that can leverage a particular XRI if multiple VIMs could be used (in the absence of a specified VIM name). The value *n* must be a positive integer value that is unique to the entire lookup order namespace and indicates the ordinal position of the VIM to use for the lookup. The value of *n* must be both sequential (beginning with either 0 or 1) and  $\geq 0$ .

- `.xam.config.vim.preload.{n}={aliasvalue}`

Preloads a VIM library during XAM Library initialization (instead of on first use), where *n* is a positive integer (starting at 0 or 1) that is unique to the entire namespace. The preload order will define the lookup order in the absence of both *vim.lookuporder* properties and a specified VIM name in the XRI.

- `.xam.log.{option}={value}`

Supports all defined logging field controls, including:

- `xam_int..xam.log.level`
- `xam_int..xam.log.verbosity`
- `xam_string..xam.log.path`
- `xam_int..xam.log.max.size`
- `xam_int..xam.log.max.rollovers`
- `xam_boolean..xam.log.append`
- `xam_string..xam.log.message.filter`
- `xam_string..xam.log.component.filter`

### Configuration File Syntax

The XAM configuration file has the following attributes (similar to the java Properties format):

- Each configured field name is specified as a name/value pair delimited by '=' (e.g. my\_field\_name=my\_field\_value).
- White space and commented lines (prefixed with the '#' character) are ignored.
- All defined fields are of type xam\_string unless otherwise specified using a simple type prefix (e.g. <stype>.name=value), where stype may be either xam\_string, xam\_int, xam\_double, or xam\_datetime (note that xam\_xuid types are not currently supported).

### Example Configuration File

```
# C VIM Aliases
.xam.config.vim.alias.emc=C:\VIMS\centera_vim.dll
.xam.config.vim.alias.http=C:\VIMS\http_vim.dll
.xam.config.vim.alias.encrypter=C:\VIMS\encryption_vim.dll

# Java VIM Aliases
.xam.config.vim.alias.hp=com.hp.xam.VIMImpl
.xam.config.vim.alias.sun=com.sun.xam.VIMImpl

# VIM Stacks
.xam.config.vim.alias.stack.isolated_centera=http:emc
.xam.config.vim.alias.stack.isolated_encrypted_hp=http:encrypter:hp
.xam.config.vim.alias.stack.isolated_sun=http:sun

# VIM Stack name/value pairs (passed to the http VIM in the XRI)
.xam.config.vim.alias.isolated_centera.param.xsystem=10.241.44.10
.xam.config.vim.alias.isolated_encrypted_hp.param.xsystem=10.241.44.20
.xam.config.vim.alias.isolated_sun.param.xsystem=10.241.44.42

# Preloaded VIMs
.xam.config.vim.preload.1=emc
.xam.config.vim.preload.2=isolated_encrypted_hp

# Centera VIM XSystem specific properties
xam_int.com.emc.centera.retrycount=100

# Logging
xam_int..xam.log.level=5
xam_int..xam.log.verbosity=5
.xam.log.path=C:\xam.log
```

### *Installing a VIM*

Refer to the vendor-specific installation instructions.

## Configuring the HTTP VIM

The HTTP VIM can be configured using a configuration file or by using parameters passed as part of the XRI.

### HTTP VIM Parameters

Parameters that control the HTTP VIM can be passed as part of the XRI. The HTTP VIM recognizes the following parameters:

- *targetServer.ipAddress={IP address}* - The IP address to which the HTTP server should connect
- *targetServer.port={port}* - The port to which the HTTP server should connect
- *targetServer.vimname={vimname}* - The vimname of the VIM to be loaded by the HTTP server.

Example XRI (one string, with "\" indicating line continuation):

```
snia-xam://xam_vim_http_g!127.0.0.1?\\
targetServer.vimname=Remote&\\
targetServer.ipAddress=1.2.3.4&\\
targetServer.port=9923";
```

### HTTP VIM Configuration File

A properties file may be used to configure any of the following HTTP VIM options:

- IP Address of HTTP Server
- Port of the HTTP Server
- vimname of VIM to be loaded by the HTTP Server

The HTTP VIM will check for a config file named "vim.properties" in the working directory.

### Supported Configuration Namespaces

The HTTP VIM recognizes the following string property namespaces:

- *.org.snia.xam.vim.parameter.targetServer.ipAddress={IP address}* - The IP address to which the HTTP server should connect
- *.org.snia.xam.vim.parameter.targetServer.port={port}* - The port to which the HTTP server should connect
- *.org.snia.xam.vim.parameter.targetServer.vimname={vimname}* - The vimname of the VIM to be loaded by the HTTP server.

### Configuration File Syntax

The vim configuration file has the following attributes (similar to the java "Properties" format):

- Each configured field name is specified as a name/value pair delimited by '=' (e.g., my\_field\_name=my\_field\_value).

- White space and commented lines (prefixed with the '#' character) are ignored.

All defined fields are of type "xam\_string" unless otherwise specified using a simple type prefix (e.g. <stype>.name=value), where "stype" may be either xam\_string, xam\_int, xam\_double, or xam\_datetime (note that xam\_xuid types are not currently supported).

**Example Configuration File:**

```
.org.snia.xam.vim.parameter.targetServer.ipAddress=1.2
.3.4
.org.snia.xam.vim.parameter.targetServer.port=9923
.org.snia.xam.vim.parameter.targetServer.vimname=Remote
```



---

## Chapter 6: Java Reference VIM

This chapter provides information about the Java Reference VIM that is provided with the XAM™ Storage System SDK.

The following topics are discussed:

- Overview
- Unpacking your ZIP file
- Installing Compilers and Dependent Libraries
- Building the Code for Your Platform
- Installing the Binaries for Your Platform
- Configuring and Operating the XAM™ Reference VIM and Library
- Building and Running the Client Example
- Using Reference VIM Example Programs
- Building and Running Tests and Examples
- Running Your Application with XAM™

### *Overview*

The Reference VIM is part of the SNIA XAM™ SDK, which is intended to implement the semantics of the XAM Storage System in a correct and meaningful manner. The Reference VIM supplies functionality that allows an application writer to exercise each of the XAM API methods. Application writers can use the Reference VIM to develop and test an application, without needing access to a commercial XAM Storage System. Because the Reference VIM is a development tool and is not intended to be deployed as part of a robust product, it should not be distributed to end users. While the Reference VIM is stable and reasonably robust, it will not scale in terms of performance or large numbers of XSets.

## Summary of Features

For the reference implementation, the XSets are persisted as `xml` files. By default, the directory used as the repository is defined by `java.io.tmpdir`.<sup>1</sup> The `xml` file is named `XSet_<XUID>.xml`. Any XStreams persisted for the XSet will be in a directory named `XSet_<XUID>`.

The Java Reference VIM can specify a different repository location. See “Specification of a Different Reference VIM Repository Location” under “Running Your Application with XAM™” for more information.

## XAM™ API Features

The Java Reference VIM supports:

- The ability to specify a different repository location. See “Specification of a Different Reference VIM Repository Location” under “Running Your Application with XAM™” for more information.
- Anonymous and plain authentication, but does not support authorization level checking. When used without additional configuration, the username and password are hard coded to `testuser` and `testpasswd`, respectively. The Reference VIM may optionally be configured to use an external file containing clear text usernames and passwords. This configuration allows applications to experience access restrictions, such as may be present in a production system.
- Single-user mode. Multi-threaded applications and multiple clients have not been tested and will not work as specified in the [XAM-ARCH].
- XAM-specified retention model. The Reference VIM supports a minimal set of retention policies. These policies are intended to give programmers experience in using retention policies. Users should not expect that these policies will be available on other XAM Storage Systems.
- Setting autodelete and shred values and policies.
- Basic job support; only support for XAM Query jobs.
  - Can submit and halt jobs.
  - Can commit XSets that define a job before or after running the job.

**Note:** Commit of a XSet with a running job is **NOT** supported.

- Complete support for XAM Level 1 and XAM Level 2. The Reference VIM implements query using two important technologies: level 1 query and level 2 query.

---

1. The directory may also be changed by specifying an argument on the XSystem's XRI. See “Configuring and Operating the XAM™ Reference VIM and Library”.

- The level 1 query is implemented using Java DB, which is present in Java 6. Users of Java 5 may download Derby DB from the Apache project and put the Derby `jar` file in the class path.
- The level 2 query is implemented using the Lucene full text search engine. Users that want to use the textual search capabilities of the Reference VIM must download the Lucene core `jar` file and add it to the class path. If Lucene is not in the class path, the Reference VIM indicates that level 2 functionality is not supported by setting the appropriate property in the `XSystem` instance.
  - This functionality has been tested with Lucene 1.9 and Lucene 2.4.1. The Reference VIM only supports this functionality on `XStreams` of type `text/plain`. Level 2 functionality is implemented using the Lucene `StandardAnalyzer` which provides case-insensitive searches.
  - If you are adding support for level 2 queries to an existing `XSystem`, you may wish to delete the `ReferenceVimDB` directory before starting the Reference VIM. This will cause existing `XSets` to be indexed for level 2 functionality.
- Exporting and importing of `XSets`, including those containing streams. However, importing `XSets` with retention information is not guaranteed to work when importing `XSets` from a different VIM implementation. The Reference VIM will validate imported `XSets`, according to the [XAM-ARCH], and reject those `XSets` that contain retention settings that the Reference VIM is unable to honor.
- Example client program that uses the reference VIM (`ReferenceXSetClient.java`). This program demonstrates how a client would configure and use the Reference VIM.

### API Methods Supported

The Java Reference VIM supports all of the API methods:

- **VIM:** `createXSystem`
- **XSystem:** `connect`, `authenticate`, `abandon`, `close`, `createXSet`, `openXSet`, `deleteXSet`, `isXSetRetained`, `getXSetAccessTime`, `asyncOpenXSet`, and `asyncCopyXSet`
- **XSet:** `abandon`, `applyAccessPolicy`, `applyAutodeletePolicy`, `applyShredPolicy`, `close`, `commit`, `createProperty`, `deleteField`, `createXStream`, `openXStream`, `getField` (most of the field manipulation methods are supported), `openExportStream`, `openImportStream`, `submitJob`, `haltJob`, `createRetention`, `setBaseRetention`, `setRetentionEnabledFlag`, `setRetentionDuration`, and `setRetentionStarttime`, `holdXSet`, `releaseXSet`, `accessXSet`, `getXSetAccessTime`, `getActualAutodelete`, `getActualShred`, `asyncCommit`, and `asyncOpenXStream`
- **XStream:** `read`, `write`, `close`, `tell`, `seek`, `asyncWrite`, `asyncRead`, and `asyncClose`

- **XUID:** toBytes, toString, and equals. Also XUID creation and encoding/decoding and ability to instantiate XUID instance based on a XUID value.

## Unpacking your ZIP file

The working draft VIM SDK is provided as a ZIP file of the SDK source code. Please unzip this file in the desired directory and follow the instructions included with each component.

The directory contents are listed as follows:

|                                          |                                                                                   |
|------------------------------------------|-----------------------------------------------------------------------------------|
| <code>/Java_Reference_VIM</code>         |                                                                                   |
| <code>/build</code>                      | Temporary directory created and used during building                              |
| <code>build.xml</code>                   | The Ant build script                                                              |
| <code>/config/ReferenceVIM.config</code> | The XAM Library configuration file used while running the unit tests              |
| <code>/deliverables</code>               | Output directory for the final build components that are created during the build |
| <code>/doc</code>                        | Output directory for the Java doc process                                         |
| <code>/examples</code>                   | Example program source files                                                      |
| <code>/src</code>                        | Java Source for the Reference VIM                                                 |
| <code>/test</code>                       | Java Source for the unit tests                                                    |
| <code>xam.test.props</code>              | A Java properties file to configure the unit test program                         |

## Installing Compilers and Dependent Libraries

### Installing JUnit

JUnit is a simple framework for writing and running automated tests.

*To install JUnit:*

- 1 Download the latest version of JUnit (`junit.zip`) from <http://download.sourceforge.net/junit/>.
- 2 Install JUnit in the top level `/lib` directory (top level XAM that is), as it does not need to update the class path. The class path is updated automatically inside of the ANT script.

**Note:** Only JDK and ANT are required to build and run unit tests. During development, more software may be required, but those requirements are outside of the scope of this document.

Windows

*To install JUnit on Windows:*

- 1 Unzip the `junit.zip` distribution file to a directory referred to as `%JUNIT_HOME%`.
- 2 Add JUnit to the class path:

```
set CLASSPATH=%CLASSPATH%;%JUNIT_HOME%\junit.jar
```

Unix (bash)    *To install JUnit on Unix:*

- 1    Unzip the `junit.zip` distribution file to a directory referred to as `$JUNIT_HOME`.
- 2    Add JUnit to the class path:  
  

```
export CLASSPATH=$CLASSPATH:$JUNIT_HOME/junit.jar
```
- 3    (Optional) Unzip the `$JUNIT_HOME/src.jar` file.
- 4    Test the installation by using either the textual or graphical test runner to run the sample tests distributed with JUnit.

**Note:** The sample tests are not contained in `junit.jar`, but in the installation directory directly. Therefore, make sure that the JUnit installation directory is in the class path.

— For the textual TestRunner, type:

```
java junit.textui.TestRunner
junit.samples.AllTests
```

— For the graphical TestRunner, type:

```
java junit.swingui.TestRunner
junit.samples.AllTests
```

All the tests should pass with an OK (textual) or a green bar (graphical). If the tests don't pass, verify that `junit.jar` is in the class path.

## *Building the Code for Your Platform*

### **Supported Platforms**

Currently tested platforms are Solaris 10 (tested on X86 and SPARC), Windows (WIN32), Fedora 10, Open SUSE 10, Ubuntu 9, and Apple OS X.

### **Software Requirements**

The following packages are required to build or run the Reference VIM: ant, Derby, JUnit, Java, JavaCC, JavaMail, JavaBeans Activation Framework (JAF), and Lucene. Derby is required to build AND run. Lucene is required to build and optional to run. If Lucene is not in the runtime class path, Level 2 query is not supported at run time.

Ant    Download ant from <http://ant.apache.org>. This software was tested with version 1.7.0.

Derby    Download Derby from <http://db.apache.org/derby/> (if using JDK/JRE earlier than 1.6).

**Note:** Sun distributes this as Java DB with JDK/JRE 1.6.

JUnit    1    Download JUnit from <http://www.junit.org>.

- 2 Make sure you copy `junit-3.8.1.jar` in the `<sdk-install-dir>/trunk/lib` subdirectory.
- Java Download Java 1.5 (or later) from <http://java.sun.com> (or the platform JVM provider). This software was tested with version 1.5.0\_07.
- JavaCC Download JavaCC from <https://javacc.dev.java.net/> (Version 4.2 or later). This software is required to build the Reference VIM, but is not required at run time.
- JavaMail
- 1 Download JavaMail 1.4 or later from <http://java.sun.com/products/javamail>.
  - 2 Copy `mailapi.jar` or `mail.jar` to the standard extension library directory, which will vary depending on java version: `<jdk-home>/jre/lib/ext`.  
**OR**  
Download `mailapi.jar` or `mail.jar` to the `<sdk-install-dir>/trunk/lib` subdirectory.
- JavaBeans Activation Framework (JAF)
- 1 Download JAF from <http://java.sun.com/javase/technologies/desktop/javabeans/jaf/index.jsp>.  
**Note:** JAF is a standard extension of the Java platform. You must download this unless you are using Java SE 6.0.
  - 2 Copy the `activation.jar` file to the standard extension library directory or to `<sdk-install-dir>/trunk/lib`.
- Lucene
- 1 Download Lucene from <http://apache.org/lucene> (1.9 through 2.4.1).  
**Note:** This software provides optional functionality, which if present, will enable level 2 query. Only the `Lucene-core.jar` file is needed in the class path. It is recommended to use the latest version.
  - 2 Install this software in the `<sdk-install-dir>/trunk/lib` subdirectory.

### PATH Requirements

Make sure to set your *PATH* as follows:

- Set the *JAVA\_HOME* environment variable to point to the root java installation directory.
- Include the compiler for each platform in the *PATH*.
- Include the *JAVA\_HOME* and the java directories containing the java compiler (`javac`) and java in the *PATH*.

### Building the XAM SDK using ant

To build the XAM SDK using ant, follow these steps:

- 1 If you haven't already done so, download and install ant and java. Download the other required `jar` files and place them in the `<sdk-`

```
install-dir>/trunk/lib subdirectory: junit-3-8.1.jar,  
mailapi.jar, activation.jar.
```

- 2 Build the `xamlib.jar` (Java\_XAM\_Library) and `snia-xam.jar` (Java\_Interfaces).
- 3 Enter the `Java_Reference_VIM` directory.
- 4 Run `ant deliverables` to build the Reference VIM `jar` file. Deliverables will be generated at the following locations:

```
Java_Reference_VIM/dist/referenceVIM.jar -  
reference VIM jar
```

### Running the ant test

Build the additional build targets, as follows:

- 1 To build the tests, use the “`build_test`” target.
- 2 To build the tests and the reference VIM `jar` files, use the “`deliverables`” target.

**Note:** Retention tests are not run by default; they are run via “`ant test_retention`”.

### *Installing the Binaries for Your Platform*

Install all XAM `jar` files with other application `jar` files. Make sure you put them in the application’s class path.

### *Configuring and Operating the XAM™ Reference VIM and Library*

This section contains the following configuration topics:

- Basic Configuration
- User Configuration
- Access Control Policies
- Autodelete Configuration
- Autodelete and Shred Policies
- Retention Policies
- Import Processing
- Asynchronous Methods
- Reference VIM Logging
- Summary of Configuration Properties
- Scalability

## Basic Configuration

The tests and examples use one basic property file to provide configuration information: `./config/ReferenceVIM.config`. This XAM configuration file is passed to the XAM Library. It contains important property definitions, including the name of the VIM and its associated Java class name. This file must contain a property of the form `.xam.config.vim.alias.<VIM-Name>`. The last part of the property name has to match the name of the VIM. The property value must specify the VIM class name.

### Example :

```
.xam.config.vim.alias.SNIA_Reference_VIM=org.snia.xam.vim
.reference.ReferenceVIM
```

Logging properties can be specified as well. Please see “Java Library” in this guide or view the `README.txt` file in the `Java_XAM_Library` directory for more information.

## User Configuration

The Reference VIM is capable of being configured to use an external file describing additional users. Without configuration, a hardcoded user (`testuser`) and password (`testpasswd`) are available. The following example shows a basic username/password text file which may be used to provide additional authenticated users to a running instance of the Reference VIM.

```
user1 pass1
user2 pass2
user3 pass3
xyzzz plugh
```

**Note:** This file is a simple text file, and the passwords are not encrypted or protected. The intention is to provide users with the opportunity to experience XAM behavior in a multiple, user authenticated environment, not to provide a secure storage system! If you want to make this more secure, we suggest protecting the file with the appropriate file permissions.

To use the external user file, set the XAM system property to refer to the path of the file. For example, the following line may appear in your XAM Configuration file:

```
org.snia.xam.reference.passwd=/tmp/passwd.txt
```

The Reference VIM will not have the built-in username and password available if it has been configured to use the external user file.

## Access Control Policies

The Reference VIM will create a unique access control policy for each defined user. User policies grant read and write access to the named user; other users have no access.



The following example shows what access policies are created if the ReferenceVIM was created using the password file example in “User Configuration”:

**Table 3 – Access Policy Example**

| User Name    | XSystem Policy List                                                      | Policy Name                                  |
|--------------|--------------------------------------------------------------------------|----------------------------------------------|
| <i>user1</i> | <i>.xsystem.access.policy.list.user1</i>                                 | <i>user1</i>                                 |
| <i>user2</i> | <i>.xsystem.access.policy.list.user2</i>                                 | <i>user2</i>                                 |
| <i>user3</i> | <i>.xsystem.access.policy.list.user3</i>                                 | <i>user3</i>                                 |
| <i>xyzzz</i> | <i>.xsystem.access.policy.list.xyzzz</i>                                 | <i>xyzzz</i>                                 |
|              | <i>.xsystem.access.policy.list.org.snia.refvim.access.read.write.all</i> | <i>org.snia.refvim.access.read.write.all</i> |
|              | <i>.xsystem.access.policy.list.org.snia.refvim.access.read.all</i>       | <i>org.snia.refvim.access.read.all</i>       |

Setting the access policy on an XSet to “user1” denies all other authenticated users the ability to read or write the XSet. Also note that if a user is removed from the password file, any XSets having an access policy of that user are inaccessible by all users. To regain access to those XSets, the user must be added to the password file.

Two additional policies are defined. *org.snia.refvim.access.read.all* grants all users read access to the XSet, and *org.snia.refvim.access.read.write.all* grants read and write access to all XSets. Note that once *org.snia.refvim.access.read.all* has been applied and committed, even the original owner of the XSet is unable to modify the XSet.

**Note:** These access policies are provided for illustrative purposes; users are cautioned not to assume that this behavior will be present in systems supplied by storage vendors.

### Autodelete Configuration

The Reference VIM supports autodelete and allows you to configure the time period used by the autodelete daemon process. The property *org.snia.xam.reference.autodelete.period* contains the number of seconds between autodelete sweeps of the store. The default value is 300 seconds (five minutes). Shorter times are acceptable to provide a more aggressive processing and autodelete schedule, but doing so may result in reduced performance as the Reference VIM spends more time evaluating XSets for autodeletion criteria.

The value of *org.snia.xam.reference.autodelete.period* to a negative value will disable the autodelete process, although setting the autodelete property on the XSet is still supported.

Because of autodelete functionality, only a single instance of the autodelete daemon, per store path, is created. Multiple instances of the ReferenceXSystem using the same store path are all sharing the same autodelete daemon. Thus, only the first instance of a ReferenceXSystem will create an autodelete daemon with the specified autodelete period.

Be aware that setting autodelete on a simple XSet without setting any retention will result in the XSet being deleted the next time the autodelete daemon runs. Depending on where the autodelete daemon is in its timing cycle, the XSet could be deleted as soon as commit has completed. Generally, autodelete should not be used on XSets without some retention settings because the default management policy provides no retention. This may not be true of other XAM Storage Systems.

### Autodelete and Shred Policies

The Reference VIM provides three policies to set autodelete and shred settings on an XSet using XSet.applyAutodeletePolicy and XSet.applyShredPolicy settings. The policies provided are summarized in Table 4.

**Table 4 – Autodelete and Shred Policies**

| Policy Name                                             | Description                         |
|---------------------------------------------------------|-------------------------------------|
| <i>org.snia.refvim.disposition.autodelete</i>           | autodelete = TRUE,<br>shred = FALSE |
| <i>org.snia.refvim.disposition.autodelete.and.shred</i> | autodelete = TRUE,<br>shred = TRUE  |
| <i>org.snia.refvim.disposition.shred</i>                | autodelete = FALSE,<br>shred = TRUE |

The Reference VIM does not support external creation or modification of these policy parameters.

### Retention Policies

The Reference VIM provides a set of retention policies allowing the application to set retention criteria using policies instead of explicit settings. These policies may be used for base, event, or application-defined retentions, as shown in Table 5.

**Table 5 – Retention Policies**

| Policy Name                                  | Description                            |
|----------------------------------------------|----------------------------------------|
| <i>org.snia.refvim.retention.none</i>        | duration = 0, enabled = FALSE          |
| <i>org.snia.refvim.retention.one.second</i>  | duration = 1000 mS, enabled = TRUE     |
| <i>org.snia.refvim.retention.one.day</i>     | duration = one day, enabled = TRUE     |
| <i>org.snia.refvim.retention.thirty.days</i> | duration = 30 days, enabled = TRUE     |
| <i>org.snia.refvim.retention.one.year</i>    | duration = 365.25 days, enabled = TRUE |

The Reference VIM does not support external creation or modification of these policy parameters.

### Import Processing

According to the [XAM-ARCH], the Reference VIM will validate retention and disposition policies when the XSet is imported. Any error during policy validation will cause an appropriate exception to be thrown when closing the import XStream. Another effect of the import validation failing is to place the XSet in a corrupt state, making it unusable. At this point, the application may only abandon and close the XSet.

While it is permissible for an XSystem to make adjustments to its policies, or adjust XSet properties in such a way as to avoid violating retention criteria, the Reference VIM does neither. Unless the imported XSet's policy parameters match the Reference VIM's retention policy parameters, the import process will not successfully complete.

The following is a list of conditions that will cause the import process to fail:

- A policy name in the imported XSet is unknown in the Reference VIM XSystem.
- The importing XSet policy specifies a retention policy duration longer than that supported by the Reference VIM's policy of the same name.
- The importing XSet policy specifies a retention enabled differing from that supported by the Reference VIM's policy of the same name.

In all cases of the import process failing (when closing the import XStream), the XSet becomes corrupt.

If the XSet already exists in the Reference storage, additional processing takes place.

- If the binding attribute in the import XSet is different than the binding attributes of the previously stored XSet, a new XUID will be issued when the XSet is committed. The previously stored XSet is not affected.
- If the effective retention of the importing XSet is less than that of the previously stored XSet, the import will fail when closing the importXStream.

### Asynchronous Methods

The Reference VIM supports all the specified asynchronous operations. The methods are implemented with a single operation queue and worker threads to execute the operations. The default implementation provides a single worker thread, but more may be configured.

The XAM\_INT property `org.snia.refvim.async.thread.count` is used to configure the number of worker threads that the Reference VIM will use. For example:

```
xamLibraryInstance.createProperty(  
    "org.snia.refvim.async.thread.count", false, 2 );
```

configures the Reference VIM to use two worker threads to execute the asynchronous operations. Applications should treat this parameter with care; too many threads will result in degraded system performance. A few threads will provide the most benefit.

### Reference VIM Logging

The Reference VIM supports the XAM-specified logging property settings and integrates with the Java Logging as implemented by the Java XAM Library. The Reference VIM uses and requires the `org.snia.xam.util.LogManager` class.

The LogManager restricts the use of Java Logging to a single logger per logfile instance. With this pattern, it is possible to integrate log entries from multiple instances of the Reference VIM. When multiple Reference VIM instances are running in the same JVM and are using the same logging parameters, log entries will be integrated into a single log file.

XAM Levels correspond to Java Logging levels as shown in Table 6:

**Table 6 – XAM™ and Java Logging Levels**

| XAM Level | Java Level | Type of Information Logged                                                     |
|-----------|------------|--------------------------------------------------------------------------------|
| NONE      | OFF        | Nothing                                                                        |
| FATAL     | SEVERE     | Non-recoverable errors                                                         |
| ERROR     | SEVERE     | Recoverable errors                                                             |
| WARN      | WARN       | Notification of potential problems                                             |
| INFO      | INFO       | Configuration, performance; information of interest to application programmers |
| ALL       | FINEST     | Debugging information                                                          |

The Reference VIM will detect a change to the property `.xam.log.verbosity`. The following setting, `xam.log.verbosity xam_int 100`, will turn on method entry/exit trace when the logging level is at XAM\_LOG\_ALL. Other verbosity values have no effect at other levels of tracing (i.e., less than XAM\_LOG\_ALL).

**Note:** Configuration of logging when running the Reference VIM in a separate JVM or on different hosts is not within the scope of this document.

### Summary of Configuration Properties

Table 7 summarizes the Reference VIM configuration properties.

**Table 7 – Summary of Configuration Properties**

| Property Name                               | Type       | Description                                          |
|---------------------------------------------|------------|------------------------------------------------------|
| <i>org.snia.reference.passwd</i>            | XAM_STRING | Path to the username password file.                  |
| <i>org.snia.reference.autodelete.period</i> | XAM_INT    | Number of seconds between autodelete sweeps          |
| <i>org.snia.refvim.async.thread.count</i>   | XAM_INT    | Number of worker threads to execute async operations |

The Reference VIM also supports the XAM-specified logging properties.

### Scalability

The Reference VIM has been implemented with a focus on adhering to the [XAM-ARCH]. Because of this, no attempt has been made to make the Reference VIM scale in terms of throughput or large numbers of XSets. While the Reference VIM is reasonably robust and stable, it may generate run-time errors when large numbers of XSets have been stored. Programmers are cautioned to limit the maximum number of XSets in tests to less than 5,000.

### *Building and Running the Client Example*

Currently there is only one example client program:

`ReferenceXSetClient.java`. This program shows you how to write a simple XAM client that uses the Reference VIM. Because the reference VIM is run in the same JVM as the client program, the client program also must load and enable the reference VIM using the XAM Library. The example program can be run using ant or from the command line. The example program also depends on the two Reference VIM property files to provide configuration information. For more information, please see “Configuring and Operating the XAM™ Reference VIM and Library.”

### Building and Running the Example Using ant

Run `ant examples`.

This command builds and runs the example program. See “Viewing Build Example Output - ant”.

## Building and Running the Example From the Command Line

- 1 Set the default to the `Java_Reference_VIM` directory.
- 2 To build the client program, set the class path to include the two `jar` files specified above or specify the `jar` files using the `javac` commands `-cp` option.

For example, on Solaris:

```
javac -cp "./deliverables/referenceVIM.jar:../
Java_XAM_Library/deliverables/xamlib.jar"
examples/ReferenceXSetClient.java
```

- 3 To run the client program, you must also include the `Java_Reference_VIM` directory as part of the class path.

For example, on Solaris:

```
java -cp ".../deliverables/referenceVIM.jar:../
Java_XAM_Library/deliverables/xamlib.jar"
examples.ReferenceXSetClient
```

**Note:** If you do not specify an output directory, then the java compiler creates the class file in the same directory as the java file. By contrast, the ant examples target creates the class file in the `./build/classes/examples` directory. Make sure that you specify the class path so that you are running the desired `ReferenceXSetClient` instance.

## Viewing Build Example Output - ant

XAM XSet Client Example Program - Uses Reference VIM by default

```
Initializing VIM
Loading test properties from file: xam.test.props
Loading the VIM using the Java XAM Library.
VIM Configuration contained in file: ./config/
ReferenceVIM.config

=====
Client application example program testing:
Client application connecting to the VIM
Connection arguments: snia-xam://
SNIA_Reference_VIM!localhost

Client application authenticating user credentials

Client application creating & persisting XSet with
properties.
XSet created - XUID:
AAA6AwAeQQsxMjI4NDAXNzM0NTI3AXcWl90AupNA
XSet updated (binding change) - XUID:
AAA6AwAekJsxMjI4NDAXNzM0NTU0AngnwDsAPonD
Close & reopen XSet and test getting some property
values
test.boolean=true
test.double=1234.5
```

```
test.string=testing string....
Closing XSet and XSystem
=====
```

### Using Reference VIM Example Programs

The directory `examples` contains three simple programs to exercise basic features of the Reference VIM.

- `ReferenceXSetClient` - Connects to a Reference VIM instance and creates an XSet.
- `ExportXSetClient` - Connects to a Reference VIM instance and exports an XSet. The `ExportXSetClient` takes exactly one argument, a base 64-encoded XUID. This program will create a file with the name `<XUID>.dat` and put the canonical XSet data in it.
- `ImportXSetClient` - Connects to a Reference VIM instance and imports an XSet. The `ImportXSetClient` takes exactly one argument, a filename. The program expects that the file will contain canonical XSet data. This data is read and imported into the XSystem.

All three programs extend a simple class, `ExampleBase`, which contains enough structure to hold instance variables for a XAM Library and an XSystem instance.

### Building and Running Tests and Examples

#### Configuring Unit Tests

`xam.test.props` is a properties file used by tests and client example programs. It contains definitions for the xri connection argument along with the name of the XAM configuration file and the default authentication credentials.

The `xam.test.xri` property specifies the VIM connection xri value.

To edit the `xam.test.props` file, modify the `xam.test.xri` line by specifying an absolute directory path for the `dir` parameter. The path specified will obviously vary depending on the operating system.

Example *xri* values:

- Default value - no directory location specified:

```
xam.test.xri=snia-xam://
SNIA_Reference_VIM!localhost
```

- Directory location of `/home/mytest/xam_storage` (Unix)

```
xam.test.xri=snia-xam://
    SNIA_Reference_VIM!localhost?dir=/home/
    mytest/xam_storage
```

- Directory location of `C:\mytest\xam_storage` (Windows)

```
xam.test.xri=snia-xam://
  SNIA_Reference_VIM!localhost?dir=C:\mytest\xam_storage
```

See “Configuring and Operating the XAM™ Reference VIM and Library” for more information.

### Building and Running Tests Using ant

- 1 Specify directories used for `tmp` files and persisted XSets. By default, XSets are persisted to the `java.io.tmpdir`. This directory also contains any `tmp` files created. On Solaris, this defaults to something like: `/var/tmp`.

- To specify a different storage location for persisted XSets, modify the `xam.test.xri` parameter specified in the `xam.test.props` file and add a `dir` parameter value. An `xri` value specifying a storage location (when running on Solaris) would look like:

```
xam.test.xri=snia-xam://
  SNIA_Reference_VIM!localhost?dir=/home/mytest/xam_storage
```

**Note:** See “Running Your Application with XAM™” for more information on specifying a storage location.

- The value of `java.io.tmpdir` can also be set to specify the storage location for `tmp` files. It will also be used as the XSet persistence directory if no `dir` parameter value is specified in the `xri` connect information.

- 2 Run `ant test` to build (if necessary) and run all the JUnit tests EXCEPT retention. Other targets include:

- `test_retention` - runs the retention tests
- `clean_test` - cleans all test `.class` files
- `build_test` - builds test files
- also targets to run individual tests: `test_xsystem`, `test_xset`, `test_xstream`, `test_auth`, `test_retention`

- 3 Manually delete the XSet `xml` files and directories when you no longer want them (e.g., `XSet_*`). You may also have to delete temp copies of XStreams (e.g., `XStream_####.tmp`).

### Running Your Application with XAM™

The Reference VIM implements the XAM API but also implements a backing store. The repository created by the Reference VIM is based on the machine's file system. The reference VIM persists XSets and associated XStream objects as files. In a vendor-specific VIM, the backing store would probably be some entity outside the VIM implementation.

### Default Repository Location

By default, the Reference VIM persists XSets and XStreams to the temp directory specified by `java.io.tmpdir`. This directory is also where any



temp files are created. On Solaris, this defaults to something like `/var/tmp`. On Windows, it is likely to be `c:\temp`. To specify a different location for the repository, see “Specification of a Different Reference VIM Repository Location”.

### Directories and Files Created

To create unique file names, a file name-safe variant of the Base64-encoded XUID value is used. The Reference VIM uses the filename-safe Base64-encoded specified in RFC-4648 (Table 2) for filename. The Reference VIM only supports XUID interchange using the originally specified Base64 variant.

The XSet and its properties are persisted in an XML file. The format of the XML file follows the XAM specification's export format layout (see [XAM-ARCH]). In addition, data for each XStream is stored in a separate file. The XStream data files are located in a subdirectory that is also named using the XUID's filename-safe string. The XStream contents are stored in the same format that the application used when creating the XStream. No translation is performed on the data.

#### Files Created for a Persisted XSet

The following files are created for a persisted XSet:

- `XSet_<xuid-hex-string>.xml`

This file contains the XML description of the XSet and conforms to the XAM Export format.

- `XSet_<xuid-hex-string>`

This directory contains any XStream (stream field data for the XSet).

- `XSet_<xuid-hex-string>/XStream_####.data`

This stream field contains payload/data for a single XStream. Note that the field definition in the XSet XML file will contain the name of the associated XStream data file.

#### Temp Files Created

The following temporary files are created:

- `XSet_<xuid-string>.tmp`
- `XStream_####.tmp`

#### Database For Query Support

The `ReferenceVimDB` directory is created by the SQL database that the Reference VIM uses to support internal housekeeping and to query job support. Generally, this database is automatically created and maintained by the Reference VIM.

If an error occurs, you can rebuild the database by stopping the Reference VIM (or the application that has the Reference VIM embedded), by deleting the database directory, and by restarting the Reference VIM (or application). The database is automatically rebuilt when this process has been executed.

The `ReferenceVimDB` directory has a subdirectory called `contentIndex`, where Lucene segment data is stored. If the Lucene indexes become

corrupted, remove the `ReferenceVimDB` directory and restart the Reference VIM so that the database and content indexes are rebuilt.

**Note:** If the database has to be rebuilt, the Reference VIM start-up time will be longer than usual. It could take a few extra seconds to several minutes.

### Specification of a Different Reference VIM Repository Location

Because XRI's support the specification of a host, when an application uses an XRI that refers directly to a Reference VIM, the host specification is ignored. The Reference VIM is only connecting directly to a local embedded XAM Library. To connect across hosts, the http VIM is required (see Chapter 7, "HTTP Protocol VIM").

You can specify the location of the repository by providing a value for the `dir` on the XRI information passed to the `XSystem.connect` command. The absolute path that you specify must exist and be correctly formatted for the operating system. The user must have full privileges for the directory.

For example, to change the storage location used by the Reference VIM:

- 1 Set the directory to the `Java_Reference_VIM` sub-directory in the SDK installation.
- 2 Modify the `xam.test.xri` line of the `xam.test.props` file by specifying a absolute directory path for the `dir` parameter. The path you specify will obviously vary depending on the operating system.

See "Configuring and Operating the XAM™ Reference VIM and Library" for more information.

#### Example xri values:

- Directory location of `/home/mytest/xam_storage` (Unix)

```
xam.test.xri=snia-xam://  
    SNIA_Reference_VIM!localhost?dir=/home/mytest/  
    xam_storage
```

- Directory location of `C:\mytest\xam_storage` (Windows)

```
xam.test.xri=snia-xam://  
    SNIA_Reference_VIM!localhost?dir=C:\mytest\xam_s  
    torage
```

---

|                 |                                                                                                                                                                                                           |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>WARNING:</b> | Applications will be able to access the hosts entire disk structure using this XRI parameter. The Reference VIM does not provide any security restrictions on this setting. Users are strongly cautioned. |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

---

### Repository Maintenance

Running the tests or running sample programs over and over again may create large numbers of files or directories in the repository. Currently, there is no automated way to clean up the repository once the XSet files and directories are created. Periodically, it may be necessary to clean out the files

in the repository by manually deleting files and directories like XSet\_\* and XStream\_\*. Developers may want to specify a private location for the repository to isolate their files and make debugging and tracking XSets easier.

---

## Chapter 7: HTTP Protocol VIM

This chapter provides information about the HTTP VIMs that are provided with the XAM™ Storage System SDK. Topics include the following:

- Description
- Functionality
- Server Configuration, Installation, Building, and Testing
- Protocol VIM Use
- Java VIM Requirements

**Note:** For additional information on the architecture and protocol of the HTTP VIM, see Appendix B: “HTTP VIM Architecture”.

### *Description*

The HTTP Protocol VIM is a prototype that demonstrates how VIMs may be “stacked” and provides remote access via the HTTP protocol. Stacking VIMs is an architectural feature of the SNIA XAM™ API and is allowed because the VIM API is standardized within the SNIA XAM API specification (see [XAM-ARCH]).

Stacking VIMs allows systems to be configured with functionality that may not be supplied by a storage vendor's VIM. Examples of added functionality may include storage federation, compression, encryption, etc. The HTTP VIM provides remote access to VIMs, via HTTP, which may otherwise not have access. Currently, the SNIA XAM SDK contains a reference VIM which provides correct behavior but is unable to operate over networks. The HTTP Protocol VIM allows an application to use the Reference VIM across a network link.

The HTTP Protocol VIM is contained in two parts. The upper half (client) is a traditional VIM implementation that provides connectivity to the XAM Library. The client VIM portion translates all XAM API method calls into HTTP operations and sends them to the HTTP Protocol VIM Server. The HTTP Protocol VIM Server creates a local instance of an arbitrary VIM and relays HTTP VIM method calls to the VIM.

The HTTP Protocol VIM Server is written in the Java (1.5) programming language and can host any VIM that has also been written in the Java language. Currently, the HTTP Protocol VIM Server cannot host VIMs written in other languages.

This release of the HTTP Protocol VIM also supplies a Java Client HTTP VIM that will work with this server. The C XAM Library also supplies functionality for working with the HTTP Protocol VIM Server.

---

**CAUTION:** The HTTP Protocol VIM is a prototype and technology demonstration. It has not been engineered for performance, throughput scaling, or security. The HTTP Protocol VIM can function as a development tool but should not be deployed as part of a shipping product.

---

---

**CAUTION:** This protocol and VIM server do not currently support HTTPS. The only security available is the authentication provided by the target XSystem and the configured legal XRI list in `startup.properties`. There is no security provided on XObject handles, and clients could possibly guess new handle values and obtain access to objects to which they may not otherwise have access. Applications should close unused XObject instances when they are no longer used. Doing so frees up needed resources in the HTTP Protocol VIM Server.

---

## Functionality

All specified SNIA XAM™ methods are supported for these objects:

- XSystem
- XSet
- XStream
- XIterator
- XAsync

**Note:** Some methods, although specified by the [XAM-ARCH], are not supported by the Reference VIM and have not been fully tested (e.g., `XSystem.openXStream`).

## Server Configuration, Installation, Building, and Testing

This section includes the following topics:

- Configuring the HTTP VIM Implementation Target of the HTTP VIM Client
- Installing Required Runtime Libraries
- Building the Server

- Running Ant Tasks
- Verifying the Server
- Starting the Server

### Configuring the HTTP VIM Implementation Target of the HTTP VIM Client

The HTTP VIM Server configuration files must be located in the run time "base" directory. The code looks up the configuration files using the "." directory for the file named `startup.properties`. This configuration file is in a standard XAM configuration format file that allows you to adjust the server address and the configured VIMs, as follows:

- HTTP VIM Server address - identifies the host IP address to use, the port, and the server version. The default supplied is 127.0.0.1, which limits connections to those being sent to 127.0.0.1. To use an external interface, change this value to match the IP address of your external interface.
- Configured VIMs - lists the VIM classes and VIM stacks supported for the server.

Here is an example `startup.properties` file:

```
.org.snia.xam.http.server.host=127.0.0.1
xam_int..org.snia.xam.http.server.port=9923
.org.snia.xam.http.server.version=1.0
.org.snia.xam.vim.alias.Remote=dummyVIM|SNIA_Reference_VIM
.org.snia.xam.vim.alias.SNIA_Reference_VIM=org.snia.xam.vim.r
eference.ReferenceVIM
xam_int..xam.log.level=3
xam_int..xam.log.max.size=1024
```

Two other configuration files are also present in the directory:

- `handlers.properties` - This configuration file contains a list of handlers identified by name followed by the class that implements them. This file must not be modified. It will be integrated into the build in a later release.
- `HTTPTestClient.config` - This XAM configuration file is used by the unit test program and its instance of the XAM Library. Your application will use a different one.

### Installing Required Runtime Libraries

To install required runtime libraries, get the following external `jar` file libraries and place them in the `HTTP_Protocol_VIM/lib` directory:

- `concurrent.jar` - public thread library often used for thread pools, etc. (<http://g.oswego.edu/dl/classes/EDU/oswego/cs/dl/util/concurrent/intro.html>)
- `jetty04.2.27.jar` - Jetty Web Server (<http://jetty.mortbay.org/jetty/>)
- `servelet.jar` - Java Servlet framework (<http://java.sun.com>)

### Building the Server

This project requires Java 5 (or later) and the ant build tool (<http://ant.apache.org/>). After retrieving the required library files and ensuring that Java and ant are installed, you may begin building the HTTP Protocol VIM Server.

To build the server:

- 1 Build the Java XAM Java Interfaces by executing **ant deliverables** in the `Java XAM Interfaces` directory.
- 2 Build the Java XAM Library by executing **ant deliverables** in the `Java XAM Library` directory.
- 3 Build the Reference VIM by executing **ant deliverables** in the `Java Reference VIM` directory.
- 4 Build the HTTP server by executing **ant** in the `HTTP Protocol VIM` directory.

### Running Ant Tasks

The following ant tasks are described as follows:

- **ant** - Builds the Isolation VIM `jar` and places the resulting `jar` file in `./deliverables`
- **ant clean** - Cleans up and remove all generated files
- **ant docs** - Builds the associated Java docs
- **ant server** - Executes a default configuration with the Reference VIM (using `/tmp` for storage)
- **ant test** - Executes unit tests against a currently running server

### Verifying the Server

After building the HTTP VIM Protocol server, you may run the unit tests to verify the server. To do so, you will need two command line shells.

To verify the server:

- 1 In the first command line shell, start the default server:

```
ant server
```

Entering this command produces a listing of log output as the server register handlers for each VIM method. Successful startup of the server will conclude with the lines similar to:

```
[java] 11:39:00.942 EVENT Started SocketListener on
127.0.0.1:9923
[java] 11:39:00.942 EVENT Started
org.mortbay.http.HttpServer@6d75
```

- 2 In the second command shell, execute the unit tests with the command:

```
ant test
```

Successful completion of the unit tests will produce output similar to:

```
[java] .computer.name.local File System
[java]
[java] Time: 46.491
[java]
[java] OK (33 tests)
[java]
```

### Starting the Server

To start the server:

- 1 Make sure that all of the runtime libraries are in your *classpath*.
- 2 Make sure that all of the properties files are set up properly and are located in the "current" directory.
- 3 To run the server, execute **ant server** within the HTTP Protocol VIM directory. A default configuration for the Reference VIM is available in this directory.

### Protocol VIM Use

To use the HTTP Protocol VIM from your application, do the following:

- 1 Add the protocol VIM jar file (*Isolation\_VIM\_Java.jar*) to your *classpath*.
- 2 Use an XRI that points to the HTTP Protocol VIM machine, such as:

```
snia-xam://
Remote!localhost?targetServer.ipAddress=127.0.0.
1&targetServer.port=9923
```

**Note:** The host and VIM portion of the XRI are not interpreted by the HTTP Protocol VIM. To connect to the HTTP Protocol VIM, you must add the **targetServer.host** and **port** arguments to your XRI. These arguments tell the HTTP Client VIM where to find the HTTP VIM Server.

- 3 Include a VIM alias to cause **Remote** to map to the HTTP Protocol VIM Client name. For example, the unit test client uses the following configuration item in its config file:

```
.xam.config.vim.alias.Remote=org.snia.xam.vim.http.
client.VIM
```

The server should alias **Remote** to point to the actual VIM to be instantiated. The vimname supplied in the XRI argument will be translated using the alias mechanism in the *server.properties* file.

- 4 If you wish, you can change the VIM name mapping contained in *startup.properties*. Changing the mapping allows your application to use any appropriate VIM or system names. Your application must also change its XRI, and the legal XRI list in



`startup.properties` must also be changed to match the new VIM name.

### *Java VIM Requirements*

The Java VIM requirements for use by the HTTP VIM Server include the following:

- Public, no parameter constructor
- Implements the SNIA Java Bindings XAM Library as top-level object
- No “proprietary” method implementations
- Implements XAM operations appropriately (these will pass through the HTTP VIM)

---

## Chapter 8: Error Codes

This chapter maps the C XAM™ error codes to Java XAM exceptions. The mapping was generated by examining the C XAM Library JNI class `XAMErrors.java`. When a mapping was not present, the code `sXAMException` is typically used (for instance, in the generic exceptions, like `XSetException`, `XStreamException`, etc).

**Note:** VIMs may override these codes for vendor-specific errors.

This chapter contains the following mappings:

- `XAMException`
- `FieldContainerException`
- `JobException`
- `XSetException`
- `XStreamException`
- `XSystemException`
- Non-Categorized C Errors

### *XAMException*

Table 8 contains the `XAMException` mapping:

**Table 8 – XAMException Mapping**

| Java Exception                              | C Error Code                          |   |
|---------------------------------------------|---------------------------------------|---|
| <code>AuthenticationException</code>        | <code>sAuthenticationException</code> |   |
| <code>AuthenticationExpiredException</code> | <code>sAuthenticationException</code> |   |
| <code>AuthorizationException</code>         | <code>sAuthorizationException</code>  |   |
| <code>FieldContainerException</code>        | <code>sXAMException</code>            | * |
| * Other status code possible                |                                       |   |

**Table 8 – XAMException Mapping**

| Java Exception                 | C Error Code               |   |
|--------------------------------|----------------------------|---|
| InsufficientResourcesException | sXAMException              | * |
| InvalidArgumentException       | sInvalidArgumentException  |   |
| InvalidOperationException      | sInvalidOperationException |   |
| InvalidXUIDException           | sInvalidXUIDException      |   |
| JobException                   | sXAMException              | * |
| ObjectInUseException           | sObjectInUseException      |   |
| XSetException                  | sXAMException              | * |
| XStreamException               | sXAMException              | * |
| XSystemException               | sXAMException              | * |
| * Other status code possible   |                            |   |

*FieldContainerException* Table 9 contains the FieldContainerException mapping.

**Table 9 – FieldContainerException Mapping**

| Java Exception             | C Error Code                |
|----------------------------|-----------------------------|
| FieldDoesNotExistException | sFieldDoesNotExistException |
| FieldExistsException       | sFieldExistsException       |
| FieldInUseException        | sFieldInUseException        |
| FieldReadOnlyException     | sFieldReadOnlyException     |
| InvalidFieldNameException  | sInvalidFieldNameException  |
| InvalidFieldTypeException  | sInvalidFieldTypeException  |
| MaximumFieldException      | sMaximumFieldException      |

*JobException* Table 10 contains the JobException mapping.

**Table 10 – JobException Mapping**

| Java Exception               | C Error Code             |  |
|------------------------------|--------------------------|--|
| JobCommandException          | sJobCommandException     |  |
| JobPermissionsException      | sJobPermissionsException |  |
| JobResourceException         | sJobResourcesException   |  |
| JobRunningException          | sJobRunningException     |  |
| * Other status code possible |                          |  |

**Table 10 – JobException Mapping**

| Java Exception               | C Error Code             |   |
|------------------------------|--------------------------|---|
| JobUnsupportedException      | sJobUnsupportedException |   |
| QueryException               | sXAMException            | * |
| * Other status code possible |                          |   |

*XSetException*

Table 11 contains the XSetException mapping.

**Table 11 – XSetException Mapping**

| Java Exception              | C Error Code                 |
|-----------------------------|------------------------------|
| HoldIdException             | sHoldIdException             |
| InvalidXSetModeException    | sInvalidXSetModeException    |
| PolicyNameException         | sPolicyNameException         |
| PolicyMismatchException     | sPolicyMismatchException     |
| RetentionValueException     |                              |
| XsetInaccessibleException   | sXSetDoesNotExistException   |
| XsetUnderRetentionException | sXSetUnderRetentionException |
| XsetUnderHoldException      | sXSetUnderHoldException      |
| XsetAbandonException        | sXSetAbandonException        |
| XsetCorruptException        | sXSetCorruptException        |

*XStreamException*

Table 12 contains the XStreamException mapping.

**Table 12 – XStreamException Mapping**

| Java Exception              | C Error Code                 |
|-----------------------------|------------------------------|
| InvalidXStreamModeException | sInvalidXStreamModeException |
| XstreamAbandonException     | sXStreamAbandonException     |
| XstreamCorruptException     | sXStreamCorruptException     |

*XSystemException* Table 13 contains the XSystemException mapping.

**Table 13 – XSystemException Mapping**

| Java Exception            | C Error Code                |
|---------------------------|-----------------------------|
| ConnectException          | sConnectException           |
| InvalidXRIException       | sInvalidXRIException        |
| VIMLoadException          | sVIMLoadException           |
| XsystemCorruptException   | sXSystemCorruptException    |
| XsystemAbandonException   | sXSystemAbandonException    |
| AsyncPendingException     | sAsyncPendingException      |
| AsyncHaltedException      | sXAM_XASYNC_HALTED          |
| XAMException              | sXAM_INVALID_HANDLE         |
| XsystemAbandonException   | sXAM_FILESYSTEM_ERROR       |
| AthenticationException    | sXAM_AUTH_DATA_NEEDED       |
| InvalidOperationException | sXAM_NOT_SUPPORTED          |
| JobException              | sXAM_NOT_A_JOB              |
| JobCommandException       | sXAM_JOB_INVALID_CMD_SYNTAX |

*Non-Categorized  
C Errors*

Table 14 contains the mapping for non-categorized C errors.

**Table 14 – Mapping for Non-Categorized C Errors**

| Java Exception            | C Error Code                |
|---------------------------|-----------------------------|
| XAMException              | sXAM_INVALID_HANDLE         |
| XsystemAbandonException   | sXAM_FILESYSTEM_ERROR       |
| AthenticationException    | sXAM_AUTH_DATA_NEEDED       |
| InvalidOperationException | sXAM_NOT_SUPPORTED          |
| JobException              | sXAM_NOT_A_JOB              |
| JobCommandException       | sXAM_JOB_INVALID_CMD_SYNTAX |

---

## Appendix A: Reference VIM Architecture

This appendix documents the architecture of the SNIA XAM™ Reference VIM. The Reference VIM is written in and leverages the object-oriented capabilities of the Java language. Most of the implementation is straightforward; however, parts of the implementation are rather complex and require further explanation, which is the purpose of this chapter.

As of this writing, the Reference VIM is complete and the structure and sequences documented here are extracted directly from the existing code. While much of this document has been written for this purpose, some of the content has been repurposed from the Reference VIM's `readme.txt`.

This document has the following major sections:

- Class Structure
- Operational Flow

An important point to consider when examining the class structure is the relationship between the `ReferenceXSystem` and single or multiple instances of managers. The most critical dynamic element to understand is the Reference VIM query processing.

### *Class Structure*

This section discusses the following classes with respect to the class structure:

- `XSystem`
- `XSet`
- `XStream`
- `Persistence Manager`
- `Policy`
- `Jobs`
- `DBManager`

## XSystem

The ReferenceXSystem class is the core of the Reference VIM. Each XSystem Instance is implemented with an in-memory instance of the ReferenceXSystem class. This object mediates the creation of top level XAM objects, as well as various manager classes to help perform XAM operations.

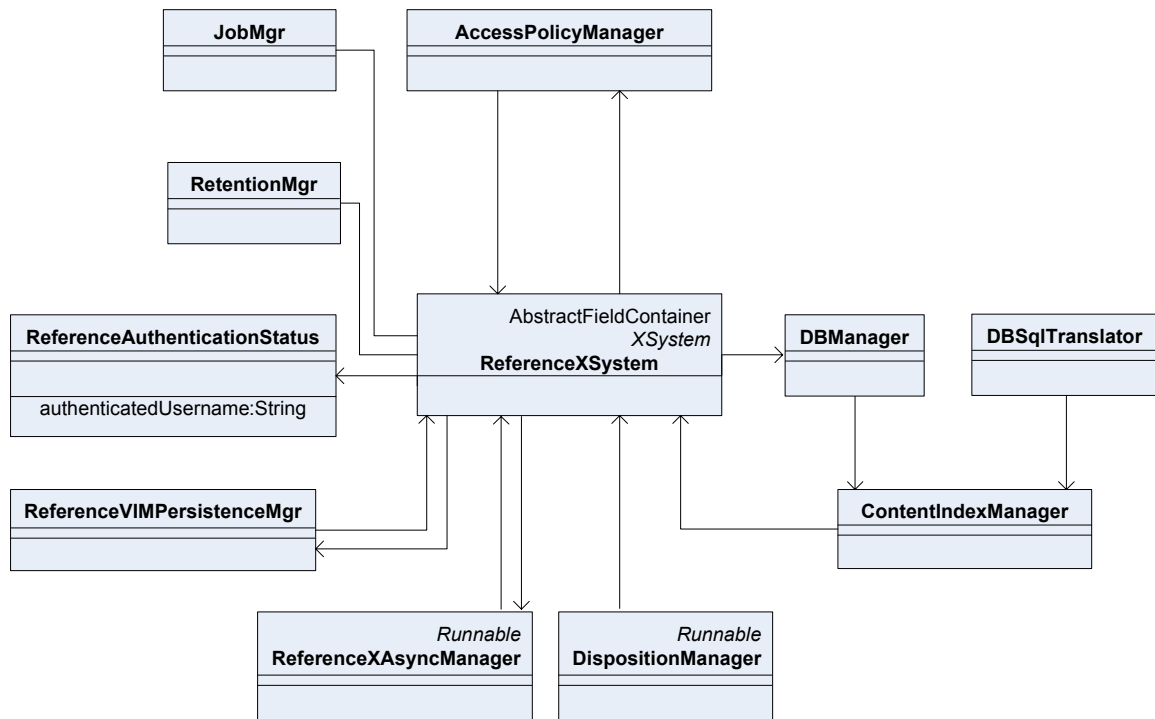
The Reference VIM uses manager instances to manage functionality that is too complex to implement “in line” with the ReferenceXSystem methods. Manager classes are described later in this document. The Reference VIM has been designed so that new implementations for functional areas can be integrated by changing a manager implementation.

The manager classes that are used are described in Table 15:

**Table 15 – Manager Classes**

| This manager class...      | Is responsible for...                                                                                                                                                                                                      |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| JobMgr                     | Creating new job instances and managing those jobs during run time.                                                                                                                                                        |
| AccessPolicyManager        | Checking the currently authenticated user and providing access and permission checks.                                                                                                                                      |
| RetentionMgr               | Managing XSet retention and retention policies and for providing data retention checks on XSets.                                                                                                                           |
| DispositionManager         | Implementing autodelete and shred functionality for XSets (if requested) and using the RetentionMgr at run time.                                                                                                           |
| ReferenceVIMPersistenceMgr | Storing XSets in the file system.                                                                                                                                                                                          |
| ReferenceXAsyncManager     | Maintaining a list of asynchronous operations and dispatching them to worker threads. This implementation is simple and just runs the operations on the appropriate object.                                                |
| DBManager                  | Implementing a database of XSet field values so that XAM Queries may be serviced. This implementation uses a combination of an SQL relational database management system and the Lucene search engine for Level 2 queries. |

Figure 1 shows the interaction between the manager classes.



**Figure 1 – Manager Classes**

### XSet

The Reference XSet is the Reference VIM's implementation of the XSet interface (see Figure 2, "Reference XSet"). This implementation uses the Java XAM Library's AbstractFieldContainer class to implement most of the field storage. All in-memory instances of property fields are stored using the Java XAM Library's Property implementation class. Streams are implemented using the Reference XStream implementations.

Each XSet tracks which state it is in, via an Operational State variable. This state matches the finite state machine (FSM) of XSets that are specified in the [XAM-ARCH]. States also include corrupt and abandoned. Closed XSets exist in the memory space of the Reference VIM.

When an XSet is committed, the Persistence Manager becomes involved and causes the XSet to store its contents to the file system. This implementation of the Reference VIM stores the XSet core as XML files, most of which are



compatible with the canonical export format. Streams are stored as unmodified, simple byte stream files.

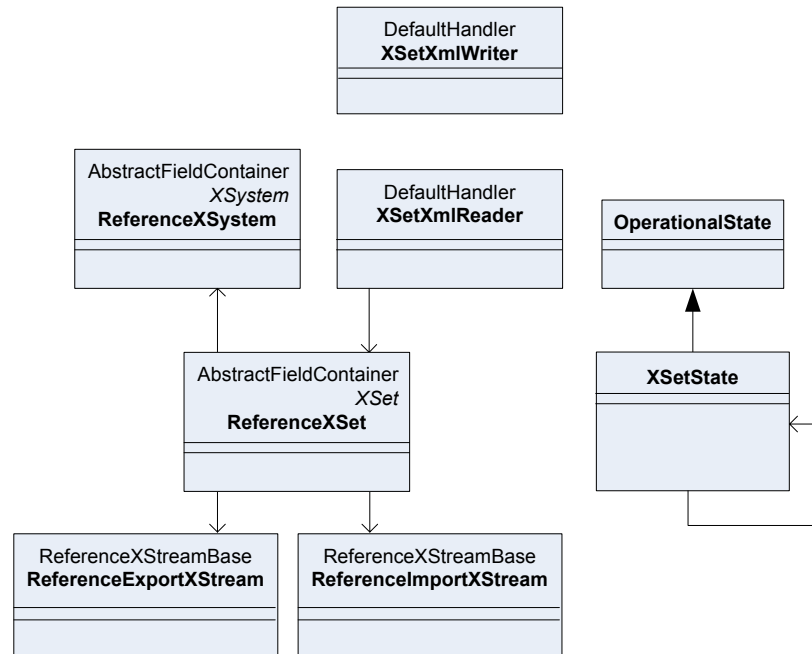


Figure 2 – Reference XSet

### XStream

XStreams are the most complicated entity related to XSets. An XStream implementation must represent normal XStream data and import and export XStream functionality. It must also maintain XStream attributes (see Figure 3, “Reference XStream”).

The ReferenceXStreamBase provides most of the attribute containment and state tracking. State is tracked with an operational state object similar to XSets. These states match the FSM in the [XAM-ARCH], including corrupt and abandon.

The actual byte stream is wrapped by a StreamContents object. Actual reading and writing of the contents is performed by this class.

ReferenceImportXStream and ReferenceExportXStream classes provide the unique functionality required for these objects. Each implementation makes use of the XSet's XML and XOP writers.

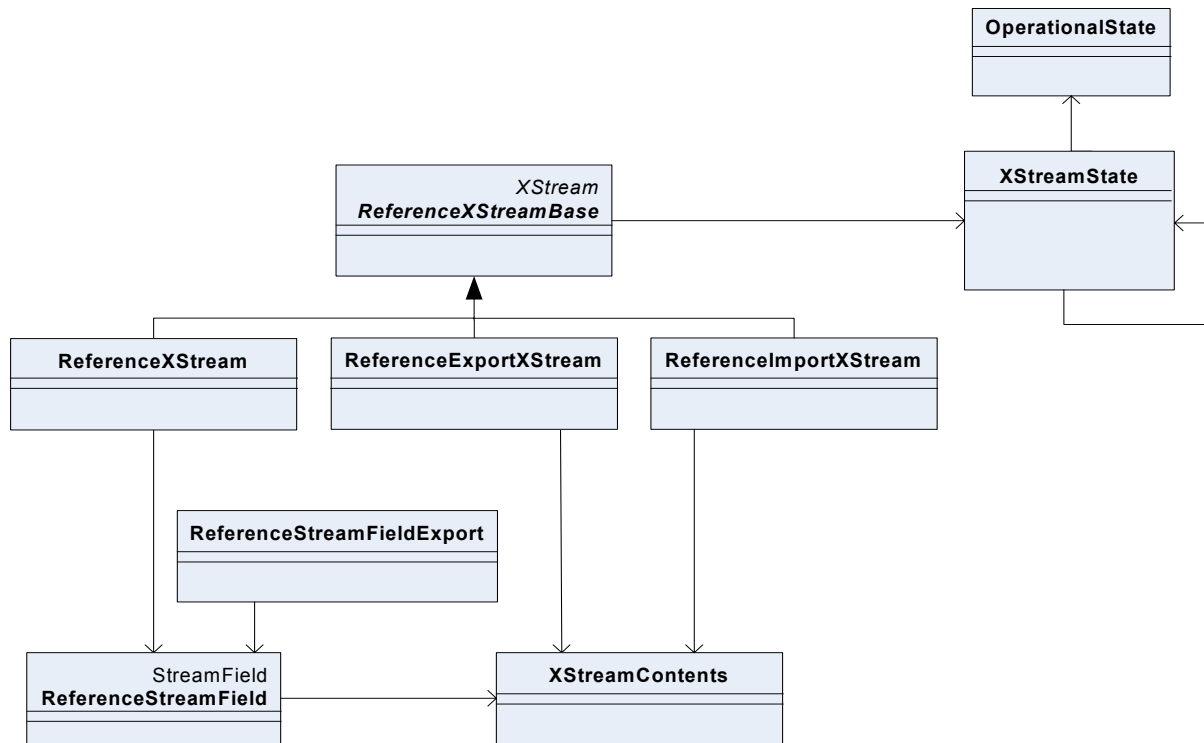


Figure 3 – Reference XStream

### Persistence Manager

The Reference VIM implements the XAM API but also implements a backing store. The repository created by the Reference VIM is based on the machine's file system. The reference VIM persists XSets and associated XStream objects as files. In a vendor-specific VIM, the backing store would probably be some entity outside the VIM implementation.

This section discusses the following topics with respect to the Persistence Manager:

- Default Repository Location
- Directories and Files Created
- Files Created for a Persisted XSet
- Temp Files Created
- Specification of a VIM Repository Location

#### Default Repository Location

By default, the Reference VIM persists XSets and XStreams to the temp directory specified by `java.io.tmpdir`. This directory is also where any temp files are created. On Solaris, the temp directory defaults to something

like `/var/tmp`. On Windows, it is likely to be `c:\temp`. To specify a different location, see “Specification of a VIM Repository Location”.

|                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|--------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Directories and Files Created              | <p>To create unique filenames, a filename-safe variant of the base64-encoded XUID value is used. The Reference VIM uses the filename-safe, Base64-encoded value specified in RFC-4648 (table 2) for filename. The Reference VIM only supports XUID interchange using the originally specified Base64 variant.</p> <p>The XSet and its properties are persisted in an XML file. The format of the XML file follows the export format layout in the [XAM-ARCH]. In addition, data for each XStream is stored in a separate file. The XStream data files are located in a subdirectory, which is also named using the XUID's filename-safe string. The XStream contents are stored in the same format that the application used when creating the XStream. No translation is performed on the data.</p>                                                                                                                                                          |
| Files Created for a Persisted XSet         | <p>The following files are created for a persisted XSet:</p> <ul style="list-style-type: none"> <li>• <code>XSet_&lt;xuid-string&gt;.xml</code> - Contains the XML description of the XSet. Conforms to the XAM Export format.</li> <li>• <code>XSet_&lt;xuid-string&gt;</code> - Directory containing any XStream (stream field data for the XSet).</li> <li>• <code>XSet_&lt;xuid-string&gt;/XStream_####.data</code> - Payload/data for a single XStream (stream field). Note that the field definition in the XSet XML file will contain the name of the associated XStream data file.</li> </ul>                                                                                                                                                                                                                                                                                                                                                         |
| Temp Files Created                         | <p>The following temp files are created:</p> <ul style="list-style-type: none"> <li>• <code>XSet_&lt;xuid-string&gt;.tmp</code></li> <li>• <code>XStream_####.tmp</code></li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Specification of a VIM Repository Location | <p>You can specify the location of the repository by providing a value for the "dir" on the XRI information that is passed to <code>XSystem.connect</code>.</p> <p>The absolute path specified must exist and be correctly formatted for the operating system. The user must also have full privileges for directory</p> <p>For example, to change the storage location used by the Reference VIM, do the following:</p> <ol style="list-style-type: none"> <li>1 First set the directory to the <code>Java_Reference_VIM</code> subdirectory in the SDK installation.</li> <li>2 Edit the <code>xam.test.props</code> file by specifying an absolute directory path for the "dir" parameter in the <code>xam.test.xri</code> line. The path you specify will obviously vary depending on the operating system.</li> </ol> <p>See “Configuring and Operating the XAM™ Reference VIM and Library” in Chapter 6, “Java Reference VIM” for more information.</p> |

*Example XRI Values*

Example XRI values include the following:

- Directory location of /home/mytest/xam\_storage (Unix)

```
xam.test.xri=snia-xam://
  SNIA_Reference_VIM!localhost?dir=/home/mytest/
  xam_storage
```

- Directory location of C:\mytest\xam\_storage (Windows)

```
xam.test.xri=snia-am://
  SNIA_Reference_VIM!localhost?dir=C:\mytest\xam_s
  torage
```

*Repository Maintenance*

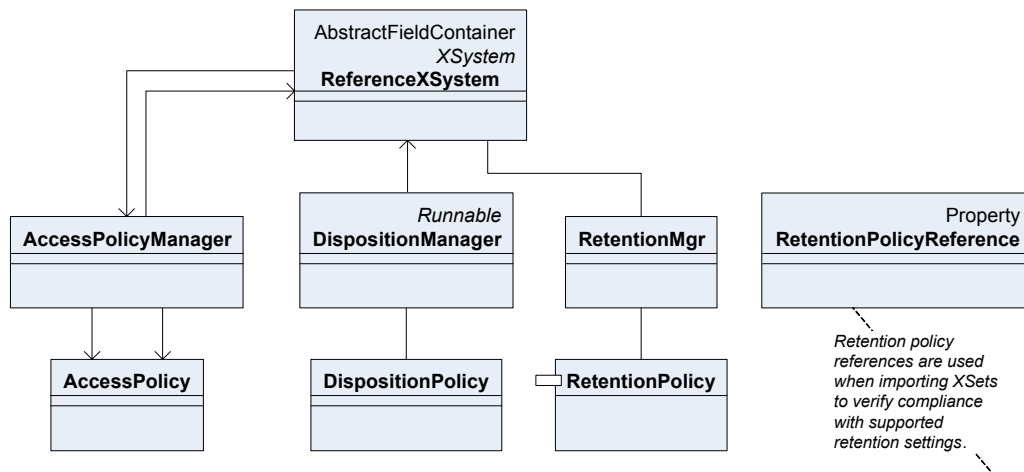
Running the tests or repeatedly running sample programs may create large numbers of files or directories in the repository. Currently, there is no automated way to clean up the repository once the XSet files and directories are created. Periodically, it may be necessary to clean out the files in the repository by manually deleting files and directories like **xSet\_\*** and **xStream\_\***. Developers may want to specify a private location for the repository to isolate their files and make debugging and tracking XSets easier.

**Policy**

The Policy system in the Reference VIM covers three major areas:

- Access Policies
- Disposition Policies
- Retention Policies

The policy system is shown in Figure 4.



**Figure 4 – Policy System**

**Access Policies** Each associated manager maintains a list of supported policies and publishes those policies in the XSystem property list as required by the [XAM-ARCH]. In this implementation of the Reference VIM, policies generally do not implement dynamic behavior, but rather serve as a tag to direct the behavior of the appropriate manager.

To implement policies derived from an external source (say an LDAP system, etc.), the manager for that functionality should be extended or modified.

**Disposition Policies** The Reference VIM provides three policies to set autodelete and shred settings on an XSet using the XSet.applyAutodeletePolicy and XSet.applyShredPolicy settings.

The disposition policies provided are summarized in the Table 16:

**Table 16 – Disposition Policies**

| Policy Name                                             | Description                      |
|---------------------------------------------------------|----------------------------------|
| <i>org.snia.refvim.disposition.autodelete</i>           | autodelete = TRUE, shred = FALSE |
| <i>org.snia.refvim.disposition.autodelete.and.shred</i> | autodelete = TRUE, shred = TRUE  |
| <i>org.snia.refvim.disposition.shred</i>                | autodelete = FALSE, shred = TRUE |

The Reference VIM does not support external creation or modification of these policy parameters.

**Retention Policies** The Reference VIM provides a set of retention policies that allow the application to set retention criteria using policies instead of explicit settings. These policies may be used for base, event, or application-defined retentions (see Table 17).

**Table 17 – Retention Policies**

| Policy Name                                  | Description                           |
|----------------------------------------------|---------------------------------------|
| <i>org.snia.refvim.retention.none</i>        | duration= 0, enabled = FALSE          |
| <i>org.snia.refvim.retention.one.second</i>  | duration= 1000 mS, enabled = TRUE     |
| <i>org.snia.refvim.retention.one.day</i>     | duration= one day, enabled = TRUE     |
| <i>org.snia.refvim.retention.thirty.days</i> | duration= 30 days, enabled = TRUE     |
| <i>org.snia.refvim.retention.one.year</i>    | duration= 365.25 days, enabled = TRUE |

The Reference VIM does not support external creation or modification of these policy parameters.

## Jobs

The Job Manager is responsible for creating job instances and running them (see Figure 5, “Job Manager”). A JVM instance contains a single static instance of the Job Manager. The Job Manager controls jobs across the entire JVM. The Job Manager ensures that a single job may be running on an XSet at any time and handles the job start, stop, and abort.

A job object links to the XSet that is running the job and tracks the status of the job specified by the Job FSM in the [XAM-ARCH]. A ReferenceJob object is not an XSet because the job-XSet association is not set up until XSet.submit is called. The ReferenceJob object is responsible for creating generic job fields and changing their read-only status as reflected by the job's run state.

The only job implemented in this release of the Reference VIM is the ReferenceQueryJob. This job interacts with the DBManager to cause the XAM QL command to be parsed and run. Job-specific fields (e.g., query results, etc.) are the responsibility of the job implementation. Results are stored in the XSet according to the [XAM-ARCH].

To extend the types of jobs, a new, specific job class must be implemented to extend ReferenceJob, and the job command needs to be published in the XSystem instance (ReferenceXSystem). The XSet.submit code path must be altered to extend to the new job, as this is not yet fully extensible.

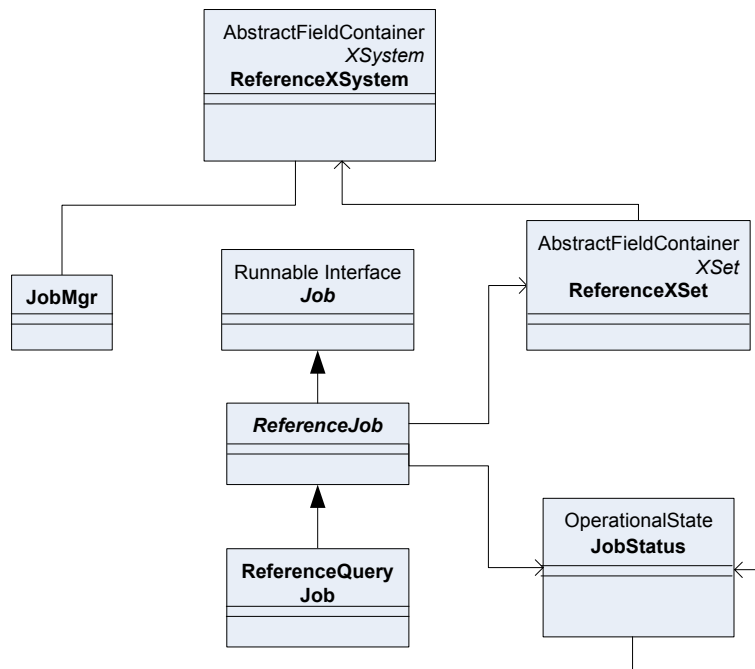
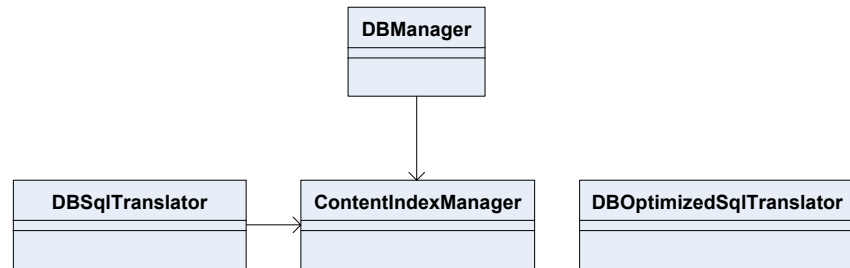


Figure 5 – Job Manager

## DBManager

The DBManager is responsible for maintaining XSet property and stream data in a manner which can later satisfy a query. This implementation of the

Reference VIM uses an SQL database (JavaDB AKA Derby) and search engine (Apache Lucene) to implement XAM Query compliance (see Figure 6).



**Figure 6 – DBManager**

When an XSet is committed, properties are stored to the database and any text/plain streams are indexed by the search engine. Since the expectation of the [XAM-ARCH] is that once committed an XSet is instantly available for query, this processing is done before XSet.commit completes.

Because SQL-based databases require columns to be homogeneously typed, it is impossible to create a single column per unique property/field. Because XAM is completely late bound and not as strongly typed as SQL databases, it is impossible to use a straightforward implementation. Additionally, most databases have limits on the number of columns allowable in a table. This limitation would not allow the Reference VIM to be able to scale to 16,000 fields per XSet.

The Reference VIM takes another approach. This approach stores property and field attributes in a simple table and relies on building a table that is specific to the query at the time the query is run. The DBManager reads each field from the XSet and stores the field attributes in a database table "XFieldValues".

|    | PROPNAME                                        | SVALUE                          | NU...ALUE | BVALUE | BINDING | REA...NLY | LENGTH | TYPE                           |
|----|-------------------------------------------------|---------------------------------|-----------|--------|---------|-----------|--------|--------------------------------|
| 1  | .xset.hold                                      | <NULL>                          | 0         | 0      | 0       | 1         | 1      | application/vnd.snia.xam.boole |
| 2  | .xset.management.policy                         | .org.snia.refvim.default.mgmt.p | 0         | 0      | 1       | 1         | 36     | application/vnd.snia.xam.strin |
| 3  | .xset.retention.base.enabled                    | <NULL>                          | 0         | 1      | 1       | 1         | 1      | application/vnd.snia.xam.boole |
| 4  | .xset.retention.base.starttime                  | 2009-03-13T10:31:08.577-07:00   | 0         | 0      | 1       | 1         | 29     | application/vnd.snia.xam.datet |
| 5  | .xset.retention.list.base                       | base                            | 0         | 0      | 1       | 1         | 4      | application/vnd.snia.xam.strin |
| 6  | .xset.retention.list.event                      | event                           | 0         | 0      | 1       | 1         | 5      | application/vnd.snia.xam.strin |
| 7  | .xset.time.access                               | 2009-03-13T10:31:08.577-07:00   | 0         | 0      | 0       | 1         | 29     | application/vnd.snia.xam.datet |
| 8  | .xset.time.commit                               | 2009-03-13T10:31:08.577-07:00   | 0         | 0      | 0       | 1         | 29     | application/vnd.snia.xam.datet |
| 9  | .xset.time.creation                             | 2009-03-13T10:31:08.575-07:00   | 0         | 0      | 1       | 1         | 29     | application/vnd.snia.xam.datet |
| 10 | .xset.time.residency                            | 2009-03-13T10:31:08.577-07:00   | 0         | 0      | 0       | 1         | 29     | application/vnd.snia.xam.datet |
| 11 | .xset.time.xuid                                 | 2009-03-13T10:31:08.577-07:00   | 0         | 0      | 0       | 1         | 29     | application/vnd.snia.xam.datet |
| 12 | .xset.xuid                                      | AAA6AwAe+MMxMjM20TY1NDY4NTc3EE4 | 0         | 0      | 1       | 1         | 30     | application/vnd.snia.xam.xuid  |
| 13 | com.example.property.1236965468315.testStream   | <NULL>                          | 0         | 0      | 0       | 0         | 3072   | application/1                  |
| 14 | com.example.property.1236965468315.xam_boolean  | <NULL>                          | 0         | 0      | 0       | 0         | 1      | application/vnd.snia.xam.boole |
| 15 | com.example.property.1236965468315.xam_datetime | 2009-03-13T10:31:06.366-07:00   | 0         | 0      | 0       | 0         | 29     | application/vnd.snia.xam.datet |
| 16 | com.example.property.1236965468315.xam_double   | <NULL>                          | 0.01      | 0      | 0       | 0         | 8      | application/vnd.snia.xam.doubl |
| 17 | com.example.property.1236965468315.xam_int      | <NULL>                          | 1         | 0      | 0       | 0         | 8      | application/vnd.snia.xam.int   |
| 18 | com.example.property.1236965468315.xam_odd      | <NULL>                          | 1         | 0      | 0       | 0         | 8      | application/vnd.snia.xam.int   |
| 19 | com.example.property.1236965468315.xam_promote  | <NULL>                          | 1         | 0      | 0       | 0         | 8      | application/vnd.snia.xam.doubl |
| 20 | com.example.property.1236965468315.xam_string   | testSimpleWhere.value           | 0         | 0      | 0       | 0         | 21     | application/vnd.snia.xam.strin |
| 21 | .vnd.org.snia.xam.reference.access_policy       |                                 | 0         | 0      | 0       | 1         | 0      | application/vnd.snia.xam.strin |
| 22 | .xset.hold                                      | <NULL>                          | 0         | 0      | 0       | 1         | 1      | application/vnd.snia.xam.boole |
| 23 | .xset.management.policy                         | .org.snia.refvim.default.mgmt.p | 0         | 0      | 1       | 1         | 36     | application/vnd.snia.xam.strin |
| 24 | .xset.retention.base.enabled                    | <NULL>                          | 0         | 1      | 1       | 1         | 1      | application/vnd.snia.xam.boole |
| 25 | .xset.retention.base.starttime                  | 2009-03-13T10:47:48.965-07:00   | 0         | 0      | 1       | 1         | 29     | application/vnd.snia.xam.datet |
| 26 | .xset.retention.list.base                       | base                            | 0         | 0      | 1       | 1         | 4      | application/vnd.snia.xam.strin |
| 27 | .xset.retention.list.event                      | event                           | 0         | 0      | 1       | 1         | 5      | application/vnd.snia.xam.strin |
| 28 | .xset.time.access                               | 2009-03-13T10:47:48.965-07:00   | 0         | 0      | 0       | 1         | 29     | application/vnd.snia.xam.datet |
| 29 | .xset.time.commit                               | 2009-03-13T10:47:48.965-07:00   | 0         | 0      | 0       | 1         | 29     | application/vnd.snia.xam.datet |
| 30 | .xset.time.creation                             | 2009-03-13T10:47:48.958-07:00   | 0         | 0      | 1       | 1         | 29     | application/vnd.snia.xam.datet |
| 31 | .xset.time.residency                            | 2009-03-13T10:47:48.965-07:00   | 0         | 0      | 0       | 1         | 29     | application/vnd.snia.xam.datet |

Figure 7 – XFieldValues Database



The columns of the table are described as follows:

- *Propname* – The name of the field
- *Svalue* – The string equivalent value of XAM\_STRING, XAM\_DATETIME, and XAM\_XUID property types
- *NumValue* – The numeric value, as a double, for XAM\_INT or XAM\_DOUBLE property values
- *Bvalue* – XAM\_BOOLEAN property values
- *ReadOnly* – A Boolean indicating if the field is readonly
- *Binding* – The Boolean value indicating if the field is bound
- *Type* – The MIME type of the field
- *XUID* – The XUID of the XSet to which this field belongs

While the XFieldValues database stores the property values and field attributes, this table is not generally directly usable to service queries. While processing a query job, the DBManager subsystem constructs a temporary table with the structure required to execute the XAM query that has been translated into SQL. For details of these temporary tables, see “Operational Flow”.

## *Operational Flow*

The operational flow of the Reference VIM includes the following topics:

- Initializing an XSystem
- Importing an XSet
- Processing a Query

### **Initializing an XSystem**

When an application calls XAMLibrary.connect, the Reference VIM class creates the XSystem. According to the [XAM-ARCH], the XSystem is created first, then the library copies the library fields to the XSystem.

After the fields are copied, the library calls XSystem.connect, as shown in Figure 8, “XSystem Initialization”.

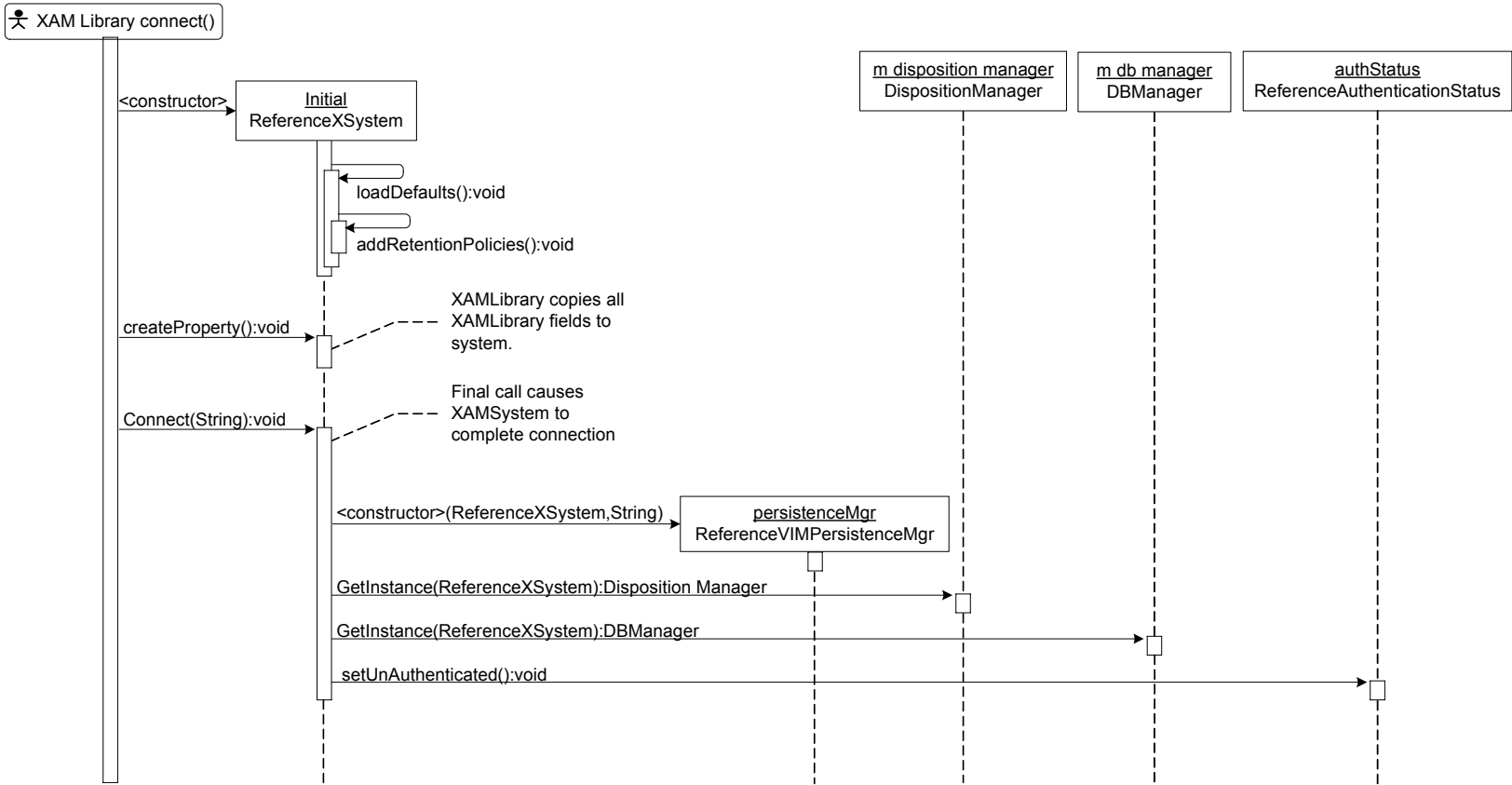


Figure 8 – XSystem Initialization

The connect method of the ReferenceXSystem is used to complete the creation of the ReferenceXSystem by creating or obtaining the references to the various managers that are needed to run and maintain the ReferenceXSystem instance.

### **Importing an XSet**

According to the [XAM-ARCH], the Reference VIM will validate retention and disposition policies when the XSet is imported. If an error occurs during policy validation, an appropriate exception is thrown when closing the import XStream. When the import validation fails, the XSet is also placed in a corrupt state, making it unusable. At this point, the application must abandon and close the XSet. Even though an XSystem may adjust its policies or may adjust XSet properties to avoid violating retention criteria, the Reference VIM does neither. Unless the imported XSet's policy parameters match the Reference VIM's retention policy parameters, the import process will fail.

When the import process fails (when closing the import XStream), the XSet becomes corrupt (see Figure 9, “Importing XSets”).

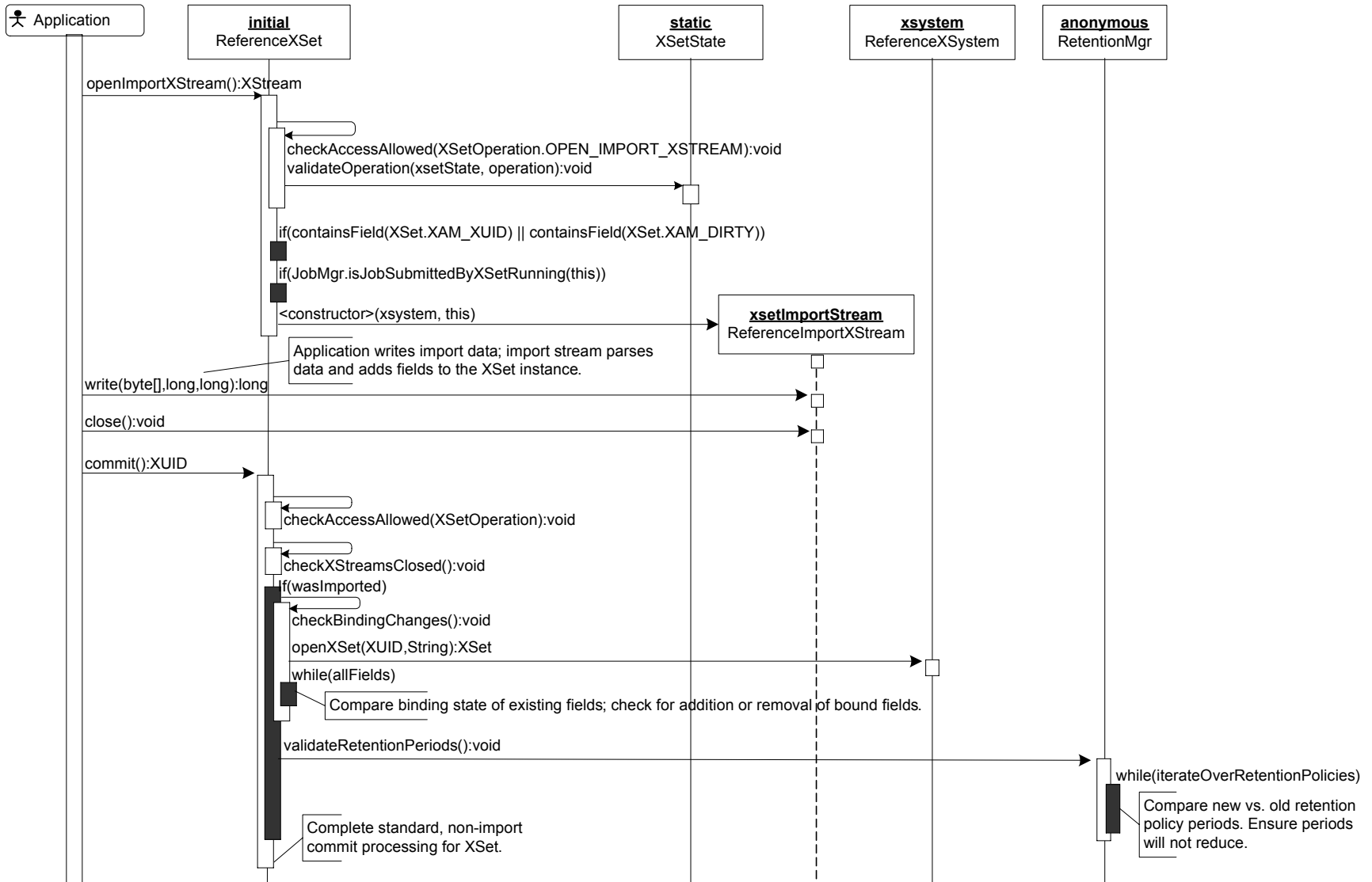


Figure 9 – Importing XSets

The following conditions will cause the import process to fail:

- A policy name in the imported XSet is unknown in the Reference VIM XSystem.
- The importing XSet policy specifies a retention policy duration longer than that supported by the Reference VIM's policy of the same name.
- The importing XSet policy specifies a retention enabled differing from that supporter by the Reference VIM's policy of the same name.

If the XSet already exists in the Reference storage, additional processing takes place.

- If the binding attribute in the import XSet is different than the binding attributes of the previously stored XSet, a new XUID will be issued when the XSet is committed. The previously stored XSet is not affected.
- If the effective retention of the importing XSet is less than that of the previously stored XSet, the import will fail when closing the importXStream.

### Processing a Query

As previously mentioned, in the static class structure of the DBManager, all Property values and field attributes are stored in the table "XFieldValues". The DBManager subsystem executes query jobs with these steps (see Figure 10, "Processing a Query"):

- 1 Parse the query using the JavaCC-based parser to produce a ParsedQuery (implemented in the *org.snia.xam.vim.reference.query* package).
- 2 Construct a temporary table to hold values for only this query job (createQueryTable method). The table's columns are:
  - Unique property and field attributes appearing in the query
  - Fields appearing in any "exists()" subclause of the query. The default value of this column is FALSE.
  - Unique Level 2 subexpressions.
- 3 Select all property values specified in the query, ordered by the XUID value. (polulateQueryTable method).
  - XAM requires all property values to be present in an XSet before it may be included in the result set.
  - XSets not containing all field values/attributes are not included in the temporary table.
- 4 Select all fields from the "exists()" clauses, setting their exists column values to TRUE.
- 5 For each Level 2 clause (if Lucene is installed), run Lucene queries matching the Level 2 subexpressions. Each document (XUID) returned from the Lucene queries are inserted into the temporary table.

- 6 Translate the parsed query (in the form of an abstract syntax tree) into SQL.
- 7 Execute the SQL against the temporary table, selecting the XUID values. Process the result set from the SQL, adding the XUIDs into the job XSet's result XStream.
- 8 Complete the query job.

Figure 10 shows the transformations that take place as a query is processed.

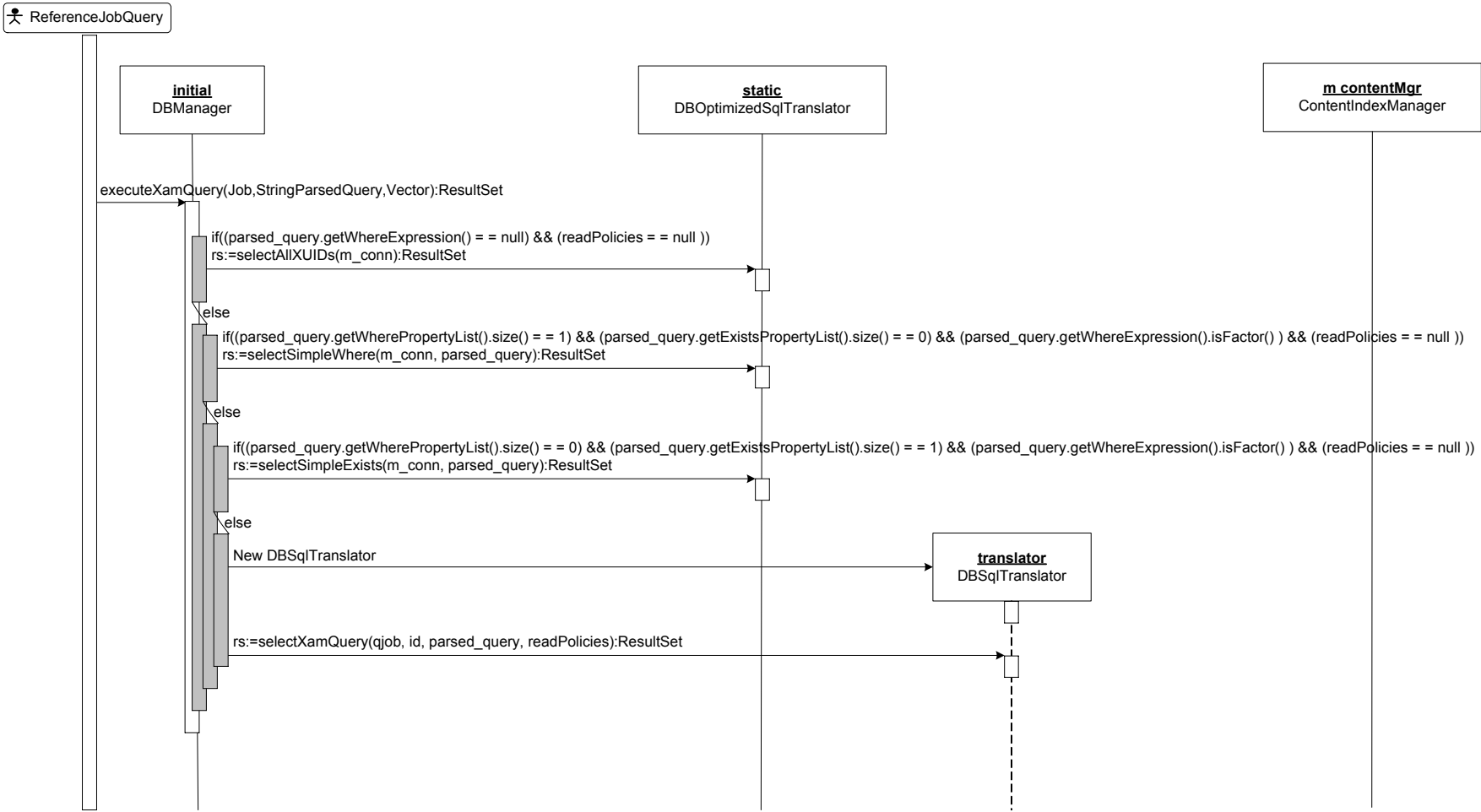


Figure 10 – Processing a Query

The following examples show the various transformations taking place as the query is processed.

*Example Query:*

```
select ".xset.xuid" where binding(".xset.xuid") and not
readonly("com.example.property.1236288089957.xam_int")
and
length("com.example.property.1236288089957.xam_double") =
8 and
typeof("com.example.property.1236288089957.xam_string") =

    'application/vnd.snia.xam.string' and
("com.example.property.1236288089957.xam_int" > 0 or
"com.example.property.1236288089957.xam_double" > 0 or
"com.example.property.1236288089957.xam_string" =

    'testSimpleWhere.value' or
"com.example.property.1236288089957.xam_xuid" =

    xuid('AAA6AwAeS+4xMjM2Mjg4MDg3Mzc5B1Lwu4gAE1gt') or
"com.example.property.1236288089957.xam_datetime" =

    date('2009-03-05T13:21:27.352-08:00'))
```

*Parsed Version of the Query:*

```
###T(F(binding(.xset.xuid)) AND
T(F(readonly(com.example.property.1236288089957.xam_int))
AND
T(F(length(com.example.property.1236288089957.xam_double)
= 8) AND
T(F(typeof(com.example.property.1236288089957.xam_string)
= 'application/vnd.snia.xam.string') AND
T(F(com.example.property.1236288089957.xam_int > 0) OR
T(F(com.example.property.1236288089957.xam_double > 0) OR
T(F(com.example.property.1236288089957.xam_string =
testSimpleWhere.value) OR
T(F(com.example.property.1236288089957.xam_xuid =
AAA6AwAeS+4xMjM2Mjg4MDg3Mzc5B1Lwu4gAE1gt) OR
F(com.example.property.1236288089957.xam_datetime =
'2009-03-05T13:21:27.352-08:00'))))))))
```



The following table is created and populated (see Figure 11):

|    | P0 | P1_RO | P2_BND | P3                | P4             | P5_TYP                          | P6              | P7_LEN | P8 | XUID               |
|----|----|-------|--------|-------------------|----------------|---------------------------------|-----------------|--------|----|--------------------|
| 1  | 6  | 0     | 1      | 2009-03-05T13:21: | AAA6AwAeS+4xMj | application/vnd.snia.xam.string | testSimpleWhere | 8      | 0  | AAA6AwAe850xMjM2Mj |
| 2  | 0  | 0     | 1      | 2009-03-05T13:21: | AAA6AwAeS+4xMj | application/vnd.snia.xam.string | testSimpleWhere | 8      | 0  | AAA6AwAe8C4xMjM2Mj |
| 3  | 4  | 0     | 1      | 2009-03-05T13:21: | AAA6AwAeS+4xMj | application/vnd.snia.xam.string | testSimpleWhere | 8      | 0  | AAA6AwAeBvYxMjM2Mj |
| 4  | 8  | 0     | 1      | 2009-03-05T13:21: | AAA6AwAeS+4xMj | application/vnd.snia.xam.string | testSimpleWhere | 8      | 0  | AAA6AwAeG2IxMjM2Mj |
| 5  | 2  | 0     | 1      | 2009-03-05T13:21: | AAA6AwAeS+4xMj | application/vnd.snia.xam.string | testSimpleWhere | 8      | 0  | AAA6AwAeKcwxMjM2Mj |
| 6  | 7  | 0     | 1      | 2009-03-05T13:21: | AAA6AwAeS+4xMj | application/vnd.snia.xam.string | testSimpleWhere | 8      | 0  | AAA6AwAeYQYxMjM2Mj |
| 7  | 3  | 0     | 1      | 2009-03-05T13:21: | AAA6AwAeS+4xMj | application/vnd.snia.xam.string | testSimpleWhere | 8      | 0  | AAA6AwAeMIXxMjM2Mj |
| 8  | 9  | 0     | 1      | 2009-03-05T13:21: | AAA6AwAeS+4xMj | application/vnd.snia.xam.string | testSimpleWhere | 8      | 0  | AAA6AwAeRbMxMjM2Mj |
| 9  | 5  | 0     | 1      | 2009-03-05T13:21: | AAA6AwAeS+4xMj | application/vnd.snia.xam.string | testSimpleWhere | 8      | 0  | AAA6AwAeT4xMjM2Mj  |
| 10 | 1  | 0     | 1      | 2009-03-05T13:21: | AAA6AwAeS+4xMj | application/vnd.snia.xam.string | testSimpleWhere | 8      | 0  | AAA6AwAeV4gxMjM2Mj |

Figure 11 – Query Results

The columns contain the property values as described in Table 18:

Table 18 – Table Columns and Value Property Names

| Column Name | Value/Property Name                      |
|-------------|------------------------------------------|
| P0          | com.example.property#.xam_int            |
| P1_RO       | readonly(com.example.property#.xam_int)  |
| P2_BND      | binding(.xset.xuid)                      |
| P3          | com.example.property#.xam_datetime       |
| P4          | com.example.property#.xam_double         |
| P5_TYP      | typeof(com.example.property#.xam_string) |
| P6          | com.example.property#.xam_string         |
| P7_LEN      | length(com.example.property#.xam_double) |
| P8          | com.example.property#.xam_double         |
| XUID        | The XSet XUID                            |

Note that column names with an alias (e.g., P0) represent a property value, whereas columns named with suffixes (P1\_RO) represent field attributes.

**9** Finally, translate the query into SQL and run it against the table.

```
select xuid from JOBID_35_20090305_21_21_41GMT
where
  (p2_BND=1) and
  ( not (p1_RO=1) and
  ((p7_LEN=8) and
  ((p5_TYP='application/vnd.snia.xam.string') and
  ((p0>0) or ((p8>0) or
  ((p6='testSimpleWhere.value') or
  ((p4='AAA6AwAeS+4xMjM2Mjg4MDg3Mzc5B1Lw4gAE1gt')
  or
  (p3= '2009-03-05T13:21:27.352-08:00'))))))))
```

Optimized Query

The DBManager provides an optimized SQL translator (see Figure 12) to handle simplified queries. Simplified queries are those in which the `where` clause consists of a single subexpression (FactorNode).

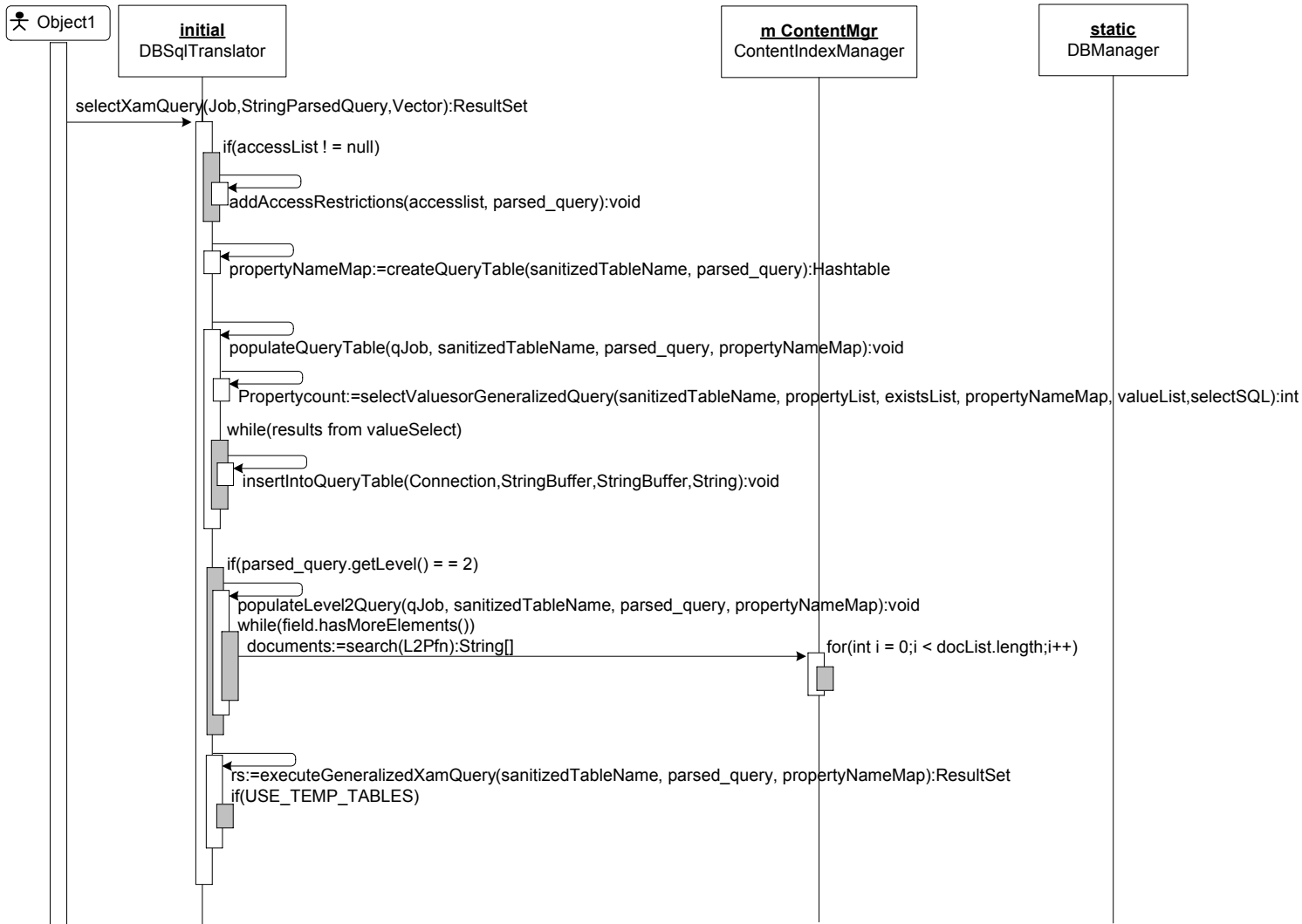


Figure 12 – Optimized Query

Examples of simplified queries:

- `select ".xset.xuid" where "com.example.prop" = 1234`
- `select ".xset.xuid" where "com.example.string" like '%foo%'`

When these queries execute, they are able to do so directly against the XFieldValues without needing to construct a temporary table and populate it. Because the access policy restrictions are applied early in the process, no user queries will execute in this code path.

This module was written for early implementations and testing of the Reference VIM structure. This functionality is being retained for future internal use. At this time, removing this functionality will not impair the Reference VIM.

## *Future Ideas*

This list contains some ideas for future modifications to the Reference VIM.

- To simplify the code, remove the optimized SQL submodule.
- Remove the SQL entirely and use the Lucene search engine to provide a more scalable solution.
- Externalize authentication and authorization roles.
- Add new jobs, perhaps the job chaining proposal being considered in the FCAS TWG.
- Make the DBManager so that other databases may be configured via external configuration.
- Externalize policy management, perhaps to an LDAP implementation.
- Rewrite the XML Parser. The current implementation is rather fragile; the syntax rules for the canonical format make it difficult to change.



---

## Appendix B: HTTP VIM Architecture

This appendix documents the architecture and protocol of the SNIA XAM™ HTTP VIM and contains the following topics:

- Terms and Scope
- Overview of HTTP VIM Design
- Java HTTP VIM Client
- Java HTTP VIM Server
- VIM Class and Library Load Operations
- VIM Wire Protocol
- Known Issues

### *Terms and Scope*

#### **Terms**

The following terms are provided as a convenience and an extension for terms that are particular to the HTTP VIM Design. For all terms, see [XAM-ARCH].

**Table 19 – Terms - HTTP VIM Design**

| Term          | Definition                                                                                                                                                                                                                                                                                                                                                                                         |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| HTTP VIM      | A “Stackable VIM” that is split into two halves. The “top” half of the VIM plugs into a XAM Library via the VIM API and translates VIM requests to a wire protocol. The “bottom” half of the VIM resides in a different process (that may or may not be co-located with the top half of the HTTP VIM), receives requests via the wire protocol, and makes requests of another VIM via the VIM API. |
| Marshall      | The act of bundling information into a package that can be transmitted                                                                                                                                                                                                                                                                                                                             |
| Stackable VIM | In whole, a VIM that implements a VIM API on the “top” to facilitate plugging into a VIM Manager and a VIM “consuming” API on the bottom                                                                                                                                                                                                                                                           |

**Table 19 – Terms - HTTP VIM Design**

| Term                    | Definition                                                                                                                                                                  |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Unmarshall              | Restoring information to a native format that was previously bundled for transmission                                                                                       |
| Vendor Interface Module | The unit or logical software component that represents a vendor-specific implementation of a system to the vendor agnostic model of the XAM interface.                      |
| VIM                     | Acronym for Vendor Interface Module                                                                                                                                         |
| Wire Protocol           | A protocol and model for making “transactions” across physical and logical barriers, such as separate deployment machines or separate processes within an operating system. |
| XAM                     | Acronym for eXtensible Access Method                                                                                                                                        |
| XAM SDK                 | The XAM Software Developer's Kit                                                                                                                                            |

**Scope**

The following items are in scope for this document (items not listed here are, by default, out of scope):

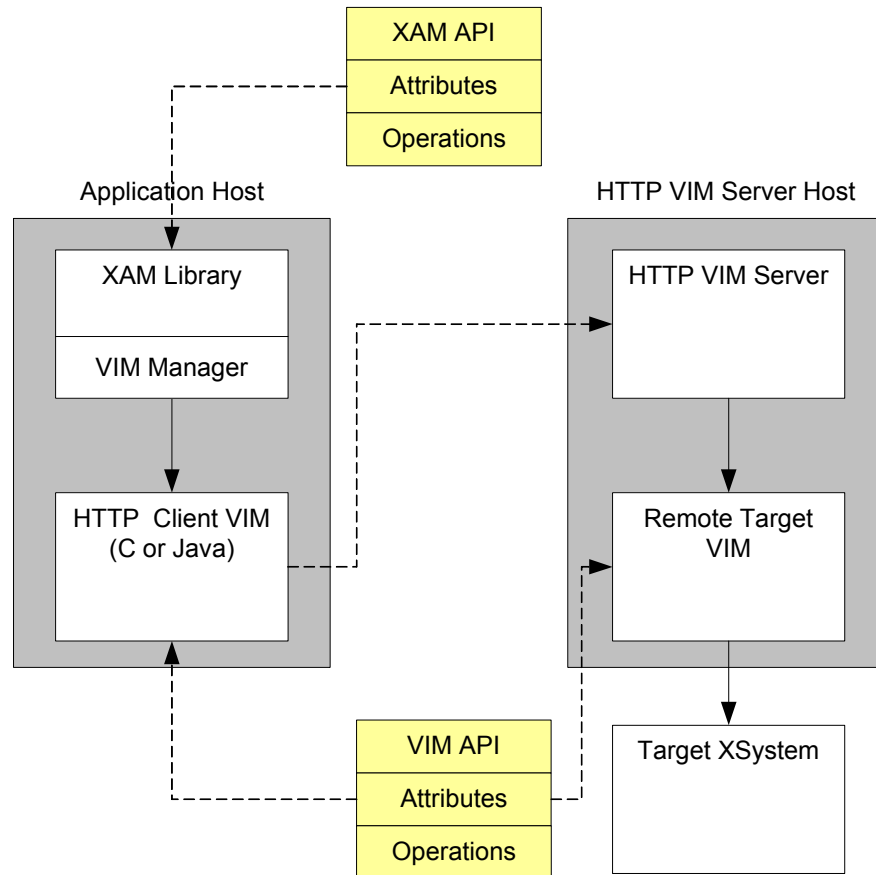
- Design decisions for implementing the bridge between the VIM Application Programming Interface that plugs into the C- and Java-based XAM Library and compatible XAM VIM that implements the Java VIM API and all discovery and deployment decisions for the stack
- Design of the HTTP VIM, including the following components:
  - Wire protocol to communicate between HTTP VIM Client and the HTTP VIM Server
  - The design of the HTTP VIM Client that plugs into the C-based XAM Library
  - The design of the HTTP VIM Client that plugs into the Java-based XAM Library
  - The design of the HTTP VIM Server that hosts VIMs that implement the Java-based VIM API
- Build for the entire stack from the HTTP VIM Server
- Deployment environment for the HTTP VIM Server
- Known issues in the design

The VIM Application Programming Interface (C and Java) is out of scope for this document. This programming interface is used to communicate “upward” in the XAM Library stack and is dictated by the [XAM-ARCH].

## Overview of HTTP VIM Design

The HTTP VIM provides remote access to a hosted Java VIM (such as the XAM SDK Reference VIM) via the HTTP Protocol. The purpose for this mechanism allows remote access to VIMs which may not support such a feature (Reference VIM), or access to Java only VIM from the C XAM Library.

The HTTP VIM separates most of the logic and implementation from the XAM Library process. Further, by the nature of its implementation, a separation of deployment systems can easily be achieved. All of the high-level software components to be delivered are shown in Figure 13, “High-Level Components and Deployment”.



**Figure 13 – High-Level Components and Deployment**

**Note:** Not shown in the diagram is an actual XAM Storage System. The Target VIM resides in the same process on the same host as the HTTP Protocol VIM Server, but the XAM Storage System which operates with the Target VIM is potentially on another host.

From the illustration above, the following Component, Responsibilities, and Collaborations (CRC) are derived (along with additional notes about design issues, documentation and owners).

**Table 20 – CRC for XAM\_API**

| Interface        | XAM_API                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Responsibilities | Represent XAM functionality to vendor applications, including <ul style="list-style-type: none"> <li>- Defines interfaces for locating individual systems</li> <li>- Defines interfaces for interacting with individual systems</li> <li>- Defines interfaces for interacting with sets of objects</li> <li>- Defines interfaces on individual objects</li> <li>- Defines security interfaces</li> <li>- Defines query interfaces</li> </ul> |
| Documentation    | <ul style="list-style-type: none"> <li>- C API – See xam*.h from src/xam/include directory in the xam_sdk package available in Subversion</li> <li>- Java API – See [XAM-Java-API]</li> </ul>                                                                                                                                                                                                                                                |

**Table 21 – CRC for VIM\_API**

| Interface        | VIM_API                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Responsibilities | Represent VIM functionality to the VIM Manager and up to the XAM Library itself, including <ul style="list-style-type: none"> <li>- Defines interfaces for locating individual systems</li> <li>- Defines interfaces for interacting with individual systems</li> <li>- Defines interfaces for interacting with sets of objects</li> <li>- Defines interfaces on individual objects</li> <li>- Defines security interfaces</li> <li>- Defines query interfaces</li> </ul> |
| Documentation    | <ul style="list-style-type: none"> <li>- C API – See vim*.h from src/xam/vim directory in the xam_sdk package available in Subversion</li> <li>- Java API – Generate JavaDoc from Interface code</li> </ul>                                                                                                                                                                                                                                                               |

**Table 22 – CRC for XAM\_Library**

| Component        | XAM_Library                                                                                                                                                                                                                                                                                                                                        |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Responsibilities | Receive API requests and process accordingly <p>Several of the XAM Library and API calls are “organizational” in nature and do not pass through to the underlying VIMs and VIM Objects. These calls include functions to:</p> <ul style="list-style-type: none"> <li>- Discover individual systems</li> <li>- Load individual libraries</li> </ul> |



**Table 22 – CRC for XAM\_Library**

| Component      | XAM_Library                   |
|----------------|-------------------------------|
| Collaborations | VIM Manager – Local API call  |
| Documentation  | - [XAM-ARCH]<br>- [XAM-C-API] |

**Table 23 – CRC for HTTP\_VIM\_Client**

| Component        | HTTP_VIM_Client                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Responsibilities | <ul style="list-style-type: none"> <li>- Implement the VIM_API</li> <li>- Implement a mechanism to input the target HTTP_VIM_Server (network location including host name or IP address and port)</li> <li>- Receive VIM API requests (on the various VIM API Objects, including XSystems, XSets, and XStreams)</li> <li>- Marshall VIM API parameters and method calls into the neutral wire format</li> <li>- Do any mappings necessary between the VIM API and the neutral wire format</li> <li>- Pass VIM API request in neutral “wire” format to the target HTTP_VIM_Requestor and receive response in neutral “wire” format</li> <li>- Unmarshall / Map the response from the HTTP_VIM_Requestor back to the C VIM API</li> <li>- Return VIM API request to requestor (asynchronous and/or synchronous responses must be implemented)</li> </ul> |
| Collaborations   | HTTP_VIM_Server – HTTP(S) – Asynchronous and Synchronous – neutral wire format, one HTTP VIM Requestor may use one HTTP VIM Server. One HTTP VIM Server may be used by many HTTP VIM Requestors.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Documentation    | VIM C API available from xam_sdk                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

**Table 24 – CRC for HTTP\_VIM\_Server**

| Component        | HTTP_VIM_Server                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Responsibilities | <ul style="list-style-type: none"> <li>- Receive requests in the defined neutral wire format</li> <li>- Unmarshall requests</li> <li>- Map any requests that require mapping to the Java VIM Interface</li> <li>- Call the appropriate VIM or [XAM-ARCH] object (XSystem, XSet, etc.) with the appropriate parameters</li> <li>- Keep track of active VIM API objects and maintain handle to object relationships that the HTTP VIM Client can use to associate with the objects that it is manipulating</li> <li>- Receive responses from a VIM or VIM API Object</li> <li>- Do any required mapping between the VIM response and the neutral wire format</li> <li>- Marshall the parameters and method to the neutral wire format</li> <li>- Return a response to the requestor</li> </ul> |
| Collaborations   | Java VIM Implementation via the Java VIM Interface (such as the XAM SDK Reference VIM)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| Documentation    | Java VIM Interface                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

### *Java HTTP VIM Client*

The supplied Java HTTP Client VIM provide implementations for each of the required XAM™ Objects:

- HTTPFieldContainer.java
- HTTPXAsync.java
- HTTPAsyncCallbackManager.java
- HTTPIterator.java
- HTTPXSet.java
- HTTPXStream.java
- HTTPXSystem.java

Each of these objects directly implements the similarly named XAM Interface and present the expected XAM methods to the application. Each of the XAM objects retains a reference to the handle created in the HTTP VIM Server as well as host and port information used to form the appropriate HTTP Request.

Each HTTP XAM Object utilizes the utility class HTTPRequest, which mainly marshals arguments to HTTP request form. The HTTPRequest methods also unmarshal responses to produce the correct return values for presentation to the application. The HTTP XAM objects are “thin client,” in that no caching is performed and little state is retained locally. Each XAM method translates directly to a HTTP request to the HTTP VIM Server.

A sample implementation of the the Java Client VIM looks like the following for the method XSet.getActualRetentionEnabled():

```
public boolean getActualRetentionEnabled(String retentionID)
    throws AuthenticationExpiredException,
    InvalidArgumentException,
    PolicyMismatchException, ObjectInUseException,
    XSetAbandonException,
    XSetCorruptException, XAMException
{
    HashMap<String, String> responses;
    try
    {
        responses = HTTPRequest.request(m_host,

ProtocolConstants.XSET_GET_ACTUAL_RETENTION_ENABLED_HANDLER,
        ProtocolConstants.ARGUMENT_RETENTION_ID,
        URLEncoder.encode(retentionID,
ProtocolConstants.UTF8),
        ProtocolConstants.XSET_HANDLE, m_handle);
        boolean enables = Boolean.parseBoolean(responses.get
        (ProtocolConstants.ARGUMENT_ENABLED));
        responses.clear();
        return enables;
    } catch (UnsupportedEncodingException e )
    {
        throw new XAMException( ProtocolConstants.TRANSPORT_ENCODE_ERROR,
e );
    }
}
```

The actual HTTP request is performed in the method HTTPRequest.request(). Input arguments are passed to the request() method, to be marshaled in the URL connection. The result page is processed and passed back to the caller as a property map. This method extracts the single argument needed, ARGUMENT\_ENABLED, converts its string value to the boolean required, and returns to the application. The method HTTPRequest.request() also processes the “status” and “statusMsg” values. If any value other than SUCCESS (zero) is returned, these two items are used to construct an appropriate XAMException, or subclass, which is thrown to the application.

### Connect Processing

Because the XSystemConnect method expects a buffer of initialization properties to be sent, the HTTPXSystem object must buffer up all properties from the local XAM Library and send them to the remote VIM for the XSystemConnect method. To do this, HTTPXSystem instantiates a local FieldContainer, class TempFieldContainer. This object is deleted once the connect has completed.

### XAsyncCallback Management

Because the HTTP VIM Server does not, and cannot, handle callback methods (for XAsyncListeners), the Java HTTP Client VIM does so locally. Each HTTPXSystem instantiates a local copy of a HTTPXAsyncCallbackManager manager. Each manager is passed a

reference to a newly created HTTPXAsync object. Periodically, the callback manager executes the POLL method to the HTTP VIM Server.

When an async operation has been indicated that it is complete (a result comes back from the HTTP VIM Server), the callback manager will do the following:

- 1 If the XAsync is the result of an XStreamAsyncRead operation, perform a GetData method to retrieve the bytes. Data is copied to the buffer that is supplied in the HTTPXStream.asyncRead method.
- 2 Mark the HTTPXAsync object as complete so that calls to the HTTPXAsync.isComplete() return the correct value. The Java HTTP VIM does not call the HTTP VIM Server isComplete() method because of a race condition between the application, the callback manager, and the HTTP VIM Server.
- 3 If an XAsyncListener object has been provided for the XAsync operation, call the listener.

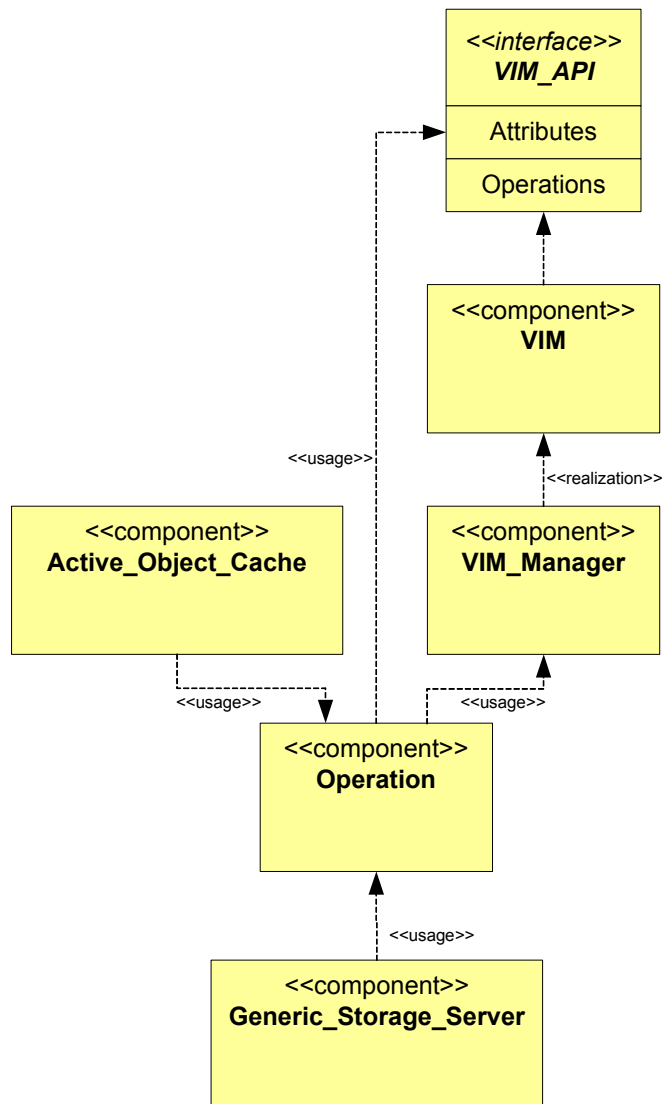
Closing the HTTPXAsync object removes it from the callback manager's watch list.

### *Java HTTP VIM Server*

The HTTP VIM Server uses Java as the implementation language to create an end-to-end implementation stack that combines with the Java-based XAM VIM implementations. The Java-based VIM implementations should also be usable directly from the Java-based XAM Library.

The generic responsibilities of the Java HTTP VIM Server are documented in "Functionality". The components described here interact to deliver the HTTP VIM Server functionality. The actual implementation of the components is documented within the implementation code for the Java HTTP VIM Server.

A component diagram is shown in Figure 14, “HTTP VIM Server Overview”



**Figure 14 – HTTP VIM Server Overview**

An HTTP VIM Requestor delivers a message to the **Generic\_Storage\_Server**. The interfaces for the Storage Server are, theoretically, generic in nature, but the implementation itself is bound to the HTTP protocol. As a result, the message is delivered to the **Generic\_Storage\_Server** via a socket connection.

The **Generic\_Storage\_Server** in the Java HTTP VIM Server implementation package uses the Jetty WebServer. The various wire operations subclass (often multiple generations) the **AbstractHttpHandler** from the Jetty WebServer. These subclasses are generically labeled “**Operation**” in Figure 14, “HTTP VIM Server Overview”. The Jetty WebServer calls the proper **Operation** depending on the contents of the HTTP Header sent from the HTTP VIM Requestor.

Other than the **XSystemConnect** call itself, each **Operation** includes a “handle” that identifies an active object in the **Active\_Object\_Cache**. The

active object may be a top-level object (like an XSystem, XSet or XStream) or a lower-level object that can be independently manipulated, like an XIterator. The active objects should only be associated with a single HTTP VIM Requestor. In practice, multiple HTTP VIM Requestors may be using active objects from a single HTTP VIM Server, so the active object handles should not be available between requestors.

The VIM Manager locates the proper class for XSystem top-level objects, identified by XRI. The VIM Manager loads classes at bootstrap time into the virtual machine and then instantiates the VIM classes as objects as needed or at bootstrap time. Each XMLIB.connect Operation will go to the VIM Manager to instantiate a VIM object instance that can be used to retrieve an XSystem. Once the Operation retrieves the XSystem, it is stored in the Active\_Object\_Cache, and a handle gets returned to the requestor.

### *VIM Class and Library Load Operations*

The design of the HTTP Protocol VIM includes two locations where VIM Class and Library loading come into play.

- The first load operation occurs when the XAM Library loads the C-based HTTP VIM Requestor. This operation must adhere to the XAM Library process for locating classes and identifying the XRIs associated with that class.
- The second load operation occurs within the HTTP VIM Server, as it contains its own “version” of a VIM Manager. This VIM Manager and the VIM Manager in the Java-based XAM Library could rely on the same file formats or process for identifying loadable classes and the XRIs to which those classes attach when such a format is available.

### *VIM Wire Protocol*

While not a first-class software component, the wire protocol used between the “halves” of the HTTP VIM is the most important component in the architecture, design, and implementation of the HTTP VIM. The wire protocol must be open and flexible for use in languages and collaborations that this design specification does not address. Uses of the wire protocol include building VIMs with language bindings other than Java and C or producing other variants of “split” VIMs that cross process, network, and/or language boundaries.

It is assumed that the wire protocol will be passed between components via the preferred route of HTTPS or the unsecure path of HTTP. HTTP has a variety of advantages over HTTPS during development (i.e., traceability, debugging, performance, etc.) but will be widely shunned by customers in production environments for security reasons.

As HTTPS and HTTP are the transmission protocol, the overall wire protocol will make heavy use of the HTTP standard for the headers and bodies of messages, as well as the encoding and mime types that constitute an HTTP message. The wire protocol will also inherit the attributes of HTTP/HTTPS, including the attributes of the REST architecture, such as the statelessness inherent in the architecture, transparency, and the ability to insert proxies and routers, and more. For a complete dissertation on the REST architecture, see [REST]. Note also that the HTTP/HTTPS protocols do not exhibit all of the characteristics of the REST architecture.

## Organization

The following sections document the various HTTP Headers and bodies that are exchanged over the course of an operation. Details about behavior and state are also listed. Please refer to [XAM-ARCH] for a complete discussion of the life cycle of the objects on which these operations occur and a complete description of the semantics and behavior for the operations.

The sections are labelled with the operation names in the [XAM-ARCH]. The names are translated to more accessible wire format names within the sections.

## Method Access

Each method is accessed by presenting a properly formed HTTP GET or HTTP Post request to the HTTP VIM Server. For the most part, each method is represented by a simple URL. Arguments may be passed via the standard URL argument method or as HTTP Header values.

## Return Values

All return values from the method are returned as name/value pairs in the HTTP response page. The names are defined in the following sections for each method. Values are generally printable value, except for reading byte buffers from XStreams. In addition to method-specific return values, each method returns “status” and “statusMsg”. The “status” value is the XAM status code (as defined by the XAM SDK). The HTTP VIM Server also provides the “statusMsg”, which is a descriptive string corresponding to the status code. The “statusMsg” value is an obtained exception message() of the VIM method. If the status value is success (zero), statusMsg is generally not included.

## Value Encoding

All string values (xam\_string, xam\_xuid) are encoded to be URL safe [URL-Encoding]. All XUID values are transmitted using the base64 representation (also URL safe encoded). Numeric values are transmitted in their printable formats. Boolean values are transmitted using the values “true” and “false”. Datetime values are transmitted using millisecond values of the stdclib format, in a printable numeric format.

## Example Exchange

This example shows the method “XSet.abandon()”.

```
Method invocation:
GET /XSetAbandon HTTP/1.1
handle: 2000202
User-Agent: Jakarta Commons-HttpClient/3.0.1
Host: 192.168.1.100:9925
```

A complete URL to perform the same thing looks like:

```
http://192.168.1.100:9925/XSetAbandon?handle=20000202
```

A complete response may look like the following:

```
HTTP/1.1 200 OK
Date: Wed, 22 Aug 2007 12:45:13 GMT
Server: Jetty/4.2.20 (Windows XP/5.1 x86 java/
1.6.0_02)
Transfer-Encoding: chunked
status=0
```

Example exchanges are not included for the following methods but generally follow the example which has just been presented.

## Operations

**XAMCreateFieldIterator** This operation abandons an XSet and all of the resources associated with it. The XSet handle implies an association with an XSystem. The handle will no longer identify a valid active XSet object instance after the abandon operation completes.

| HTTP Attribute | Type                    | Example                            | Description                                                                   |
|----------------|-------------------------|------------------------------------|-------------------------------------------------------------------------------|
| Method         | /XAMCreateFieldIterator | GET /XCreateFieldIterator HTTP/1.1 | Identifies operation and scope                                                |
| class          | xam_string              | class=xset                         | Identifies to what class of XAM object this method is to be applied           |
| handle         | xam_handle              | handle=2000202                     | Identifies the XAM object instance on which the operation should be performed |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

**XAsyncClose** This operation closes the specified XAsync XAM operation.

| HTTP Attribute | Type         | Example                   | Description                    |
|----------------|--------------|---------------------------|--------------------------------|
| Method         | /XAsyncClose | GET /XAsyncClose HTTP/1.1 | Identifies operation and scope |
| handle         | xam_handle   | handle=2000202            | The XAsync handle              |



## Response

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

XAsyncGetBytesRead This operation returns the number of bytes read by the XAsync operation.

| HTTP Attribute | Type                | Example                          | Description                    |
|----------------|---------------------|----------------------------------|--------------------------------|
| Method         | /XAsyncGetBytesRead | GET /XAsyncGetBytesRead HTTP/1.1 | Identifies operation and scope |
| handle         | xam_handle          | handle=2000202                   | The XAsync handle              |

## Response:

| Name             | Type        | Example                    | Description                                                                                                             |
|------------------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status           | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg        | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| async_bytes read | xam_integer | async_bytes_read=1024      | The number of bytes read by the XAsync operation                                                                        |

XAsyncGetBytesWritten This method returns the bytes written by the XAsync operation. The body will contain the following information in name/value pairs.

| HTTP Attribute | Type                   | Example                             | Description                    |
|----------------|------------------------|-------------------------------------|--------------------------------|
| Method         | /XAsyncGetBytesWritten | GET /XAsyncGetBytesWritten HTTP/1.1 | Identifies operation and scope |
| handle         | xam_handle             | handle=2000202                      | The XAsync handle              |

Response: XAsyncGetData

| Name                | Type        | Example                    | Description                                                                                                             |
|---------------------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status              | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg           | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| async_bytes_written | xam_integer | async_byteswritten=728     | The number of bytes written by the XAsync operation.                                                                    |

This method returns the data obtained from the most recent XStreamAsyncRead operation. The body contains the following information in name/value pairs.

| HTTP Attribute | Type         | Example                   | Description                                                                |
|----------------|--------------|---------------------------|----------------------------------------------------------------------------|
| Method         | /XStreamRead | GET /XStreamRead HTTP/1.1 | Identifies operation and scope                                             |
| handle         | xam_handle   | handle=33333333           | Identifies the XStream instance on which the operation should be performed |

The response to the request will contain the contents of the XStream data in the body. If no body is included, an error occurred in the response. Clients should be prepared to read as much data as was originally requested.

XAsyncGetStatus

This method gets the status of the XAsync operation. The body will contain the following information in name/value pairs.

| HTTP Attribute | Type             | Example                       | Description                    |
|----------------|------------------|-------------------------------|--------------------------------|
| Method         | /XAsyncGetStatus | GET /XAsyncGetStatus HTTP/1.1 | Identifies operation and scope |
| handle         | xam_handle       | handle=2000202                | The XAsync handle              |

Response:

| Name             | Type        | Example                                      | Description                                                                                                             |
|------------------|-------------|----------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status           | xam_integer | status=0                                     | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg        | xam_string  | statusMsg=Object is closed                   | A string message associated with the status code. Not returned if status=0.                                             |
| async_status     | xam_integer | status=0                                     | The XAM operation status of the XAsync operation.                                                                       |
| async_status Msg | xam_string  | async_statusMsg=XSystem is not authenticated | A string message associated with the async_status code. Not returned if async_status=0.                                 |

#### XAsyncGetXOPID

This method returns the XOPID of the XAsync operation. The body will contain the following information in name/value pairs.

| HTTP Attribute | Type            | Example                      | Description                    |
|----------------|-----------------|------------------------------|--------------------------------|
| Method         | /XAsyncGetXOPID | GET /XAsyncGetXOPID HTTP/1.1 | Identifies operation and scope |
| handle         | xam_handle      | handle=2000202               | The XAsync handle              |

Response:

| Name        | Type        | Example                    | Description                                                                                                             |
|-------------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status      | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg   | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| async_xopid | xam_integer | async_xopid=67890          | The XOPID associated with the XAsync operation.                                                                         |

#### XAsyncGetXSet

This method returns the XSet handle associated the XAsync operation. The body will contain the following information in name/value pairs.

| HTTP Attribute | Type           | Example                     | Description                    |
|----------------|----------------|-----------------------------|--------------------------------|
| Method         | /XAsyncGetXSet | GET /XAsyncGetXSet HTTP/1.1 | Identifies operation and scope |
| handle         | xam_handle     | handle=2000202              | The XAsync handle              |

Response:

| Name       | Type        | Example                    | Description                                                                                                             |
|------------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status     | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg  | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| async_xset | xam_handle  | async_xset=2000202         | The handle of the XSet associated with the XAsync operation.                                                            |

XAsyncGetXStream

This method returns the XStream associated with the XAsync operation. The body will contain the following information in name/value pairs.

| HTTP Attribute | Type              | Example                        | Description                    |
|----------------|-------------------|--------------------------------|--------------------------------|
| Method         | /XAsyncGetXStream | GET /XAsyncGetXStream HTTP/1.1 | Identifies operation and scope |
| handle         | xam_handle        | handle=2000202                 | The handle of XAsync           |

Response:

| Name          | Type        | Example                    | Description                                                                                                             |
|---------------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status        | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg     | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| async_xstream | xam_handle  | async_xstream=2000202      | The XStream handle associated with the XAsync operation.                                                                |

XAsyncGetXUID

This method returns the XUID associated with the XAsync operation. The body will contain the following information in name/value pairs.

| HTTP Attribute | Type           | Example                     | Description                    |
|----------------|----------------|-----------------------------|--------------------------------|
| Method         | /XAsyncGetXUID | GET /XAsyncGetXUID HTTP/1.1 | Identifies operation and scope |
| handle         | xam_handle     | handle=2000202              | The handle of XAsync           |

Response:

| Name       | Type        | Example                    | Description                                                                                                             |
|------------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status     | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg  | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| async_xuid | xam_string  | async_xstream=AAA...       | The XUID handle associated with the XAsync operation.                                                                   |

**XAsyncHalt** This method halts the XAsync operation. The body will contain the following information in name/value pairs.

| HTTP Attribute | Type        | Example                  | Description                    |
|----------------|-------------|--------------------------|--------------------------------|
| Method         | /XAsyncHalt | GET /XAsyncHalt HTTP/1.1 | Identifies operation and scope |
| handle         | xam_handle  | handle=2000202           | The handle of XAsync           |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

**XAsyncIsComplete** This method returns the complete status associated with the XAsync operation. The body will contain the following information in name/value pairs.

Because of a race condition between application, client vim, and server vim threads, all Client HTTP VIMs are encouraged to use the poll mechanism to determine if the XAsync is complete. During the processing of the complete status, the client VIM should transfer data read. This problem is only likely to occur when the operation was an XStreamAsyncRead.

| HTTP Attribute | Type              | Example                        | Description                    |
|----------------|-------------------|--------------------------------|--------------------------------|
| Method         | /XAsyncIsComplete | GET /XAsyncIsComplete HTTP/1.1 | Identifies operation and scope |
| handle         | xam_handle        | handle=2000202                 | The handle of XAsync           |

Response:

| Name             | Type        | Example                    | Description                                                                                                             |
|------------------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status           | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg        | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| async_iscomplete | xam_boolean | async_iscomplete=true      | The Boolean value of the XAsync operation isComplete().                                                                 |

#### XAsync – POLL

The POLL method is available so that clients can determine, in bulk, when XAsync operations are completed. This method specifies an XSystem handle. All completed XAsync handles are returned. It is the responsibility of the client-side VIM to manage and call the callback/listener methods specified by the application.

If the XAsync operation was an XStreamAsyncRead, the data should be retrieved and copied to the application's buffer before setting the XAsync complete status.

| HTTP Attribute | Type       | Example             | Description                                                                |
|----------------|------------|---------------------|----------------------------------------------------------------------------|
| Method         | /POLL      | GET /XPOLL HTTP/1.1 | Identifies operation and scope                                             |
| handle         | xam_handle | handle=33333333     | Identifies the XSystem instance on which the operation should be performed |

The response to the request will contain the contents of the XStream in the body. If no body is included, an error occurred in the response.

| Name            | Type        | Example                    | Description                                                                                                             |
|-----------------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status          | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg       | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| async_complete# | xam_handle  | async_complete0=1248       | XAsync object with handle 1248 is complete.                                                                             |

A complete request may look like the following:

```
POST /XStreamWrite HTTP/1.1
handle: 33333333
User-Agent: Jakarta Commons-HttpClient/3.0.1
Host: 192.168.1.100:9925
```

A complete response may look like the following:

```
HTTP/1.1 200 OK
Date: Wed, 22 Aug 2007 12:45:13 GMT
Server: Jetty/4.2.20 (Windows XP/5.1 x86 java/
1.6.0_02)
Transfer-Encoding: chunked
status=0
async_complete0=1248
async_complete1=467230
async_complete2=72438
```

**XIteratorClose** This method closes the XIterator object. The body will contain the following information in name/value pairs.

| HTTP Attribute | Type            | Example                      | Description                         |
|----------------|-----------------|------------------------------|-------------------------------------|
| Method         | /XIteratorClose | GET /XIteratorClose HTTP/1.1 | Identifies operation and scope      |
| handle         | xam_handle      | handle=2000202               | The handle of the XIterator object. |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

**XIteratorHasNext** This method returns the hasNext() value of the XIterator. The body will contain the following information in name/value pairs.

| HTTP Attribute | Type              | Example                     | Description                    |
|----------------|-------------------|-----------------------------|--------------------------------|
| Method         | /XIteratorHasNext | GET /XIteratorNext HTTP/1.1 | Identifies operation and scope |
| handle         | xam_handle        | handle=2000202              | The handle of XIterator.       |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| hasnext   | xam_boolean | hasnext=true               | The Boolean value of the XIterator.hasNext() method.                                                                    |

**XIteratorNext** This method returns the next field name from the XIterator. The body will contain the following information in name/value pairs.

| HTTP Attribute | Type           | Example                      | Description                    |
|----------------|----------------|------------------------------|--------------------------------|
| Method         | /XIteratorNext | GET /XIteratorNext HTTP/ 1.1 | Identifies operation and scope |
| handle         | xam_handle     | handle=2000202               | The handle of XIterator.       |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| next      | xam_string  | next=com.example.name      | The next field name returned from the XIterator.next() method.                                                          |

**XSetAbandon** This operation abandons an XSet and all of the resources associated with it. The XSet handle implies an association with an XSystem. The handle will no longer identify a valid active XSet object instance after the abandon operation completes.

| HTTP Attribute | Type         | Example                    | Description                                                             |
|----------------|--------------|----------------------------|-------------------------------------------------------------------------|
| Method         | /XSetAbandon | GET /XSetAbandon HTTP/ 1.1 | Identifies operation and scope                                          |
| handle         | xam_handle   | handle=2000202             | Identifies the XSet instance on which the operation should be performed |



Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

XSetApplyAccessPolicy

This method applies the access policy on the specified XSet. The body will contain the following information in name/value pairs.

| HTTP Attribute | Type                   | Example                              | Description                              |
|----------------|------------------------|--------------------------------------|------------------------------------------|
| Method         | /XSetApplyAccessPolicy | GET / XSetApplyAccessPolicy HTTP/1.1 | Identifies operation and scope           |
| handle         | xam_handle             | handle=2000202                       | The handle of the XSet                   |
| binding        | xam_boolean            | binding=true                         | The binding setting for the operation    |
| policy         | xam_string             | policy:access_name                   | The access policy name for the operation |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

XSetApplyAutoDeletePolicy

This method applies the auto delete on the specified XSet. The body will contain the following information in name/value pairs.

| HTTP Attribute | Type                       | Example                                  | Description                              |
|----------------|----------------------------|------------------------------------------|------------------------------------------|
| Method         | /XSetApplyAutoDeletePolicy | GET / XSetApplyAutoDeletePolicy HTTP/1.1 | Identifies operation and scope           |
| handle         | xam_handle                 | handle=2000202                           | The handle of the XSet                   |
| binding        | xam_boolean                | binding=true                             | The binding setting for the operation    |
| policy         | xam_string                 | policy:access_name                       | The access policy name for the operation |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

XSetApplyBaseRetention

This method applies the base retention policy on the specified XSet. The body will contain the following information in name/value pairs.

| HTTP Attribute | Type                    | Example                               | Description                              |
|----------------|-------------------------|---------------------------------------|------------------------------------------|
| Method         | /XSetApplyBaseRetention | GET / XSetApplyBaseRetention HTTP/1.1 | Identifies operation and scope           |
| handle         | xam_handle              | handle=2000202                        | The handle of the XSet                   |
| binding        | xam_boolean             | binding=true                          | The binding setting for the operation    |
| policy         | xam_string              | policy:access_name                    | The access policy name for the operation |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

XSetApplyManagement  
Policy

This method applies the management policy on the specified XSet. The body will contain the following information in name/value pairs.

| HTTP Attribute | Type                   | Example                              | Description                              |
|----------------|------------------------|--------------------------------------|------------------------------------------|
| Method         | /XAsyncGetBytesWritten | GET / XAsyncGetBytesWritten HTTP/1.1 | Identifies operation and scope           |
| handle         | xam_handle             | handle=2000202                       | The handle of the XSet                   |
| binding        | xam_boolean            | binding=true                         | The binding setting for the operation    |
| policy         | xam_string             | policy:access_name                   | The access policy name for the operation |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

XSetApplyRetentionDuration  
Policy

This method applies the retention duration policy on the specified XSet. The body will contain the following information in name/value pairs.

| HTTP Attribute | Type                              | Example                                         | Description                              |
|----------------|-----------------------------------|-------------------------------------------------|------------------------------------------|
| Method         | /XSetApplyRetentionDurationPolicy | GET / XSetApplyRetentionDurationPolicy HTTP/1.1 | Identifies operation and scope           |
| handle         | xam_handle                        | handle=2000202                                  | The handle of the XSet                   |
| binding        | xam_boolean                       | binding=true                                    | The binding setting for the operation    |
| policy         | xam_string                        | policy:access_name                              | The access policy name for the operation |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

XSetApplyRetentionEnabled  
Policy

This method applies the retention enabled policy on the specified XSet. The body will contain the following information in name/value pairs.

| HTTP Attribute | Type                             | Example                                        | Description                              |
|----------------|----------------------------------|------------------------------------------------|------------------------------------------|
| Method         | /XSetApplyRetentionEnabledPolicy | GET / XSetApplyRetentionEnabledPolicy HTTP/1.1 | Identifies operation and scope           |
| handle         | xam_handle                       | handle=2000202                                 | The handle of the XSet                   |
| binding        | xam_boolean                      | binding=true                                   | The binding setting for the operation    |
| policy         | xam_string                       | policy:access_name                             | The access policy name for the operation |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

#### XSetApplyShredPolicy

This method applies the shred policy on the specified XSet. The body will contain the following information in name/value pairs.

| HTTP Attribute | Type                  | Example                            | Description                              |
|----------------|-----------------------|------------------------------------|------------------------------------------|
| Method         | /XSetApplyShredPolicy | GET /XSetApplyShredPolicy HTTP/1.1 | Identifies operation and scope           |
| handle         | xam_handle            | handle=2000202                     | The handle of the XSet                   |
| binding        | xam_boolean           | binding=true                       | The binding setting for the operation    |
| policy         | xam_string            | policy:access_name                 | The access policy name for the operation |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

#### XSetApplyStoragePolicy

This method applies the storage policy on the specified XSet. The body will contain the following information in name/value pairs.

| HTTP Attribute | Type                    | Example                              | Description                              |
|----------------|-------------------------|--------------------------------------|------------------------------------------|
| Method         | /XSetApplyStoragePolicy | GET /XSetApplyStoragePolicy HTTP/1.1 | Identifies operation and scope           |
| handle         | xam_handle              | handle=2000202                       | The handle of the XSet                   |
| binding        | xam_boolean             | binding=true                         | The binding setting for the operation    |
| policy         | xam_string              | policy:access_name                   | The access policy name for the operation |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

**XSetAsyncCommit** This creates an asynchronous XSet commit operation. The body will contain the following information in name/value pairs.

| HTTP Attribute | Type        | Example                  | Description                                    |
|----------------|-------------|--------------------------|------------------------------------------------|
| Method         | /XSetCommit | GET /XSetCommit HTTP/1.1 | Identifies operation and scope                 |
| handle         | xam_handle  | handle=2000202           | The handle of the XSet                         |
| xopid          | xam_integer | xopid:678890             | The xopid to be associated with this operation |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| handle    | xam_handle  | handle=2000202             | The handle of the XAsync object                                                                                         |

**XSetAsyncOpenXStream** This creates an asynchronous XSet open XStream operation. The body will contain the following information in name/value pairs. At some point later, the XStream will be opened and available for retrieval.

| HTTP Attribute | Type                  | Example                            | Description                                    |
|----------------|-----------------------|------------------------------------|------------------------------------------------|
| Method         | /XSetAsyncOpenXStream | GET /XSetAsyncOpenXStream HTTP/1.1 | Identifies operation and scope                 |
| handle         | xam_handle            | handle=2000202                     | The handle of the XSet                         |
| xopid          | xam_integer           | xopid:678890                       | The xopid to be associated with this operation |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| handle    | xam_handle  | handle=2000202             | The handle of the XAsync object.                                                                                        |

**XSetClose** This operation closes an XSet and all of the resources associated with it. The XSet handle implies an association with an XSystem. Various errors may occur during the close operation, often implying that the XSet has pending operations on it. In the case of a successful operation, the handle to the XSet is no longer valid once the close operation completes. In the case of some of the failure status codes (identified below), the handle will remain valid.

| HTTP Attribute | Type       | Example                 | Description                                                             |
|----------------|------------|-------------------------|-------------------------------------------------------------------------|
| Method         | /XSetClose | GET /XSetClose HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle | handle=2000202          | Identifies the XSet instance on which the operation should be performed |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

**XSetCommit** The commit operation commits the changes to an XSet to the storage server. After the operation, the handle to the XSet will no longer be valid, assuming the returned status indicates success. Some return values may leave a valid handle intact; see the values below.

| HTTP Attribute | Type        | Example                  | Description                                                             |
|----------------|-------------|--------------------------|-------------------------------------------------------------------------|
| Method         | /XSetCommit | GET /XSetCommit HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle  | handle=2000202           | Identifies the XSet instance on which the operation should be performed |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| xuid      | xam_xuid    | xuid=AAA....               | The xuid returned from the XSet.commit() method.                                                                        |

XSetContainsField The XSetContainsField will determine if the XSet contains the named field.

| HTTP Attribute | Type               | Example                         | Description                                                                  |
|----------------|--------------------|---------------------------------|------------------------------------------------------------------------------|
| Method         | /XSetContainsField | GET /XSetContainsField HTTP/1.1 | Identifies operation and scope                                               |
| handle         | xam_handle         | handle=2000202                  | Identifies the XSet instance on which the operation should be performed      |
| name           | xam_string         | name=com.example.field          | The name of the field to be checked for inclusion. Must be URL safe encoded. |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| value     | xam_boolean | value=false                | The Boolean value of the containsField() method.                                                                        |

XSetCreateProperty The XSetCreateProperty operation does not have a direct parallel in the [XAM-ARCH]. Instead, this is an “aggregate” operation that is overloaded to simplify the creation of the handlers on each side of the wire. All possible property creation operations are collapsed to this single call. As all information on the wire can be treated as string information, the value on the wire must be convertible from a string to the target property type. The CreateProperty

operation will return parameter errors in the status if the value is not readily convertible using the Java type conversion rules.

| HTTP Attribute | Type                                                                                                        | Example                          | Description                                                                                                                                  |
|----------------|-------------------------------------------------------------------------------------------------------------|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| Method         | /XSetCreateProperty                                                                                         | GET /XSetCreateProperty HTTP/1.1 | Identifies operation and scope                                                                                                               |
| handle         | xam_handle                                                                                                  | handle=2000202                   | Identifies the XSet instance on which the operation should be performed                                                                      |
| ptype          | xam_string<br>Possible values include:<br>- boolean<br>- int<br>- float<br>- xuid<br>- string<br>- datetime | ptype=boolean                    | The type of property to create. The type of property will dictate the format of the value property.                                          |
| name           | xam_string                                                                                                  | name=propertyname                | The name of the property to set                                                                                                              |
| binding        | xam_boolean                                                                                                 | binding=true                     | Whether this field should be binding                                                                                                         |
| value          | variable                                                                                                    | value=true                       | The value to which the property should be set. This value will always be a string, but the string must be convertible to the property ptype. |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

XSetCreateRetention This method will create the named retention on the XSet.

| HTTP Attribute | Type                 | Example                           | Description                                                                      |
|----------------|----------------------|-----------------------------------|----------------------------------------------------------------------------------|
| Method         | /XSetCreateRetention | GET /XSetCreateRetention HTTP/1.1 | Identifies operation and scope                                                   |
| handle         | xam_handle           | handle=2000202                    | Identifies the XSet instance on which the operation should be performed          |
| binding        | xam_boolean          | binding=true                      | The Boolean value of the binding setting.                                        |
| retentionid    | xam_string           | retentionid=test_retention        | The name of the retention to be created. The string must be encoded to URL safe. |



Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

**XSetCreateXStream** Creates an XStream within a particular XSet with the values that were passed to this function.

| HTTP Attribute | Type               | Example                         | Description                                                             |
|----------------|--------------------|---------------------------------|-------------------------------------------------------------------------|
| Method         | /XSetCreateXStream | GET /XSetCreateXStream HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle         | handle=2000202                  | Identifies the XSet instance on which the operation should be performed |
| name           | xam_string         | name: propertyname              | The name of the property to set                                         |
| binding        | xam_boolean        | binding: true                   | Whether this field should be binding                                    |
| type           | xam_string         | type: text/html                 | The MIME type for the new field.                                        |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| handle    | xam_integer | handle:2000202             | The handle of the newly created XStream                                                                                 |

**XSetDeleteField** This method deletes the named field from the XSet.

| HTTP Attribute | Type             | Example                       | Description                                                             |
|----------------|------------------|-------------------------------|-------------------------------------------------------------------------|
| Method         | /XSetDeleteField | GET /XSetDeleteField HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle       | handle=2000202                | Identifies the XSet instance on which the operation should be performed |
| name           | xam_string       | name=com.example.field        | The name of the field to be deleted                                     |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

XSetGetActualAutoDelete

This method returns the actual auto delete setting from the XSet.

| HTTP Attribute | Type                     | Example                                | Description                                                             |
|----------------|--------------------------|----------------------------------------|-------------------------------------------------------------------------|
| Method         | /XSetGetActualAutoDelete | GET / XSetGetActualAutoDelete HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle               | handle=2000202                         | Identifies the XSet instance on which the operation should be performed |

Response:

| Name       | Type        | Example                    | Description                                                                                                             |
|------------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status     | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg  | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| autodelete | autodelete  | autodelete:true            | The Boolean return value from the XSet.getActualAutoDelete() method.                                                    |

XSetGetActualRetention  
Duration

This method returns the actual auto delete setting from the XSet.

| HTTP Attribute | Type                              | Example                                        | Description                                                             |
|----------------|-----------------------------------|------------------------------------------------|-------------------------------------------------------------------------|
| Method         | / XSetGetActualRetention Duration | GET / XSetGetActualRetention Duration HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle                        | handle=2000202                                 | Identifies the XSet instance on which the operation should be performed |

Response:

| Name       | Type        | Example                    | Description                                                                                                             |
|------------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status     | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg  | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| autodelete | autodelete  | autodelete:true            | The Boolean return value from the XSet.getActualAutoDelete() method.                                                    |

XSetGetActualRetention  
Enabled

This method returns the actual retention enabled setting from the XSet.

| HTTP Attribute | Type                           | Example                                     | Description                                                             |
|----------------|--------------------------------|---------------------------------------------|-------------------------------------------------------------------------|
| Method         | /XSetGetActualRetentionEnabled | GET /XSetGetActualRetentionEnabled HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle                     | handle=2000202                              | Identifies the XSet instance on which the operation should be performed |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| enabled   | xam_boolean | enabled=true               | The Boolean return value from the XSet.getActualRetentionEnabled() method.                                              |

XSetGetActualShred

This method returns the actual shred setting from the XSet.

| HTTP Attribute | Type                | Example                          | Description                                                             |
|----------------|---------------------|----------------------------------|-------------------------------------------------------------------------|
| Method         | /XSetGetActualShred | GET /XSetGetActualShred HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle          | handle=2000202                   | Identifies the XSet instance on which the operation should be performed |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| shred     | xam_boolean | shred:true                 | The Boolean return value from the XSet.getActualShred() method.                                                         |

XSetGetFieldBinding This method returns the field binding setting from the XSet.

| HTTP Attribute | Type                 | Example                           | Description                                                             |
|----------------|----------------------|-----------------------------------|-------------------------------------------------------------------------|
| Method         | /XSetGetFieldBinding | GET /XSetGetFieldBinding HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle           | handle=2000202                    | Identifies the XSet instance on which the operation should be performed |
| name           | xam_string           | name=com.example.field            | The name of the field to interrogate                                    |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| value     | xam_boolean | value=true                 | The Boolean return value from the XSet.getFieldBinding() method.                                                        |

XSetGetLength This method returns the actual field length setting from the XSet.

| HTTP Attribute | Type                | Example                          | Description                                                             |
|----------------|---------------------|----------------------------------|-------------------------------------------------------------------------|
| Method         | /XSetGetFieldLength | GET /XSetGetFieldLength HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle          | handle=2000202                   | Identifies the XSet instance on which the operation should be performed |
| name           | xam_string          | name=com.example.field           | The name of the field to interrogate                                    |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| value     | xam_integer | value=73                   | The integer return value from the XSet.getFieldLength() method.                                                         |

XSetGetFieldReadOnly

This method returns the field readonly setting from the XSet.

| HTTP Attribute | Type                  | Example                             | Description                                                             |
|----------------|-----------------------|-------------------------------------|-------------------------------------------------------------------------|
| Method         | /XSetGetFieldReadOnly | GET / XSetGetFieldReadOnly HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle            | handle=2000202                      | Identifies the XSet instance on which the operation should be performed |
| name           | xam_string            | name=com.example.field              | The name of the field to interrogate                                    |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| value     | xam_boolean | value=true                 | The Boolean return value from the XSet.getFieldReadOnly() method                                                        |

XSetGetProperty

The XSetGetProperty operation does not have a direct parallel in the [XAM-ARCH]. Instead, this is an “aggregate” operation that is overloaded to simplify the creation of the handlers on each side of the wire. This operation returns a

string value that is convertible into the requested property type using the Java type conversion rules.

| HTTP Attribute | Type                                                                                                        | Example                       | Description                                                                                         |
|----------------|-------------------------------------------------------------------------------------------------------------|-------------------------------|-----------------------------------------------------------------------------------------------------|
| Method         | /XSetGetProperty                                                                                            | GET /XSetGetProperty HTTP/1.1 | Identifies operation and scope                                                                      |
| handle         | xam_handle                                                                                                  | handle=2000202                | Identifies the XSet instance on which the operation should be performed                             |
| ptype          | xam_string<br>Possible values include:<br>- boolean<br>- int<br>- float<br>- xuid<br>- string<br>- datetime | ptype=boolean                 | The type of property to create. The type of property will dictate the format of the value property. |
| name           | xam_string                                                                                                  | name=propertyname             | The name of the property to set                                                                     |

The response to the request will contain no additional information within the HTTP Header beyond normal response codes identified in [HTTP-RESPONSE]. The body will contain the following information in name/value pairs.

| Name      | Type        | Example                    | Description                                                                                                                      |
|-----------|-------------|----------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG.          |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                                      |
| value     | xam_string  | value=true                 | The value that the property is set to. This will always be a string, but the string must be convertible into the property ptype. |

#### XSetGetPropertyType

This method returns the field type setting from the XSet. Despite the naming, this method actually executes the getFieldtype() method on the specified XSet.

| HTTP Attribute | Type                 | Example                           | Description                                                             |
|----------------|----------------------|-----------------------------------|-------------------------------------------------------------------------|
| Method         | /XSetGetPropertyType | GET /XSetGetPropertyType HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle           | handle=2000202                    | Identifies the XSet instance on which the operation should be performed |
| name           | xam_string           | name=com.example.field            | The name of the field to interrogate                                    |

Response:

| Name      | Type        | Example                      | Description                                                                                                             |
|-----------|-------------|------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                     | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed   | A string message associated with the status code. Not returned if status=0.                                             |
| value     | xam_string  | value=application/octet-type | The MIME content type of the named field.                                                                               |

**XSetHaltJob** This method executes the method haltJob() on the XSet.

| HTTP Attribute | Type         | Example                   | Description                                                             |
|----------------|--------------|---------------------------|-------------------------------------------------------------------------|
| Method         | /XSetHaltJob | GET /XSetHaltJob HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle   | handle=2000202            | Identifies the XSet instance on which the operation should be performed |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

**XSetOpenExportStream** This method returns the handle of an export XStream on the XSet.

| HTTP Attribute | Type                  | Example                            | Description                                                             |
|----------------|-----------------------|------------------------------------|-------------------------------------------------------------------------|
| Method         | /XSetOpenExportStream | GET /XSetOpenExportStream HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle            | handle=2000202                     | Identifies the XSet instance on which the operation should be performed |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| handle    | xam_handle  | handle=2000202             | The handle of the newly created export stream.                                                                          |

XSetOpenImportStream

This method returns the handle of an import stream on the XSet.

| HTTP Attribute | Type                  | Example                             | Description                                                             |
|----------------|-----------------------|-------------------------------------|-------------------------------------------------------------------------|
| Method         | /XSetOpenImportStream | GET / XSetOpenImportStream HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle            | handle=2000202                      | Identifies the XSet instance on which the operation should be performed |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| handle    | xam_handle  | handle=2000202             | The handle of the newly created import stream.                                                                          |

XSetOpenXStream

Opens an existing XStream within a particular XSet with the values that were passed to this function. This operation also creates a handle for the operation and returns it to the caller.

| HTTP Attribute | Type                                                | Example                       | Description                                                             |
|----------------|-----------------------------------------------------|-------------------------------|-------------------------------------------------------------------------|
| Method         | /XSetOpenXStream                                    | GET /XSetOpenXStream HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle                                          | handle=2000202                | Identifies the XSet instance on which the operation should be performed |
| name           | xam_string                                          | name=com.example.name         | The name of the XStream to open                                         |
| mode           | xam_string where value is "readonly" or "writeonly" | mode=readonly                 | The mode in which the XStream should be opened                          |



Response:

| Name      | Type        | Example                    | Description                                                                                                                                                               |
|-----------|-------------|----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG.                                                   |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                                                                               |
| handle    | xam_handle  | handle = 33333333          | Identifies the XStream instance that was opened. Future operations on the XStream must include this handle. Use of this handle IMPLIES a connection to a particular XSet. |

XSetResetAccessFields This method executes the method resetAccessFields() on the XSet.

| HTTP Attribute | Type                   | Example                              | Description                                                             |
|----------------|------------------------|--------------------------------------|-------------------------------------------------------------------------|
| Method         | /XSetResetAccessFields | GET / XSetResetAccessFields HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle             | handle=2000202                       | Identifies the XSet instance on which the operation should be performed |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

XSetResetManagement Fields This method executes the method resetManagementFields() on the XSet.

| HTTP Attribute | Type                         | Example                                   | Description                                                             |
|----------------|------------------------------|-------------------------------------------|-------------------------------------------------------------------------|
| Method         | / XSetResetManagement Fields | GET / XSetResetManagement Fields HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle                   | handle=2000202                            | Identifies the XSet instance on which the operation should be performed |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

XSetSetAutoDelete This method sets the explicit AutoDelete setting on the XSet.

| HTTP Attribute | Type               | Example                         | Description                                                             |
|----------------|--------------------|---------------------------------|-------------------------------------------------------------------------|
| Method         | /XSetSetAutoDelete | GET /XSetSetAutoDelete HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle         | handle=2000202                  | Identifies the XSet instance on which the operation should be performed |
| autodelete     | xam_boolean        | autodelete=true                 | The auto delete setting                                                 |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

XSetSetBaseRetention This method sets the base retention on the XSet.

| HTTP Attribute | Type                  | Example                            | Description                                                             |
|----------------|-----------------------|------------------------------------|-------------------------------------------------------------------------|
| Method         | /XSetSetBaseRetention | GET /XSetSetBaseRetention HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle            | handle=2000202                     | Identifies the XSet instance on which the operation should be performed |
| binding        | xam_boolean           | binding=false                      | The binding setting for the base retention.                             |
| duration       | xam_integer           | duration=1000                      | The duration value for the base retention.                              |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

XSetSetFieldAsBinding

This method sets the named field to be bound on the XSet.

| HTTP Attribute | Type                   | Example                              | Description                                                             |
|----------------|------------------------|--------------------------------------|-------------------------------------------------------------------------|
| Method         | /XSetSetFieldAsBinding | GET / XSetSetFieldAsBinding HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle             | handle=2000202                       | Identifies the XSet instance on which the operation should be performed |
| name           | xam_string             | name=com.example.field               | The name of the field to be bound.                                      |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

XSetSetFieldAsNonbinding

This method returns the handle of an import stream on the XSet.

| HTTP Attribute | Type                       | Example                                 | Description                                                             |
|----------------|----------------------------|-----------------------------------------|-------------------------------------------------------------------------|
| Method         | / XSetSetFieldAsNonbinding | GET / XSetSetFieldAsNonbinding HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle                 | handle=2000202                          | Identifies the XSet instance on which the operation should be performed |
| name           | xam_string                 | name=com.example.field                  | The name of the field to be unbound                                     |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

### XSetSetProperty

The XSetSetProperty operation does not have a direct parallel in the [XAM-ARCH]. Instead, this is an “aggregate” operation that is overloaded to simplify the creation of the handlers on each side of the wire. All possible operations for setting properties are collapsed to this single call. As all information on the wire can be treated as string information, the value on the wire must be convertible from a string to the target property type. The SetProperty operation will return parameter errors in the status if the value is not readily convertible using the Java type conversion rules.

| HTTP Attribute | Type                                                                                                        | Example                       | Description                                                                                                                                  |
|----------------|-------------------------------------------------------------------------------------------------------------|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| Method         | /XSetSetProperty                                                                                            | GET /XSetSetProperty HTTP/1.1 | Identifies operation and scope                                                                                                               |
| handle         | xam_handle                                                                                                  | handle=2000202                | Identifies the XSet instance on which the operation should be performed                                                                      |
| ptype          | xam_string<br>Possible values include:<br>- boolean<br>- int<br>- float<br>- xuid<br>- string<br>- datetime | ptype=boolean                 | The type of property to create. The type of property will dictate the format of the value property.                                          |
| name           | xam_string                                                                                                  | name=propertyname             | The name of the property to set                                                                                                              |
| value          | various                                                                                                     | value=true                    | The value to which the property should be set. This value will always be a string, but the string must be convertible to the property ptype. |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

XSetSetRetentionDuration This method returns the handle of an import stream on the XSet.

| HTTP Attribute | Type                      | Example                                 | Description                                                             |
|----------------|---------------------------|-----------------------------------------|-------------------------------------------------------------------------|
| Method         | /XSetSetRetentionDuration | GET / XSetSetRetentionDuration HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle                | handle=2000202                          | Identifies the XSet instance on which the operation should be performed |
| retentionid    | xam_string                | retentionid=test_retention              | The name of the retention to modify                                     |
| duration       | xam_integer               | duration=1000                           | The new retention duration value to be set                              |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

XSetSetRetentionEnabled Flag This method returns the handle of an import stream on the XSet.

| HTTP Attribute | Type                         | Example                                    | Description                                                             |
|----------------|------------------------------|--------------------------------------------|-------------------------------------------------------------------------|
| Method         | /XSetSetRetentionEnabledFlag | GET / XSetSetRetentionEnabledFlag HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle                   | handle=2000202                             | Identifies the XSet instance on which the operation should be performed |
| retentionid    | xam_string                   | retentionid=test_retention                 | The name of the retention to modify                                     |
| enabled        | xam_boolean                  | enabled=true                               | The new enabled value of the retention                                  |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

**XSetSetRetentionStarttime** This method executes the `setRetentionStarttime()` method on the XSet.

| HTTP Attribute | Type                       | Example                                  | Description                                                             |
|----------------|----------------------------|------------------------------------------|-------------------------------------------------------------------------|
| Method         | /XSetSetRetentionStarttime | GET / XSetSetRetentionStarttime HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle                 | handle=2000202                           | Identifies the XSet instance on which the operation should be performed |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

**XSetSetShred** This method sets the shred setting on the XSet.

| HTTP Attribute | Type          | Example                    | Description                                                             |
|----------------|---------------|----------------------------|-------------------------------------------------------------------------|
| Method         | /XSetSetShred | GET /XSetSetShred HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle    | handle=2000202             | Identifies the XSet instance on which the operation should be performed |
| shred          | xam_boolean   | shred=true                 | The new shred value to be set on the XSet                               |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

**XSetSubmitJob** This method executes the `submitJob()` method on the XSet.

| HTTP Attribute | Type           | Example                     | Description                                                             |
|----------------|----------------|-----------------------------|-------------------------------------------------------------------------|
| Method         | /XSetSubmitJob | GET /XSetSubmitJob HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle     | handle=2000202              | Identifies the XSet instance on which the operation should be performed |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

XStreamAbandon

This operation abandons an XStream and all of the resources associated with it. The XStream handle implies an association with an XSet. After this call, the handle will no longer identify an active XStream object instance. The XStream will have to be re-opened, and a new handle will have to be retrieved.

| HTTP Attribute | Type            | Example                      | Description                                                                |
|----------------|-----------------|------------------------------|----------------------------------------------------------------------------|
| Method         | /XStreamAbandon | GET /XStreamAbandon HTTP/1.1 | Identifies operation and scope                                             |
| handle         | xam_handle      | handle=33333333              | Identifies the XStream instance on which the operation should be performed |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

XStreamAsyncClose

This creates an asynchronous XStream operation. The body will contain the following information in name/value pairs.

| HTTP Attribute | Type               | Example                         | Description                                    |
|----------------|--------------------|---------------------------------|------------------------------------------------|
| Method         | /XStreamAsyncClose | GET /XStreamAsyncClose HTTP/1.1 | Identifies operation and scope                 |
| handle         | xam_handle         | handle=2000202                  | The handle of the XSet                         |
| xopid          | xam_integer        | xopid=678890                    | The xopid to be associated with this operation |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| handle    | xam_handle  | handle=2000202             | The handle of the XAsync object.                                                                                        |

#### XStreamAsyncRead

This method creates an asynchronous XStream operation. At some point later, the number of bytes read will be available, and bytes themselves will be available. The client must retrieve the data with the method XAsyncGetData. The body will contain the following information in name/value pairs.

| HTTP Attribute | Type              | Example                        | Description                                     |
|----------------|-------------------|--------------------------------|-------------------------------------------------|
| Method         | /XStreamAsyncRead | GET /XStreamAsyncRead HTTP/1.1 | Identifies operation and scope                  |
| handle         | xam_handle        | handle=2000202                 | The handle of the XSet                          |
| xopid          | xam_integer       | xopid=678890                   | The xopid to be associated with this operation. |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| handle    | xam_handle  | handle=2000202             | The handle of the XAsync object.                                                                                        |

#### XStreamAsyncWrite

This operation starts an asynchronous write operation to an XStream. The write of this buffer will start sequentially after the last byte written or after the position of the last byte of the stream (if the seek operation is supported). The XAsync method will return as soon as the data has been transferred.

| HTTP Attribute | Type               | Example                          | Description                                                                |
|----------------|--------------------|----------------------------------|----------------------------------------------------------------------------|
| Method         | /XStreamAsyncWrite | POST /XStreamAsyncWrite HTTP/1.1 | Identifies operation and scope                                             |
| handle         | xam_handle         | handle=33333333                  | Identifies the XStream instance on which the operation should be performed |

The body will contain the bytes that should be written to the XStream.



Response:

| Name   | Type         | Example        | Description                                                                                                                                                                                                                                                                                                                       |
|--------|--------------|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| status | status=value | status=0       | Gives the status of the connection. The status uses the standard or extended error codes as defined by the [XAM-ARCH], [XAM-C-API] and/or the vim.h header files. The Java API does not leverage status codes in its internal communication, and the Exceptions will have to be converted appropriately to standard status codes. |
| handle | xam_handle   | handle=2000202 | The handle of the XAsync object.                                                                                                                                                                                                                                                                                                  |

**XStreamClose** This operation closes an XStream and all of the resources associated with it. The XStream handle implies an association with an XSet. Note that the handle to the XStream “may” still be valid after this operation, depending on the status returned. If the operation is successful, the handle will never be valid after the operation completes.

| HTTP Attribute | Type                                                                                      | Example                     | Description                                                                |
|----------------|-------------------------------------------------------------------------------------------|-----------------------------|----------------------------------------------------------------------------|
| Method         | /XStreamClose                                                                             | GET /XStreamClose HTTP/ 1.1 | Identifies operation and scope                                             |
| handle         | handle: value where “value” is an 8-byte signed integer, we will use the “long” Java type | handle: 33333333            | Identifies the XStream instance on which the operation should be performed |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

**XStreamRead** This operation reads from an XStream. For the purpose of the first iteration, the size of the requested read is not possible; only the entire XStream can be read.

| HTTP Attribute | Type         | Example                    | Description                                                                |
|----------------|--------------|----------------------------|----------------------------------------------------------------------------|
| Method         | /XStreamRead | GET /XStreamRead HTTP/ 1.1 | Identifies operation and scope                                             |
| handle         | xam_handle   | handle=33333333            | Identifies the XStream instance on which the operation should be performed |

The response to the request will contain the contents of the XStream in the body. If no body is included, an error occurred in the response.

**XStreamSeek** This operation executes the seek() method on the specified XStream object.

| HTTP Attribute | Type         | Example                   | Description                                                                |
|----------------|--------------|---------------------------|----------------------------------------------------------------------------|
| Method         | /XStreamSeek | GET /XStreamSeek HTTP/1.1 | Identifies operation and scope                                             |
| handle         | xam_handle   | handle=33333333           | Identifies the XStream instance on which the operation should be performed |
| whence         | xam_integer  | whence=0                  | The whence value from the seek method.                                     |
| offset         | xam_integer  | offset=12                 | The number of bytes to seek, relative to whence.                           |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

**XStreamTell** This operation executes the tell() method on the specified XStream object.

| HTTP Attribute | Type         | Example                   | Description                                                                |
|----------------|--------------|---------------------------|----------------------------------------------------------------------------|
| Method         | /XStreamTell | GET /XStreamTell HTTP/1.1 | Identifies operation and scope                                             |
| handle         | xam_handle   | handle=33333333           | Identifies the XStream instance on which the operation should be performed |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| position  | xam_integer | position=42                | Byte offset relative to the beginning of the XStream                                                                    |

|              |                                                                                                                                                                                                              |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| XStreamWrite | This operation writes to an XStream. The write of this buffer will start sequentially after the last byte written or after the position of the last byte of the stream (if the seek operation is supported). |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

| HTTP Attribute | Type          | Example                     | Description                                                                |
|----------------|---------------|-----------------------------|----------------------------------------------------------------------------|
| Method         | /XStreamWrite | POST /XStreamWrite HTTP/1.1 | Identifies operation and scope                                             |
| handle         | xam_handle    | handle=33333333             | Identifies the XStream instance on which the operation should be performed |

The body will contain the bytes that should be written to the XStream.

The response to the request will contain no additional information within the HTTP Header beyond normal response codes identified in [HTTP-RESPONSE]. The body will contain the following information in name/value pairs.

| Name         | Type        | Example                    | Description                                                                                                             |
|--------------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status       | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg    | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| byteswritten | xam_int     | byteswritten=1000          | The number of bytes written to the XStream                                                                              |

A complete request may look like the following:

```
POST /XStreamWrite HTTP/1.1  
handle: 33333333  
User-Agent: Jakarta Commons-HttpClient/3.0.1  
Host: 192.168.1.100:9925  
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaccccccccbbbb
```

A complete response may look like the following:

```
HTTP/1.1 200 OK
Date: Wed, 22 Aug 2007 12:45:13 GMT
Server: Jetty/4.2.20 (Windows XP/5.1 x86 java/
1.6.0_02)
Transfer-Encoding: chunked
status=0
byteswritten=1000
```

|                |                                                                                                                                                        |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| XSystemAbandon | This operation releases all resources used by an XSystem and the XSystem itself. The XSystem cannot be used after the call. The request is expected to |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|

be a generic HTTP GET request with the request parameters in the HTTP Header; there is no information contained in the HTTP Body for the request.

| HTTP Attribute | Type            | Example                      | Description                                                                |
|----------------|-----------------|------------------------------|----------------------------------------------------------------------------|
| Method         | /XSystemAbandon | GET /XSystemAbandon HTTP/1.1 | Identifies operation and scope                                             |
| handle         | xam_handle      | handle=1010101               | Identifies the XSystem instance on which the operation should be performed |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

#### XSystemAccessXSet

This operation determines if a particular XSet is accessible with a particular mode. This operation does not reserve the XSet, so the state may change before your open. The XSystem must be connected to and a valid handle to that XSystem supplied. This operation on the HTTP VIM passes through to the implementation VIM.

| HTTP Attribute | Type               | Example                         | Description                                                                                                 |
|----------------|--------------------|---------------------------------|-------------------------------------------------------------------------------------------------------------|
| Method         | /XSystemAccessXSet | GET /XSystemAccessXSet HTTP/1.1 | Identifies operation and scope                                                                              |
| xuid           | xam_xuid           | xuid=AAA....                    | Identifies the XSet that should be opened for the client. XUID value to be base64 encoded.                  |
| handle         | xam_handle         | handle=1010101                  | Identifies the XSystem instance on which the operation should be performed                                  |
| mode           | xam_string         | mode=readonly                   | Identifies the mode in which the XSet should be opened. Different language bindings vary dramatically here. |

Response:

| Name       | Type        | Example                    | Description                                                                                                             |
|------------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status     | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg  | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| accessible | xam_boolean | accessible=true            |                                                                                                                         |

**XSystemAsyncCopyXSet**

This operation initiates an asynchronous CopyXSet method on the specified XSystem object.

| HTTP Attribute | Type                  | Example                             | Description                                                                |
|----------------|-----------------------|-------------------------------------|----------------------------------------------------------------------------|
| Method         | /XSystemAsyncCopyXSet | GET / XSystemAsyncCopyXSet HTTP/1.1 | Identifies operation and scope                                             |
| handle         | xam_handle            | handle=33333333                     | Identifies the XSystem instance on which the operation should be performed |
| xuid           | xam_xuid              | xuid=AAA.....                       | The base64/URL safe encoded XUID value.                                    |
| mode           | xam_string            | mode=restricted                     | The copy mode argument                                                     |
| xopid          | xam_int               | xopid=1357                          | The XOPID argument from the AsyncCopyXSet method.                          |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| handle    | xam_handle  | handle=2000202             | The handle of the XAsync Object                                                                                         |

**XSystemAsyncOpenXSet**

This operation initiates an asynchronous OpenXSet method on the specified XSystem object.

| HTTP Attribute | Type                  | Example                             | Description                                                                |
|----------------|-----------------------|-------------------------------------|----------------------------------------------------------------------------|
| Method         | /XSystemAsyncOpenXSet | GET / XSystemAsyncOpenXSet HTTP/1.1 | Identifies operation and scope                                             |
| handle         | xam_handle            | handle=33333333                     | Identifies the XSystem instance on which the operation should be performed |
| xuid           | xam_xuid              | xuid=AAA.....                       | The base64/URL safe encoded XUID value                                     |
| mode           | xam_string            | mode=restricted                     | The copy mode argument                                                     |
| xopid          | xam_int               | xopid=1357                          | The XOPID argument from the AsyncOpenXSet method                           |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| handle    | xam_handle  | handle=2000202             | The handle of the XAsync Object                                                                                         |

SystemAsyncOpenXStream

This operation initiates an asynchronous OpenXStream method on the specified XSystem object.

| HTTP Attribute | Type                     | Example                                | Description                                                                |
|----------------|--------------------------|----------------------------------------|----------------------------------------------------------------------------|
| Method         | /XSystemAsyncOpenXStream | GET / XSystemAsyncOpenXStream HTTP/1.1 | Identifies operation and scope                                             |
| handle         | xam_handle               | handle=33333333                        | Identifies the XSystem instance on which the operation should be performed |
| name           | xam_string               | name=com.example.stream                | The name of the stream to open                                             |
| mode           | xam_string               | mode=restricted                        | The open mode argument                                                     |
| xopid          | xam_int                  | xopid=1357                             | The XOPID argument from the AsyncOpenXSet method                           |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| handle    | xam_handle  | handle=2000202             | The handle of the XAsync Object                                                                                         |

**XSystemAuthenticate** This operation calls the authenticate method on the specified XSystem. The contents of the authenticate buffer are sent as body of the POST operation.

| HTTP Attribute | Type                 | Example                             | Description                                                                |
|----------------|----------------------|-------------------------------------|----------------------------------------------------------------------------|
| Method         | /XSystemAuthenticate | POST / XSystemAuthenticate HTTP/1.1 | Identifies operation and scope                                             |
| handle         | xam_handle           | handle=33333333                     | Identifies the XSystem instance on which the operation should be performed |

The body will contain the bytes that should be passed to the authenticate.

#### Response

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| handle    | xam_handle  | handle=2000202             | A handle to the XStream containing the results from the authenticate call.                                              |

A complete request may look like the following:

```
POST /XStreamWrite HTTP/1.1
handle: 33333333
User-Agent: Jakarta Commons-HttpClient/3.0.1
Host: 192.168.1.100:9925
<NUL>Users<NUL>Password
```

A complete response may look like the following:

```
HTTP/1.1 200 OK
Date: Wed, 22 Aug 2007 12:45:13 GMT
Server: Jetty/4.2.20 (Windows XP/5.1 x86 java/1.6.0_02)
Transfer-Encoding: chunked
status=0
handle=2000202
```

**XSystemClose** This operation releases all resources used by an XSystem and the XSystem itself. The XSystem cannot be used after the call, and the handle will no longer identify a valid XSystem object instance. The request is expected to be

a generic HTTP GET request with the request parameters in the HTTP Header; there is no information contained in the HTTP Body for the request.

| HTTP Attribute | Type          | Example                    | Description                                                                |
|----------------|---------------|----------------------------|----------------------------------------------------------------------------|
| Method         | /XSystemClose | GET /XSystemClose HTTP/1.1 | Identifies operation and scope                                             |
| handle         | xam_handle    | handle=1010101             | Identifies the XSystem instance on which the operation should be performed |

The response to the request will contain no additional information within the HTTP Header beyond normal response codes identified in [HTTP-RESPONSE]. The body will contain the following information in name/value pairs.

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

#### XSystemConnect

This operation establishes a XAM connection to a system. The request is expected to be a generic HTTP GET request with the request parameters in the HTTP Header; there is no information contained in the HTTP Body. As the HTTP protocol is stateless, the result of this operation will be a “handle” returned to the requester. Future requests must include this handle to identify the active XSystem object.

| HTTP Attribute | Type                  | Example                       | Description                                       |
|----------------|-----------------------|-------------------------------|---------------------------------------------------|
| Method         | /XSystemConnect       | POST /XSystemConnect HTTP/1.1 | Identifies the operation as a /Connect operation  |
| xri            | xri: snia-xam://value | xri: snia-xam://localhost     | Identifies to which XSystem to locate and connect |

The body of the post must contain all of the initialization parameters for the XSystem. These parameters include write properties to the XSystem and are passed to the XSystem as part of the XSystem initialization protocol. Each parameter is a name value pair, three of which are needed to completely specify the semantics of an XSystem property: name, value, binding, and type. For example, an XLibrary is going to initialize the XSystem with the following properties:

```
com.example.p1,a string, value=absdefg
com.example.p2,an integer, value=42
com.example.p3,a boolean, value=false
```

A buffer containing these values is sent to in the POST body:

```
com.example.p1.type=string
com.example.p1.value=absdefg
com.example.p1.binding=false
com.example.p2.type=integer com.example.p2.value=42
```



```
com.example.p2.bdining=false
com.example.p3.type=boolean com.example.p3.value=false
com.example.p3.binding=false
```

Note that each name/value pair is space delimited. Thus, any name/value pair must be URL encoded to prevent spaces in a value from confusing the parser. Also note that as of the current level of implementation, there are no known VIMs which provide binding values for properties, but this may not be true for the future.

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| handle    | xam_handle  | handle=1010101             | Identifies the XSystem for future operations                                                                            |

A complete request may look like the following:

```
GET /XAMLIBConnect HTTP/1.1
xri: snia-xam://localhost
User-Agent: Jakarta Commons-HttpClient/3.0.1
Host: 192.168.1.100:9925

com.example.p1.type=string
com.example.p1.value=absdefg
com.example.p1.binding=false
com.example.p2.type=integer com.example.p2.value=42
com.example.p2.bdining=false
com.example.p3.type=boolean com.example.p3.value=false
com.example.p3.binding=false
```

A complete response may look like the following:

```
HTTP/1.1 200 OK
Date: Wed, 22 Aug 2007 12:45:13 GMT
Server: Jetty/4.2.20 (Windows XP/5.1 x86 java/
1.6.0_02)
Transfer-Encoding: chunked

status=1020
handle=1010101
```

**XSystemContainsField** This operation initiates an asynchronous OpenXStream method on the specified XSystem object.

| HTTP Attribute | Type                  | Example                             | Description                                                                |
|----------------|-----------------------|-------------------------------------|----------------------------------------------------------------------------|
| Method         | /XSystemContainsField | GET / XSystemContainsField HTTP/1.1 | Identifies operation and scope                                             |
| handle         | xam_handle            | handle=33333333                     | Identifies the XSystem instance on which the operation should be performed |
| name           | xam_string            | name=com.example.stream             | The name of the field to test                                              |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| value     | xam_booleam | value=true                 | The Boolean return value from the XSystem.containsField() method.                                                       |

**XSystemCopyXSet** This operation executes the method CopyXSet on the specified XSystem object.

| HTTP Attribute | Type             | Example                       | Description                                                                |
|----------------|------------------|-------------------------------|----------------------------------------------------------------------------|
| Method         | /XSystemCopyXSet | GET /XSystemCopyXSet HTTP/1.1 | Identifies operation and scope                                             |
| handle         | xam_handle       | handle=33333333               | Identifies the XSystem instance on which the operation should be performed |
| xuid           | xam_xuid         | xuid=AAA...                   | The base64/URL encoded value of the XUID                                   |
| mode           | xam_string       | mode=restricted               | The open mode of the copied XSet                                           |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| handle    | xam_handle  | handle=2000202             | The handle to the newly copied XSet.                                                                                    |

**XSystemCreateProperty**

The XSystemCreateProperty operation does not have a direct parallel in the [XAM-ARCH]. Instead, this is an “aggregate” operation that is overloaded to simplify the creation of the handlers on each side of the wire. This version of the wire format operates on a system rather than an XSet. This may be able to be collapsed into the XSetCreateProperty with an additional value in the HTTP Header on the request.

| HTTP Attribute | Type                                                                                                   | Example                                    | Description                                                                                                                                  |
|----------------|--------------------------------------------------------------------------------------------------------|--------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| Method         | /XSystemCreateProperty                                                                                 | GET /<br>XSystemCreateProperty<br>HTTP/1.1 | Identifies operation and scope                                                                                                               |
| handle         | xam_handle                                                                                             | handle=1010101                             | Identifies the XSystem instance on which the operation should be performed                                                                   |
| ptype          | xam_string<br>Possible values include:<br>- boolean<br>- int<br>- float<br>- xuid string<br>- datetime | ptype=boolean                              | The type of property to create. The type of property will dictate the format of the value property.                                          |
| name           | xam_string                                                                                             | name=propertyname                          | The name of the property to set                                                                                                              |
| binding        | xam_boolean                                                                                            | binding=true                               | Whether this field should be binding                                                                                                         |
| value          | xam_string                                                                                             | value=true                                 | The value to which the property should be set. This value will always be a string, but the string must be convertible to the property ptype. |

The response to the request will contain no additional information within the HTTP Header beyond normal response codes identified in [HTTP-RESPONSE]. The body will contain the following information in name/value pairs.

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

**XSystemCreateXSet**

This operation creates an XSet within a specified XSystem. The XSystem must be connected to and must have a valid handle to the XSystem supplied.

This operation returns a handle to the requester that must be used for future interactions with the created XSet.

| HTTP Attribute | Type               | Example                         | Description                                                                |
|----------------|--------------------|---------------------------------|----------------------------------------------------------------------------|
| Method         | /XSystemCreateXSet | GET /XSystemCreateXSet HTTP/1.1 | Identifies operation and scope                                             |
| handle         | xam_handle         | handle=1010101                  | Identifies the XSystem instance on which the operation should be performed |
| mode           | xam_string         | mode=readonly                   | Identifies the mode in which the XSet should be opened                     |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| handle    | xam_handle  | handle = 2000202           | Identifies the XSet instance that was created. Future operations on the XSet must include this handle.                  |

### XSystemCreateXStream

This operation creates an XStream within a specified XSystem. The XSystem must be connected to and must be a valid handle to the XSystem supplied. This operation returns a handle to the requester that must be used for future interactions with the created XStream. As of XAM version 1.0.1, no known XSystems support stream creation.

| HTTP Attribute | Type                  | Example                             | Description                                                                |
|----------------|-----------------------|-------------------------------------|----------------------------------------------------------------------------|
| Method         | /XSystemCreateXStream | GET / XSystemCreateXStream HTTP/1.1 | Identifies operation and scope                                             |
| handle         | xam_handle            | handle=1010101                      | Identifies the XSystem instance on which the operation should be performed |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| handle    | xam_handle  | handle = 2000202           | Identifies the XStream instance that was created. Future operations on the XStream must include this handle.            |

**XSystemDeleteXSet** This operation deletes an existing XSet within a specified XSystem. The XSystem must be connected to and must be a valid handle to the XSystem supplied. This operation on the HTTP VIM passes through to the implementation VIM.

| HTTP Attribute | Type               | Example                         | Description                                                                |
|----------------|--------------------|---------------------------------|----------------------------------------------------------------------------|
| Method         | /XSystemDeleteXSet | GET /XSystemDeleteXSet HTTP/1.1 | Identifies operation and scope                                             |
| xuid           | xam_xuid           | xuid=AAA...                     | Identifies the XSet that should be opened for the client.                  |
| handle         | xam_handle         | handle=1010101                  | Identifies the XSystem instance on which the operation should be performed |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

**XSystemGetFieldBinding** This method returns the field binding setting from the XSystem.

| HTTP Attribute | Type                    | Example                              | Description                                                                |
|----------------|-------------------------|--------------------------------------|----------------------------------------------------------------------------|
| Method         | /XSystemGetFieldBinding | GET /XSystemGetFieldBinding HTTP/1.1 | Identifies operation and scope                                             |
| handle         | xam_handle              | handle=2000202                       | Identifies the XSystem instance on which the operation should be performed |
| name           | xam_string              | name=com.example.field               | The name of the field to interrogate                                       |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| value     | xam_boolean | value=true                 | The Boolean return value from the XSystem.getFieldBinding() method                                                      |

**XSystemGetFieldLength** This method returns the actual field length setting from the XSet.

| HTTP Attribute | Type                   | Example                              | Description                                                             |
|----------------|------------------------|--------------------------------------|-------------------------------------------------------------------------|
| Method         | /XSystemGetFieldLength | GET / XSystemGetFieldLength HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle             | handle=2000202                       | Identifies the XSet instance on which the operation should be performed |
| name           | xam_string             | name=com.example.field               | The name of the field to interrogate                                    |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| value     | xam_integer | value=73                   | The integer return value from the XSet.getFieldLength() method                                                          |

**XSystemGetProperty** The XSystemGetProperty operation does not have a direct parallel in the [XAM-ARCH]. Instead, this is an “aggregate” operation that is overloaded to simplify the creation of the handlers on each side of the wire. This version of the wire format operates on a system rather than an XSet. This may be able to be collapsed into the XSetGetProperty with an additional value in the HTTP Header on the request. This operation returns a string value that is convertible into the requested property type using the Java type conversion rules.

| HTTP Attribute | Type                                                                                                        | Example                          | Description                                                                                         |
|----------------|-------------------------------------------------------------------------------------------------------------|----------------------------------|-----------------------------------------------------------------------------------------------------|
| Method         | /XSystemGetProperty                                                                                         | GET /XSystemGetProperty HTTP/1.1 | Identifies operation and scope                                                                      |
| handle         | xam_handle                                                                                                  | handle=1010101                   | Identifies the XSystem instance on which the operation should be performed                          |
| ptype          | xam_string<br>Possible values include:<br>- boolean<br>- int<br>- float<br>- xuid<br>- string<br>- datetime | ptype=boolean                    | The type of property to create. The type of property will dictate the format of the value property. |
| name           | xam_string                                                                                                  | name=propertyname                | The name of the property to set                                                                     |

Response:

| Name      | Type        | Example                    | Description                                                                                                                      |
|-----------|-------------|----------------------------|----------------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG.          |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                                      |
| value     | various     | value=true                 | The value that the property is set to. This will always be a string, but the string must be convertible into the property ptype. |

XSystemGetPropertyType

This method returns the field type setting from the XSet. Despite the naming, this method actually executes the getFieldTypes() method on the specified XSet.

| HTTP Attribute | Type                    | Example                               | Description                                                             |
|----------------|-------------------------|---------------------------------------|-------------------------------------------------------------------------|
| Method         | /XSystemGetPropertyType | GET / XSystemGetPropertyType HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle              | handle=2000202                        | Identifies the XSet instance on which the operation should be performed |
| name           | xam_string              | name=com.example.field                | The name of the field to interrogate                                    |

Response:

| Name      | Type        | Example                      | Description                                                                                                             |
|-----------|-------------|------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                     | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed   | A string message associated with the status code. Not returned if status=0.                                             |
| value     | xam_string  | value=application/octet-type | The MIME content type of the named field.                                                                               |

XSystemGetXSetAccess  
Time

This method returns the access time of the specified XSet.

| HTTP Attribute | Type                      | Example                                 | Description                                                             |
|----------------|---------------------------|-----------------------------------------|-------------------------------------------------------------------------|
| Method         | /XSystemGetXSetAccessTime | GET / XSystemGetXSetAccessTime HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle                | handle=2000202                          | Identifies the XSet instance on which the operation should be performed |
| xuid           | xam_xuid                  | xuid=AAA...                             | The base64/URL safe encoded XUID value.                                 |

Response:

| Name       | Type         | Example                    | Description                                                                                                             |
|------------|--------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status     | xam_integer  | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg  | xam_string   | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| accesstime | xam_datetime | asccesstime=12374568       | The date time of the last time the XSet was accessed.                                                                   |

XSystemHoldXSet This method executes the holdXSet() method on the XSystem

| HTTP Attribute | Type             | Example                       | Description                                                             |
|----------------|------------------|-------------------------------|-------------------------------------------------------------------------|
| Method         | /XSystemHoldXSet | GET /XSystemHoldXSet HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle       | handle=2000202                | Identifies the XSet instance on which the operation should be performed |
| holdid         | xam_string       | holdid=case_1234              | The holdID to be applied to the XSet.                                   |
| xuid           | xam_xuid         | xuid=AAAA.....                | The base64/URL safe encoded XUID value of the XSet.                     |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

XSystemIsXSetRetained This method executes the isXSetRetained() method on the XSystem.

| HTTP Attribute | Type                   | Example                             | Description                                                             |
|----------------|------------------------|-------------------------------------|-------------------------------------------------------------------------|
| Method         | /XSystemIsXSetRetained | GET /XSystemIsXSetRetained HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle             | handle=2000202                      | Identifies the XSet instance on which the operation should be performed |
| xuid           | xam_xuid               | xuid=AAAA.....                      | The base64/URL safe encoded XUID value of the XSet                      |



Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| retained  | xam_boolean | retained=true              | The Boolean return value from the isXSetRetained() method.                                                              |

#### XSystemOpenXSet

This operation opens an existing XSet within a specified XSystem. The XSystem must be connected to and a valid handle to the XSystem supplied. This operation on the HTTP VIM passes through to the implementation VIM. A handle is returned to the client that must be used to refer to the active XSet object in future operations on the XSet.

| HTTP Attribute | Type             | Example                       | Description                                                                |
|----------------|------------------|-------------------------------|----------------------------------------------------------------------------|
| Method         | /XSystemOpenXSet | GET /XSystemOpenXSet HTTP/1.1 | Identifies operation and scope                                             |
| xuid           | xam_xuid         | xuid=AAA....                  | Identifies the XSet that should be opened for the client                   |
| handle         | xam_handle       | handle=1010102                | Identifies the XSystem instance on which the operation should be performed |
| mode           | xam_string       | mode=readonly                 | Identifies the mode in which the XSet should be opened                     |

Response:

| Name      | Type                                                    | Example                    | Description                                                                                                             |
|-----------|---------------------------------------------------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer                                             | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string                                              | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| handle    | handle: value where "value" is an 8-byte signed integer | handle = 2000202           | Identifies the XSet instance that was opened. Future operations on the XSet must include this handle.                   |

#### XSystemOpenXStream

This operation opens an existing XStream within a specified XSystem. The XSystem must be connected to and a valid handle to that XSystem supplied. This operation on the HTTP VIM passes through to the implementation VIM.

A handle is returned to the client that must be used to refer to the active XSet object in future operations on the XSet.

| HTTP Attribute | Type                | Example                           | Description                                                                |
|----------------|---------------------|-----------------------------------|----------------------------------------------------------------------------|
| Method         | /XSystemOpenXStream | GET / XSystemOpenXStream HTTP/1.1 | Identifies operation and scope                                             |
| xuid           | xam_xuid            | xuid=AAA....                      | Identifies the XSet that should be opened for the client                   |
| handle         | xam_handle          | handle=1010102                    | Identifies the XSystem instance on which the operation should be performed |
| name           | xam_string          | name=com.example.name             | The stream name to be opened                                               |
| mode           | xam_string          | mode=readonly                     | Identifies the mode in which the XSet should be opened                     |

Response:

| Name      | Type                                                     | Example                    | Description                                                                                                             |
|-----------|----------------------------------------------------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer                                              | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string                                               | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |
| handle    | handle: value where "value" is an 8-byte signed integer. | handle = 2000202           | Identifies the XStream instance that was opened. Future operations on the XStream must include this handle.             |

### XSystemReleaseXSet

This method executes the releaseXSet() method on the XSystem

| HTTP Attribute | Type                | Example                          | Description                                                             |
|----------------|---------------------|----------------------------------|-------------------------------------------------------------------------|
| Method         | /XSystemReleaseXSet | GET /XSystemReleaseXSet HTTP/1.1 | Identifies operation and scope                                          |
| handle         | xam_handle          | handle=2000202                   | Identifies the XSet instance on which the operation should be performed |
| holdid         | xam_string          | holdid=case_1234                 | The holdID to be applied to the XSet                                    |
| xuid           | xam_xuid            | xuid=AAAA.....                   | The base64/URL safe encoded XUID value of the XSet                      |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

XSystemSetFieldAsBinding

This method sets the named field to be bound on the XSystem. As of XAM version 1.0.1, there is no definition as to what a bound XSystem field means. This method will pass execution to the XSystem, but it is expected that this operation will fail on current VIMs.

| HTTP Attribute | Type                      | Example                                 | Description                                                                |
|----------------|---------------------------|-----------------------------------------|----------------------------------------------------------------------------|
| Method         | /XSystemSetFieldAsBinding | GET / XSystemSetFieldAsBinding HTTP/1.1 | Identifies operation and scope                                             |
| handle         | xam_handle                | handle=2000202                          | Identifies the XSystem instance on which the operation should be performed |
| name           | xam_string                | name=com.example.field                  | The name of the field to be bound.                                         |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

XSystemSetFieldAsNon  
binding

This method sets the named field to be unbound on the XSystem. As of XAM version 1.0.1, there is no definition as to what a bound XSystem field means. This method will pass execution to the XSystem, but it is expected that this operation will fail on current VIMs.

| HTTP Attribute | Type                         | Example                                   | Description                                                                |
|----------------|------------------------------|-------------------------------------------|----------------------------------------------------------------------------|
| Method         | /XSystemSetFieldAsNonbinding | GET /XSystemSetFieldAsNonbinding HTTP/1.1 | Identifies operation and scope                                             |
| handle         | xam_handle                   | handle=2000202                            | Identifies the XSystem instance on which the operation should be performed |
| name           | xam_string                   | name=com.example.field                    | The name of the field to be unbound                                        |

Response:

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

### XSystemSetProperty

The XSystemSetProperty operation does not have a direct parallel in the [XAM-ARCH]. Instead, this is an “aggregate” operation that is overloaded to simplify the creation of the handlers on each side of the wire. This version of the wire format operates on a system rather than an XSet. This may be able to be collapsed into the XSystemSetProperty with an additional value in the HTTP Header on the request in a future version of this design specification. As all information on the wire can be treated as string information, the value on the wire must be convertible from a string to the target property type. The SetProperty operation will return parameter errors in the status if the value is not readily convertible using the Java type conversion rules.

| HTTP Attribute | Type                                                                                                        | Example                          | Description                                                                                                                                  |
|----------------|-------------------------------------------------------------------------------------------------------------|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| Method         | /XSystemSetProperty                                                                                         | GET /XSystemSetProperty HTTP/1.1 | Identifies operation and scope                                                                                                               |
| handle         | xam_handle                                                                                                  | handle=1010101                   | Identifies the XSystem instance on which the operation should be performed                                                                   |
| ptype          | xam_string<br>Possible values include:<br>- boolean<br>- int<br>- float<br>- xuid<br>- string<br>- datetime | ptype=boolean                    | The type of property to create. The type of property will dictate the format of the value property.                                          |
| name           | xam_string                                                                                                  | name=propertyname                | The name of the property to set                                                                                                              |
| value          | xam_string                                                                                                  | value=true                       | The value to which the property should be set. This value will always be a string, but the string must be convertible to the property ptype. |

The response to the request will contain no additional information within the HTTP Header beyond normal response codes identified in [HTTP-RESPONSE]. The body will contain the following information in name/value pairs.

| Name      | Type        | Example                    | Description                                                                                                             |
|-----------|-------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------|
| status    | xam_integer | status=0                   | Gives the status of the connection. The status uses the standard or extended error codes as defined by the XAM SDK TWG. |
| statusMsg | xam_string  | statusMsg=Object is closed | A string message associated with the status code. Not returned if status=0.                                             |

### *Known Issues*

The HTTP Protocol VIM has the following known issues:

- HTTPS is not currently implemented
- Handles are relatively unsecured, allowing potential clients to “guess” at handles and gain access to XAM objects that they might not otherwise have access to.
- No leasing mechanism is implemented. Improperly disconnected, or crash clients, will leave many orphaned objects in the caches of the VIM server. A possible way to solve this problem is to place a lease on XAM objects by adding a lease/last used properties to the XSystem.
- The HTTP VIM Server can periodically clean up XSystems that fail the lease time, freeing resources.
- XSystems would be abandoned, then closed.