Errata in "Information Management – Extensible Access Method (XAM) – Part 3: Java API" v1.0

SNIA FCAS TWG, June 16, 2009 – Approved Version

Summary of errata:

- 4.4.3 – Correct descriptions of VIM initialization in Figure 6
- 4.5.3 Query for data with the string literal, example code incorrect.

- 5.3.2.1 – XSystem.connect method should be public.
- 5.3.2.2 Authenticate call is missing return value, makes an erroneous statement.
- 5.3.2.9 XSet.accessXSet – Correct and add mode bits.
- 5.3.2.16/17 Add "createXUID" methods to XSystem.
- 5.3.3.3 applyManagementPolicy – Remove incorrect statement
- 5.3.3.4 resetManagementFields – Use correct field name
- 5.3.3.5 createRetention - Forbid "base" retention ID, require binding for "event"
- 5.3.3.6 setRetentionEnabledFlag – prohibit "base" retention ID, add exception for retention violation
- 5.3.3.7 applyRetentionEnabledPolicy – prohibit "base" retention ID, add exception for retention violation
- 5.3.3.8 setRetentionDuration – prohibit "base" retention ID, add exception for retention violation
- 5.3.3.9 applyRetentionDurationPolicy – prohibit "base" retention ID,
- 5.3.3.10 setRetentionStarttime – prohibit "base" retention ID,
- 5.3.3.11 setBaseRetention – Correct binding behavior and use correct field name
- 5.3.3.12 setBaseRetentionPolicy – Use correct field name, remove inappropriate exceptions
- 5.3.3.13 setAutoDelete – Use correct field name
- 5.3.3.14 applyAutoDeletePolicy – Use correct field name
- 5.3.3.15 setShred – Use correct field name
- 5.3.3.16 applyShredPolicy – Use correct field name
- 5.3.3.18 getActualRetentionDuration – Add PolicyMismatchException
- 5.3.3.18 getActualRetentionDuration – Specify duration is in milliseconds
- 5.3.3.19 getActualRetentionEnabled – Add PolicyMismatchException
- 5.3.3.20 getActualAutoDelete – Add PolicyMismatchException
- 5.3.3.21 getActualShred – Add PolicyMismatchException
- 5.3.3.25 submitJob – Add ObjectInUseException
- 5.3.5.6-8 XStream.read – Change end of stream behavior to be consistent with C API and Java.io package
- 5.3.5.8 read – Remove XStreamException from the throws.

- 5.3.6.3-9 Methods to get XAsync results throw AsyncHaltedException if the asynchronous operation was programmatically halted.
- 5.3.9.4 close – Fix method signature and remove erroneous bullet list item.
- 5.3.11 Clarification of AsyncHaltedException use.
- 5.3.11 Add PolicyMismatchException for use in XSet GetActual calls.
- 5.3.11 Remove EndOfStreamException – not consistent with C API and Java.io package.
- 5.4.3 Table 10 to be updated to match current access granules.
- 5.4.3 Table 10 – Add missing XAM_XSYSTEM_INITIALIZING property, remove duplicate XAM_XSYSTEM_ACCESS_POLICY_LIST
- 5.4.3 Table 11 – Fix SHRED constants in XSet
- 5.4.3 Table 11 – Add missing XAM_ACCESS_POLICY.
- 5.4.3 Table 11 – Remove non-existant modification time
- 5.4.3 Table 11 – Correct description of XAM_RETENTION_EVENT_DURATION
- 5.3.4.9 – 5.3.4.20 Add FieldDoesNotExistException to several method descriptions.
- 5.3.5.1 tell – Add XStreamException
- 5.3.5.13 asyncWrite – Fix exception description
- A.1 XAMLibrary.java – Add new logging property names.
- A.2 XSystem.java – Add authorization granule constants
- A.2 XSystem.java – Add constants that support multi-round authentication
- A.2 XSystem.java – Add missing XAM_XSYSTEM_INITIALIZING property.
- A.2 XSystem.java – Make connect method public
- A.2 XSystem.java – Add createXUID methods
- A.3 XSet.java – Correct field name typo
- A.3 XSet.java – Add missing XAM_ACCESS_POLICY property
- A.3 XSet.java – Add missing exception to submitJob
- A.3 FieldContainer.java – Remove InvalidArgumentException from some methods.
- A.3 FieldContainer.java – Add FieldDoesNotExistException to some methods.
- A.3 XStream.java – Add constant for end of file return value. (EOF)
- A.3 XUID.java – Add min length and max length constants.
- A.6 XAsync.java – Add exceptions to methods
- A.9 Xiterator.java – Refine the extends clause to be more compatible with modern Java.

- C.3 – Remove EndOfStreamException.

4.4.3 VIM Initialization - Correct Figure 6

In Figure 6, the first and third descriptions of what the XAM Library (blocks on the right side of the figure) are incorrect:

- The first description is currently "XAM Library createsProperty ".xsystem.initialization" to TRUE". Change it to "XAM Library creates ".xsystem.initializing" property and sets it to TRUE. Also change the label on the arrow to the left of this description block from "setProperty(Boolean)" to "createProperty(Boolean)"
- The third description is currently "XAM Library createsProperty ".xsystem.initialization" to FALSE". Change it to "XAM Library deletes ".xsystem.initializing".

- Change the send label “setProperty(Boolean)” (just prior to “connect()”) to be “deleteField()”

Note that the property names are corrected from ".xsystem.initialization" to ".xsystem.initializing" as part of this.

4.5.3 Example code - Correction

The code for writing the query into the XSet is not correct. The example is written adding the query string into the XSet as a property field, which is not allowed by the architectural specification. Replace these code lines:

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

With these corrected code lines:

```
XSet query;
query = sys.createXSet( XSet.MODE_RESTRICTED );
query.createProperty( XSet.XAM_JOB_COMMAND, false, XSet.XAM_JOB_QUERY
);
XStream qStream = query.createStream( XSet.XAM_JOB_QUERY_COMMAND,
false,
                                      "text/plain;charset=utf-8" );
String qString = "select \".xset.xuid\" where \"com.example.name\"" +
                 " = \'John Smith\'" ;
qStream.write( qString.getBytes("UTF-8"), 0, qString.length() );
qStream.close();
```

The “openXSet” method does not have the correct arguments. Replace these code lines:

```
while( (bytesRead = results.read(rawXUID)) >= 0 )
{
    XUID localXUID = new XUIDImpl(rawXUID);
    XSet data = sys.openXSet( XSet.MODE_READ_ONLY );
    // Process results
    data.close();
}
```

With these corrected code lines:

```
while( (bytesRead = results.read(rawXUID)) >= 0 )
{
    XUID localXUID = new XUIDImpl(rawXUID);
    XSet data = sys.openXSet( localXUID, XSet.MODE_READ_ONLY );
    // Process results
    data.close();
}
```

5.3.2.1 XSystem connect - Make method public

Replace the description for the XSystem connect method:

void connect(String xri)

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

With the following:

public void connect(String xri)

The connect method is called by the VIM after the XSystem has been created and initialized. This connect method operates identically to the specified method for XAMLibrary.connect() but shall not be not called by the application. This method shall be called only once by the XAM Library, and shall throw an exception if called more than once on the same XSystem instance.

5.3.2.2 Authenticate method - Allow multiple rounds of authentication exchange

Replace the text:

public byte[] authenticate(byte[] buffer)

This method will allow an application to authenticate an XSystem instance. It provides a generic interface to exchange data as part of the authentication process. The application must check the XSystem properties, prefixed constant value XAM_XSYSTEM_SASL_LIST, to determine which patterns of authentication are available for use. After a pattern is selected, the appropriate sequence of data exchanges should be made (using this call) in order to authenticate. A failed authentication will make the XSystem instance unusable. The application cannot repeat failed authentications using the same XSystem instance.

With:

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

This method will allow an application to authenticate an XSystem instance. It provides a generic interface to exchange data as part of the authentication process. The application must check the XSystem properties, prefixed constant value XAM_XSYSTEM_SASL_LIST, to determine which patterns of authentication are available for use. After a pattern is selected, the appropriate sequence of data exchanges should be made (using this call) in order to authenticate.

NOTE: SASL negotiations can require more information from an application. This method does not provide an indication that more information is required. This method shall be *deprecated* in the java interfaces. Applications are encouraged to use the alternate authenticate method providing this information.

Insert a new method description after this one (it will have a new section number).

authenticate

public int authenticate(byte[] in_buffer, ByteArrayOutputStream response_buffer)

This method will allow an application to authenticate an XSystem instance. It provides a generic interface to exchange data as part of the authentication process. The application must check the XSystem properties, prefixed constant value XAM_XSYSTEM_SASL_LIST, to determine which patterns of authentication are available for use. After a pattern is selected, the appropriate sequence of data exchanges should be made (using this call) in order to authenticate.

- Parameters:
  - in_buffer - Data being passed to the authentication system. The XSystem will use buffer.length to determine the significant bytes to be used by the authentication process. The format of the buffer differs, depending on the authentication mechanism chosen. Application authors are encouraged to refer to the appropriate SASL specifications [IANA-SASL].
  - response_buffer – The XSystem's response to the authenticate call. The contents of the stream are not read, and shall be reset before any response in written into the Stream.

- Returns: A status indicating if the authentication is complete: XAM_SASL_COMPLETE or XAM_SASL_IN_PROCESS.

- Throws:
  - — AuthenticationException - The authentication process encountered an error.
  - — InvalidArgumentException - The buffer argument is null.
  - — XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.2.9 accessXSet - Correct and add mode bits

The description of accessXSet leaves out legal access mode bits and doesn't use the right constants. Replace this list of mode bits:

— mode - Bitset of modes (READ_OK, WRITE_OK, WRITE_SYS_OK, DELETE_OK, HOLD_OK, EVENT_OK)

With the following

- — mode - Bitset of modes
  - o ACCESS_READ_OK,
  - o ACCESS_WRITE_APPLICATION_OK,
  - o ACCESS_WRITE_SYSTEM_OK,
  - o ACCESS_DELETE_OK,
  - o ACCESS_HOLD_OK,
  - o ACCESS_RETENTION_EVENT_OK,
  - o ACCESS_JOB_OK,
  - o ACCESS_JOB_COMMMIT_OK;

5.3.2.16 createXUID – New method for XSystem

Add the following method description to the XSystem Interface (it will have a new section number).

createXUID

public XUID createXUID(String base64XUID)

This method allows an application to create a XUID from a base64 encoded XUID value. The use of this method isolates the application from XAM Library implementations and allows the reconfiguration of an application's XAM implementation without needing to recompile. Applications are encouraged to use this method for creating XUID objects.

- Parameters:
  - — base64XUID – The base64 encoded string version of the XUID value.
- Returns: A XUID object.
- Throws:
  - — AuthenticationException - The authentication process encountered an error.
  - — InvalidXUIDException - The base64 string was invalid, the decoded XUID failed the CRC check, or the XUID value was malformed.
  - — XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.2.17 createXUID – New method for XSystem

Add the following method description to the XSystem Interface (it will have a new section number).

createXUID

public XUID createXUID(byte binaryXUID[])

This method allows an application to create a XUID from a binary XUID byte array value. The use of this method isolates the application from XAM Library implementations and allows the reconfiguration of an application's XAM implementation without needing to recompile. Applications are encouraged to use this method for creating XUID objects.

- Parameters:
  - binaryXUID – The binary XUID value.
- Returns: A XUID object.
- Throws:
  - AuthenticationException - The authentication process encountered an error.
  - InvalidXUIDException - The XUID value was malformed or failed a CRC check..
  - XAMException - Another error exists with the underlying XAM Storage System.
- Thread Safety: This method is thread safe.
- Blocking: This method blocks until completion.

5.3.3.3 applyManagementPolicy – Remove incorrect statement about the default management policy.

Remove the sentence: "The default policy is the first string in the XSystem management policy list."

5.3.3.4 resetManagementFields - Correct field name in description

Removes all management fields from the XSet. This includes *.xset.minimum.retention.starttime*.

Change *.xset.minimum.retention.starttime* to the correct *.xset.base.retention.starttime .*

5.3.3.5 createRetention - Forbid "base" retention ID

In the method description:

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

Replace the last sentence with the following two sentences:

> An exception shall be thrown if the retentionID is “base”. The binding attribute of this field is set according to the parameter *binding.*

In the InvalidArgumentException description:

— InvalidArgumentException - The parameter *retentionID* is null or malformed.

Add "or the value “base” was supplied" to the end of the sentence.

Add the following exception to the “throws” section of this method's description:

— RetentionValueException – When the retentionID is “event” and binding is FALSE.

5.3.3.6 setRetentionEnabledFlag - Forbid "base" retention ID, add exception for retention violation

In the method description:

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

Add the following sentence before "If the field does not exist":

> An exception shall be thrown if the retentionID is “base”.

In the InvalidArgumentException description:

— InvalidArgumentException - The parameter *retentionID* is null or malformed.

Add "or the value “base” was supplied" to the end of the sentence.

Add the following to the “throws” section of the description:

> — RetentionValueException – A previously enabled retention is being disabled.

5.3.3.7 applyRetentionEnabledPolicy - Forbid "base" retention ID, add exception for retention violation

In the method description:

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

value is set to TRUE, it cannot be changed. It will have its binding attribute set according to the binding flag that is set by the application.

Add the following sentence before "If the field does not exist":

An exception shall be thrown if the retentionID is “base”.

In the InvalidArgumentException description:

— InvalidArgumentException - The parameter *retentionID* is null or malformed.

Add "or the value “base” was supplied" to the end of the sentence.

Add the following to the “throws” section of the description:

— RetentionValueException – A previously enabled retention is being disabled.

5.3.3.8 setRetentionDuration - Forbid "base" retention ID, add exception for retention violation

In the method description:

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

Add the following sentence before "If the field does not exist":

An exception shall be thrown if the retentionID is “base”.

In the InvalidArgumentException description:

— InvalidArgumentException - The parameter *retentionID* is null or malformed.

Add "or the value “base” was supplied" to the end of the sentence.

Add the following to the “throws” section of the description:

— RetentionValueException – The supplied duration value is less than the duration previously specified for this retention definition.

5.3.3.9 applyRetentionDurationPolicy - Forbid "base" retention ID

In the method description:

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

Add the following sentence before "If the field does not exist":

An exception shall be thrown if the retentionID is “base”.

In the InvalidArgumentException description:

— InvalidArgumentException - The parameter *retentionID* is null or malformed.

Add "or the value “base” was supplied" to the end of the sentence.

5.3.3.10 setRetentionStarttime - Forbid "base" retention ID

In the method description:

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

Add the following sentence before "If the field does not exist":

An exception shall be thrown if the retentionID is “base”.

In the InvalidArgumentException description:

— InvalidArgumentException - The parameter *retentionID* is null or malformed.

Add "or the value “base” was supplied" to the end of the sentence.

5.3.3.11 setBaseRetention – Correct binding behavior and field name

The last sentence in the first paragraph of the Description is:

These fields will have their binding attribute set according to the binding flag that is set by the application.

Replace that sentence with:

The *.xset.retention.base.duration* field will have its binding attribute set according to the binding flag that is set by the application. The *.xset.retention.list.base* is always a binding field.

In this NOTE:

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

Add "as a binding field" after "will be created" and change *.xset.xuidtime* to the correct *.xset.time.xuid* .

5.3.3.12 applyBaseRetentionPolicy – correct field name, remove inappropriate exceptions.

In this NOTE:

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

Add "as a binding field" after "will be created" and change *.xset.xuidtime* to the correct *.xset.time.xuid* .

Remove the exceptions *InvalidArgumentException and InvalidFieldNameException.* The errors typically generating this will be covered by PolicyNameException.

5.3.3.13 setAutoDelete – Use correct field name

If this XSet does not have auto delete set on it, this method will create a property field on the XSet with the name of *.xset.autodelete* with the type “application/vnd.snia.xam.boolean”.

Change *.xset.autodelete* to the correct *.xset.deletion.autodelete* .

5.3.3.14 applyAutoDeletePolicy – Use correct field name

> If this XSet does not have an auto delete policy applied to it, this method will create a property field on the specified XSet with the name of *.xset.autodelete.policy* with the type set to "application/vnd.snia.xam.string."

Change *.xset.autodelete.policy* to the correct *.xset.deletion.autodelete.policy* .

5.3.3.15 setShred – Use correct field name

> If this XSet does not have the shred property set, this method creates a property field on the XSet with the name *.xset.shred* with type "application/vnd.snia.xam.boolean".

Change *.xset.shred* to the correct *.xset.deletion.shred* .

5.3.3.16 applyShredPolicy – Use correct field name

> If this XSet does not have an auto shred policy applied to it, this method will create a property field on the specified XSet with the name of *.xset.shred.policy* set to type "application/vnd.snia.xam.string".

Change *.xset.shred.policy* to the correct *.xset.deletion.shred.policy* .

5.3.3.18 getActualRetentionDuration – Add PolicyMismatchException

Add the following exception description to the method's list of throwable exceptions:

– PolicyMismatchException - When an imported XSet's policy does not match the policy present in the XSystem.

–

5.3.3.18 getActualRetentionDuration – Add PolicyMismatchException

Replace the "returns" clause from:

> Returns: The effective event retention duration for this XSet.

to the following:

> Returns: The effective event retention duration, in milliseconds, for this XSet.

5.3.3.19 getActualRetentionEnabled – Add PolicyMismatchException

Add the following exception description to the method's list of throwable exceptions:

- PolicyMismatchException - When an imported XSet's policy does not match the policy present in the XSystem.

5.3.3.20 getActualAutoDelete – Add PolicyMismatchException

Add the following exception description to the method's list of throwable exceptions:

- PolicyMismatchException - When an imported XSet's policy does not match the policy present in the XSystem.

5.3.3.21 getActualShred – Add PolicyMismatchException

Add the following exception description to the method's list of throwable exceptions:

- PolicyMismatchException - When an imported XSet's policy does not match the policy present in the XSystem.

5.3.3.25 submitJob – Needs to throw the ObjectInUseException
Add the following exception description to the method's list of throwable exceptions:
- ObjectInUseException – The XSet has open import/export streams.

5.3.4.9 setProperty – xam_boolean – Add FieldDoesNotExistException
Add the following exception to the *throws* section of the method description:

- FieldDoesNotExistException - The specified field does not exist.

5.3.4.10 setProperty – xam_datetime – Add FieldDoesNotExistException
Add the following exception to the *throws* section of the method description:

- FieldDoesNotExistException - The specified field does not exist.

5.3.4.11 setProperty – xam_double – Add FieldDoesNotExistException
Add the following exception to the *throws* section of the method description:

- FieldDoesNotExistException - The specified field does not exist.

5.3.4.12 setProperty – xam_int – Add FieldDoesNotExistException
Add the following exception to the *throws* section of the method description:

- FieldDoesNotExistException - The specified field does not exist.

5.3.4.13 setProperty – xam_string – Add FieldDoesNotExistException
Add the following exception to the *throws* section of the method description:

- FieldDoesNotExistException - The specified field does not exist.

5.3.4.14 setProperty – xam_xuid – Add FieldDoesNotExistException
Add the following exception to the *throws* section of the method description:

- FieldDoesNotExistException - The specified field does not exist.

5.3.4.15 getBoolean – Add FieldDoesNotExistException
Add the following exception to the *throws* section of the method description:

- FieldDoesNotExistException - The specified field does not exist.

5.3.4.16 getDatetime – Add FieldDoesNotExistException
Add the following exception to the *throws* section of the method description:

- FieldDoesNotExistException - The specified field does not exist.

5.3.4.17 getDouble – Add FieldDoesNotExistException
Add the following exception to the *throws* section of the method description:

- FieldDoesNotExistException - The specified field does not exist.

5.3.4.18 getLong – Add FieldDoesNotExistException
Add the following exception to the *throws* section of the method description:

- FieldDoesNotExistException - The specified field does not exist.

5.3.4.19 getString – Add FieldDoesNotExistException

Add the following exception to the *throws* section of the method description:

- FieldDoesNotExistException - The specified field does not exist.

5.3.4.20 getXUID – Add FieldDoesNotExistException

Add the following exception to the *throws* section of the method description:

- FieldDoesNotExistException - The specified field does not exist.

5.3.5.6 XStream.read method

Replace the text:

• Returns: The actual number of bytes read.

With:

• Returns: The actual number of bytes read. This value will be less than or equal to the inBufferLength. When there is no more data to be read, a value of -1 will be set.

5.3.5.7 XStream.read method

Replace the text:

• Returns: The actual number of bytes read.

With:

• Returns: The actual number of bytes read. This value will be less than or equal to the inBufferLength. When there is no more data to be read, a value of -1 will be set.

5.3.5.8 XStream.read method

Replace the text:

• Returns: The actual number of bytes read.

With:

• Returns: The actual number of bytes read. This value will be less than or equal to the inBufferLength. When there is no more data to be read, a value of -1 will be set.

5.3.5.8 read – Remove XStreamException from the throws

Remove the XStreamException from the throws description of the method.

5.3.5.13 asyncWrite – Fix exception description

Fix the text of the XStreamException to read as follows (change “write only” to “read only”)

— XStreamException - The stream is open in the wrong mode (read only).

5.3.6.1 halt

The description is:

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

Add this sentence to the end of the description "If this method causes the XAsync instance to halt before it is complete, any get method invoked on that XAsync will throw an AsyncHaltedException."

5.3.6.3 getXOPID

Insert the following exception before the XAMException in the “throws” section of the description:

- AsyncHaltedException – This operation was programmatically halted and results are not available.

5.3.6.4 getStatus

Insert the following exception before the XAMException in the “throws” section of the description:

- AsyncHaltedException – This operation was programmatically halted and results are not available.

5.3.6.5 getXSet

Insert the following exception before the XAMException in the “throws” section of the description:

- AsyncHaltedException – This operation was programmatically halted and results are not available.

5.3.6.6 getXStream

Insert the following exception before the XAMException in the “throws” section of the description:

- AsyncHaltedException – This operation was programmatically halted and results are not available.

5.3.6.7 getXUID

Insert the following exception before the XAMException in the “throws” section of the description:

- AsyncHaltedException – This operation was programmatically halted and results are not available.

5.3.6.8 getBytesWritten

Insert the following exception before the XAMException in the “throws” section of the description:

— AsyncHaltedException – This operation was programmatically halted and results are not available.

5.3.6.9 getBytesRead

Insert the following exception before the XAMException in the “throws” section of the description:

— AsyncHaltedException – This operation was programmatically halted and results are not available.

5.3.9.4 close

Replace this method signature:

```
public boolean close()
```

With this signature:

```
public void close()
```

Delete the following bulleted list item in section 5.3.9.4

- Returns: TRUE if the two values are the same.

5.3.11 AsyncHalted Exception clarification

In table 3 on p.91, Change the descriptive text of the AsyncHaltedException from:

> an asynchronous operation has been programmatically halted. All XAsynch methods, except getStatus, shall throw this exception when an XAsync has been halted.

To:

> an asynchronous operation has been programmatically halted. All XAsync get methods shall throw this exception when an XAsync has been halted.

Note that a misspelling of "XAsynch" is corrected to "XAsync" as part of this change.

5.3.11 PolicyMismatchException added to table 6

In table 6 on page 93, insert the following new exception before the PolicyNameException

PolicyMismatchException	an imported XSet's policy does not match the policy present in the XSystem.

5.3.11 EndOfStreamException removal

The EndOfStreamException is not consistent with the XAM C API nor is it consistent with the standard Java.io package. Remove EndOfStreamException from Table 7 on p.94.

Section 5.4.2: Table 10 – Replace authorization granule items with correct descriptions

Replace these rows in table 10:

public static final READ_OK	Read access is OK. Used in accessXSet.
public static final WRITE_OK	Write access is OK. Used in accessXSet.
public static final WRITE_SYS_OK	Write access to system fields is OK. Used in accessXSet.
public static final DELETE_OK	Delete access is OK. Used in accessXSet.
public static final HOLD_OK	Hold/release access is OK. Used in accessXSet.

public static final EVENT_OK	Event retention access is OK. Used in accessXSet

With these rows:

public static final ACCESS_READ_OK	Read access is OK. Used in accessXSet.
public static final ACCESS_WRITE_APPLICATION_OK	Write access to application content is OK. Used in accessXSet.
public static final ACCESS_WRITE_SYSTEM_OK	Write access to system content is OK. Used in accessXSet.
public static final ACCESS_CREATE_OK	New XSets may be created. Used in accessXSet.
public static final ACCESS_DELETE_OK	XSets may be deleted. Used in accessXSet.
public static final ACCESS_HOLD_OK	Hold and Release methods may be used. Used in accessXSet.
public static final ACCESS_RETENTION_EVENT_OK	Setting of retention start times and creating new retention identifiers is allowed. Used in accessXSet.
public static final ACCESS_JOB_OK	Running of Jobs, including query is allowed. Used in accessXSet.
public static final ACCESS_JOB_COMMIT_OK	Commit of jobs and results is allowed. Used in accessXSet.

Section 5.4.2 Table 10 – ADD XAM_XSYSTEM_INITIALIZING

There are two rows for XAM_XSYSTEM_ACCESS_POLICY_LIST in Table 10 (p.96 and p.97), replace the first one of these rows with this row:

- Field Name: public static final XAM_XSYSTEM_INITIALIZING
- Description: The XSystem instance is in the process of initializing.

5.4.3 Table 11 – Change XAM_SHRED to XAM_DELETION_SHRED

Change the constant name “XAM_SHRED” to “XAM_DELETION_SHRED” and change “XAM_SHRED_POLICY” to “XAM_DELETION_SHRED_POLICY”

5.4.3 Table 11 – Add missing XAM_ACCESS_POLICY

Add the following row to the end of Table 11

- Field Name: public static final XAM_ACCESS_POLICY
- Description: The xam_string property containing the name of the access control policy, if applied, for this XSet

5.4.3 Table 11 – remove modification time

In Table 11 – XSet Constants, remove the row that contains XAM_TIME_MODIFICATION. The XAM standard does not define a modification time.

5.4.3 Table 11 – Correct description of XAM_RETENTION_EVENT_DURATION

In Table 11 – XSet Constants, the Description of XAM_RETENTION_EVENT_DURATION refers to the "**base** retention ID". Change this to the "**event** retention ID".

A.1 XAMLibrary.java – Add new logging constants

Add the following constant values, after XAM_LOG_PATH and before XAM_API_LEVEL

```
public static final String XAM_LOG_MAX_ROLLOVERS = ".xam.log.max.rollovers";
```

```
public static final String XAM_LOG_MAX_SIZE      = ".xam.log.max.size";

public static final String XAM_LOG_APPEND        = ".xam.log.append";
```

A.2 XSystem.java – Add access constants

The ACCESS mode bits described in the XSystem interface do not include the Long indicators for the numeric values. Replace the following constant:

```
public interface XSystem extends FieldContainer
{
// The following bit fields are defined for accessXSet().
public final static long ACCESS_READ_OK = 0x80000000;
public final static long ACCESS_WRITE_APPLICATION_OK = 0x40000000;
public final static long ACCESS_WRITE_SYSTEM_OK = 0x20000000;
public final static long ACCESS_CREATE_OK = 0x10000000;
public final static long ACCESS_DELETE_OK = 0x08000000;
public final static long ACCESS_HOLD_OK = 0x04000000;
public final static long ACCESS_RETENTION_EVENT_OK = 0x02000000;
public final static long ACCESS_JOB_OK = 0x01000000;
public final static long ACCESS_JOB_COMMIT_OK = 0x00800000;
```

With these values

```
public interface XSystem extends FieldContainer
{
// The following bit fields are defined for accessXSet().
public final static long ACCESS_READ_OK = 0x80000000L;
public final static long ACCESS_WRITE_APPLICATION_OK = 0x40000000L;
public final static long ACCESS_WRITE_SYSTEM_OK = 0x20000000L;
public final static long ACCESS_CREATE_OK = 0x10000000L;
public final static long ACCESS_DELETE_OK = 0x08000000L;
public final static long ACCESS_HOLD_OK = 0x04000000L;
public final static long ACCESS_RETENTION_EVENT_OK = 0x02000000L;
public final static long ACCESS_JOB_OK = 0x01000000L;
public final static long ACCESS_JOB_COMMIT_OK = 0x00800000L
```

A.2 XSystem.java – Add constants allowing multi-round authentication

Add these constants after the ACCESS constants, and prior to the XSystem Properties:

```
final int XAM_SASL_COMPLETE   = 0;
final int XAM_SASL_IN_PROCESS = 1;
```

A.2 XSystem.java - add XAM_XSYSTEM_INITIALIZING

Immediately after the " // == XSystem Properties == " comment, add the following

```
public final static String XAM_XSYSTEM_INITIALIZING =
      ".xsystem.initializing";
```

A.2 XSystem.java – Make connect method public

Replace the XSystem.java connect method:

```
void connect(String xri) throws XAMException;
```

With this text:

```
public void connect(String xri) throws XAMException;
```

A.2 XSystem.java – Add createXUID methods

Add the following method signatures after the "abandon()" method signature.

```
public XUID createXUID( String inBase64XUID )
      throws AuthenticationExpiredException, InvalidXUIDException, XAMException;

public XUID createXUID( byte inBinaryXUID[] )
      throws AuthenticationExpiredException, InvalidXUIDException, XAMException;
```

A.3 XSet.java – Correct field name typo

On p.109, change ".xet.management.policy" to ".xset.management.policy" and make the quotes vertical.

A.3 XSet.java – Add missing XAM_ACCESS_POLICY property

Insert the following missing property name into the XSet Interface, after the existing XAM_MANAGEMENT_POLICY.

```
public final static String XAM_ACCESS_POLICY = ".xset.access.policy";
```

A.3 XStream.java – Add a constant for End of File value.

Insert the following constant value into the XStream Interface, after the definition of MODE_WRITE_APPEND:

```
    /** End of File value returned from a read() method when the end of
XStream has been reached. */
    public final static long EOF = -1;
```

A.3 XSet.java – Add ObjectInUseException to submitJob Method.

Replace the *submitJob* method signature with the following:

```
    public void submitJob()
       throws AuthenticationExpiredException, AuthorizationException,
              JobCommandException, JobUnsupportedException,
              ObjectInUseException, XSetAbandonException,
              XSetCorruptException, XAMException;
```

A.3 FieldContainer.java – Remove InvalidArgumentException from some methods

Remove the *InvalidArgumentException* from the following method signatures:

```
      public void createProperty(String fieldName, boolean binding, boolean value)
      throws AuthenticationExpiredException, InvalidFieldNameException,
             FieldExistsException,
```

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

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

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

A.3 Add FieldDoesNotExistException to some methods

Add the *FieldDoesNotExistException* to the following methods:

```
public void setProperty(String fieldName, boolean value)
public void setProperty(String fieldName, Calendar value)
public void setProperty(String fieldName, double value)
public void setProperty(String fieldName, long value)
public void setProperty(String fieldName, String value)
public void setProperty(String fieldName, XUID value)
public boolean getBoolean(String fieldName)
public Calendar getDateTime(String fieldName)
public double getDouble(String fieldName)
public long getLong(String fieldName)
public String getString(String fieldName)
public XUID getXUID(String fieldname)
```

A.3 XUID.java – Add minimum and Maximum length constants to the interface.

Insert the following constant values into the XUID interface, before the first method “toBytes”.

```
/** A XUID may be at most 80 bytes long. */
final public int MAX_LENGTH = 80;

/** A XUID must be at least 9 bytes long. */
final public int MIN_LENGTH = 9;
```

A.6 Add exceptions to XAsync methods getBytesRead() getBytesWritten;

Change the method signatures of the two specified methods to:

```
public long getBytesRead()
        throws AuthenticationExpiredException, AsyncHaltedException, XAMException;

public long getBytesWritten();
        throws AuthenticationExpiredException, AsyncHaltedException, XAMException;
```

A.6 Add AsyncHaltedException to all get methods.

Add AsyncHaltedException to the throws clause of the methods:

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

```
public void getStatus()
       throws AuthenticationExpiredException, AsyncHaltedException,
XAMException;

public XSet getXSet()
       throws AuthenticationExpiredException, AsyncHaltedException,
XAMException;

public XStream getXStream()
       throws AuthenticationExpiredException, AsyncHaltedException,
XAMException;

public XUID getXUID()
       throws AuthenticationExpiredException, AsyncHaltedException,
XAMException;

public long getBytesRead()
       throws AuthenticationExpiredException, AsyncHaltedException,
XAMException;

public long getBytesWritten()
       throws AuthenticationExpiredException, AsyncHaltedException,
XAMException;
```

A.9 Xiterator.java – Add a minor refinement to the extends clause.

In the "public interface" line of the declaration, replace this line:

```
 */
public interface XIterator extends Iterator
{
```

With

```
 */
public interface XIterator extends Iterator<String>
{
```

C.3 EndOfStreamException removal

The EndOfStreamException is not consistent with the XAM C API nor is it consistent with the standard Java.io package. Remove this line from Annex C.3:

```
import org.snia.xam.EndOfStreamException;
```