XenData Workflow Interface Programmers Guide

Version	Date	Author	Changes
November 19, 2010 build	11/9/2010	Mark Broadbent	N/A
April 12, 2011 build	5/3/2011	Robert Tantzen	Update: predefined extension support, Barcode responses and minor corrections
August 10, 2011 build 884	8/10/2011	Robert Tantzen	Update: ONLINE/OFFLINE status support and minor corrections
Build 1018 Version 1.01	11/15/2011	Mark Broadbent	Extensively revised to incorporate many new commands and options.
	04/08/2014	Ben Kelly	Made corrections to the SQL clause in SearchObjects and to priorities
Build 2062	08/04/2014	Mark Broadbent	Added support for partial file restores. Minor corrections.
Build 2125	01/16/2015	Mark Broadbent	Added support for FILE and CIFS transfers
Build 2343	11/02/2015	Mark Broadbent	Added support for end-to-end logical block protection. Implemented per-file or per-asset partial restore.

Concepts

Asset

An asset is a collection of one or more physical files which when combined make up a single entity from the archive's perspective. The individual files may be located in different directories and they may contain video frames, audio tracks, subtitle information etc. and any other data or metadata that is part of the whole asset. Within this API an asset is considered to be a single entity which is normally transferred to or from the archive as a whole, in a single atomic operation. Assets are identified either by an asset name, or by a globally unique identifier (GUID) provided as part of the request. An asset name looks like a file name, optionally omitting the extension such as .mov or .mxf which is specified in the MediaServers.xml configuration file. For assets that comprise a single file, the asset name is the file name and for assets that are described by container files, the asset name is the name of the container file. An asset GUID can have any format; the only requirement is that GUIDs are indeed unique.

Job

A Job is a unit of work, such as archiving an asset. Every job has a GUID which is represented as a string that looks like "4B45BA54-4CD854C1-00-00000001" and which is used to track the progress of the job and retrieve success or failure conditions on completion.

Request

A request is an XML string that is sent to the API to initiate a job. A typical request would look like this:

```
<Request>
  <Archive>
    <Priority>100</Priority>
    <AssetName>_AAOI0019</AssetName>
    <ServerID>Server1</ServerID>
    <ServerPath></ServerPath>
    <ArchivePath>FirstGroup</ArchivePath>
  </Archive>
</Request>
```

For certain operations the system's response to a well-formed request is to return a JobID that can be used to track the progress of the job:

```
<Response>
  <JobID>4B45BA54-4CD92C09-00-00000001</JobID>
</Response>
```

Directory Structure of the Archive

The archive may be configured either as a directory structure just like any conventional file system, or as an object store addressed by asset GUIDs. When it is organised into a directory structure, assets can be archived anywhere in the directory tree. There is no need to create directories before using them; the action of specifying an "ArchivePath" causes the directory to be created.

When the system is configured as an object store the "ArchivePath" is used to partition the archive into independent sections. An object's identity is the combination of the ArchivePath and the asset GUID. In this mode of operation the ArchivePath must be specified and must not contain any

directory separators. Asset GUIDs must be unique within a section of the archive but may be reused between sections.

End-to-end Logical Block Protection

End-to-end Logical Block Protection is a technology that is implemented in a variety of mass storage devices including hard disk drives and tape drives. It allows validation of the data stored on the physical storage medium (data cartridge) by the insertion of a checksum for every data block that is written to the mass storage drive. The drive validates the checksum as a part of its read-after-write check on the integrity of the recorded data. The combination of the XenData XML Workflow Interface and XenData Archive Series software allows the generation of end-to-end logical block protection information from a file checksum that is passed in through the XML interface. Validation of this checksum by the drive provides a very high degree of confidence that the data written to the data cartridge is an exact copy of the original source data.

The following checksum algorithms are supported: MD2, MD4, MD5, SHA/SHA1, SHA256, SHA384, and SHA512.

The XML API

Overview

The XML API is implemented by a server that listens for requests on port 3466 (the Workflow port). This port is bidirectional and responses are sent back using the same connection.

Sending a request

A request is an XML string with a root element named "Request". This contains a sub element whose name is the name of the command that is being requested. Beneath the command element are the parameters to the command which can be either attributes of the command element or sub elements in their own right. The following commands are equivalent:

```
<Request>
  <GetStatus>
    <JobID>4B45BA54-4CD93C2A-00-00000001</JobID>
  </GetStatus>
</Request>

<Request>
  <GetStatus JobID="4B45BA54-4CD93C2A-00-00000001"/>
</Request>
```

Response to a request

The response to a request depends on the type of request. A GetStatus request returns information about the progress of the request whose JobID was given. Certain requests, such as the ObjectExists request immediately return information about the requested object. Other requests that require more significant processing (such as data transfer requests) return a response that contains the JobID of the job that was created to process the request. The descriptions of individual commands describe the response returned.

JobID response

The response to a request that returns a JobID is as follows:

```
<Response>
  <JobID>4B45BA54-4CD92C09-00-00000001</JobID>
</Response>
```

Fault response

If the request was not well formed, for example if the XML did not parse correctly, if the command tag did not identify a supported command or if required parameters were missing then the response will be a "Fault" indication, formed as follows:

```
<Fault>
  <Code>3222070611</Code>
  <Message>
    XML Parse Error: The following tags were not closed: Request.
    (line 0 char 0)
  </Message>
  <Class>
    <XMLParse/>
  </Class>
  <Module>xmldocument.cpp</Module>
  <Line>526</Line>
</Fault>
```

The Fault indication contains an error code, text message and additional information that can help to determine what went wrong if this is not obvious from the message.

JobID XML schema

Name	Type	Min Occurs	Max Occurs	Description
JobID	xs:string			A string representation of the Job ID. This can be used as a parameter to the GetStatus command to track progress of the command.

Fault XML schema

Name	Type	Min Occurs	Max Occurs	Description
Code	xs:int			The error code that would be returned by the operating system if it was trying to report this error.
Event	xs:int	0		For certain error messages, an event code logged by the Workflow server in the Windows event log.
Message	xs:string			Plain text of the error message, which can be reported to a user.
JobID	xs:string	0		JobID of the job that has failed, if the Fault is returned as a response to a GetStatus request.
Class	xs:string			Internal class of the error. This is primarily to allow serialisation of errors and can be ignored by API clients.
Module	xs:string			Source code file within the Workflow server where the error was detected. Only used for advanced error analysis.
Line	xs:int			Source code line within the Workflow server where the error was detected.

GetStatus response

The format of the GetStatus response contains a fixed part that indicates the progress of the job (Complete, Working or Waiting) and a variable part that depends on the request type: see descriptions of individual commands for more details. A typical response for a job that has is currently executing looks like this:

```
<Response>
  <JobID>4B45BA54-4CD93C2A-00-00000001</JobID>
  <State>Working</State>
  <TotalBytes>2787709191</TotalBytes>
  <BytesProcessed>109254436</BytesProcessed>
  <TotalFiles>3</TotalFiles>
  <FilesProcessed>0</FilesProcessed>
  <PercentProcessed>4</PercentProcessed>
</Response>
```

GetStatus State element

The State element can have one of three values: “Waiting”, “Working “or “Complete”. “Waiting” means that the command is in the queue but not currently executing, “Working” means that it is currently executing on the server and “Complete” means that it has finished executing. A typical response for a job that has competed successfully looks like this:

```
<Response>
  <JobID>4B45BA54-4CD93C2A-00-00000001</JobID>
  <State>Complete</State>
  <Phase>Receiving Data/Phase>
  <TotalBytes>2787709191</TotalBytes>
  <BytesProcessed>2787709191</BytesProcessed>
  <TotalFiles>3</TotalFiles>
  <FilesProcessed>3</FilesProcessed>
  <PercentProcessed>100</PercentProcessed>
</Response>
```

The GetStatus response for a job that has failed is very similar to the Fault response for request submission, but with the addition of the JobID of the job that failed:

```
<Fault>
  <Code>2</Code>
  <Message>
    The system cannot find the file specified.
  </Message>
  <JobID>4B45BA54-4CD96896-00-00000001</JobID>
  <Class>
    <DWORD/>
  </Class>
  <Module>omneonmedia.cpp</Module>
  <Line>101</Line>
</Fault>
```

Commands

Archive Command

The Archive command looks for one or more assets at specified paths on a remote server. If an asset is found, the command queries the server to find all the files in the asset, and then copies them to the archive. The files comprising the asset are left in place on the remote server.

A single asset may be specified by server name, path and file name, or one or more assets may be specified by means of URLs to paths on one or more local or remote servers. If the URL form is used, it is possible to specify different URLs for FTP and CIFS access to the same file, and it is possible to specify that just the container file should be archived, not including any referenced essence files.

An asset GUID and search metadata may also be provided, which can be used in the future to find the asset.

XML Schema

Name	Type	Min Occurs	Max Occurs	Description
Priority	xs:int			Priority of this request relative to other requests in the system. Requests with low numbers are processed before requests with high numbers; i.e. Priority 1 is the highest priority.
ArchivePath	xs:string	0		Path to the asset on the archive.
ArchiveFileName	xs:string	0		Provides an optional alternative file name for the asset on the archive.
ServerID	xs:string	0		The name of the remote server that contains the file. This name refers to an entry in the MediaServers.xml configuration file.
AssetName	xs:string	0		The name of the file that contains the asset. This may be the only file in the asset, or it may be a container that references other files. If the file is a container then the files it references will also be archived.
ServerPath	xs:string	0		Path to the asset on the remote server, relative to the

				predefined root of media on the server.
ServerFileName	xs:string	0		Provides an optional alternative file name for the asset on the remote server.
ArchiveGUID	xs:string	0		GUID to be used instead of the file name to reference the file.
SearchMetadata		0	unbounded	Collection of column name/value pairs.
Asset	RemoteObject	0	unbounded	Description of a file on a remote server that contains an asset. This may be the only file in the asset or it may be a container that references other files on the server. If the file is a container then the files it references will also be archived.
File	RemoteObject	0	unbounded	Description of a file on a remote server that contains an asset. This may be the only file in the asset or it may be a container that references other files on the server. If the file is a container then the files it references will not be archived.
FileHash	HashValue	0	unbounded	An optional file hash (checksum) value that can be used to verify that the data stored in the archive is an exact copy of the original source data.

XML Schema for RemoteObject

Name	Type	Min Occurs	Max Occurs	Description
FileURL	xs:anyURI	1	unbounded	URL pointing to the file on the remote server.
ArchivePath	xs:string	0		Path to the asset on the archive.
ArchiveGUID	xs:string	0		GUID to be used instead of the file name for access to the file.
SearchMetadata	list of Column	0	unbounded	Collection of column name/value pairs.
FileHash	HashValue	0		An optional file hash (checksum) value that can be used to verify that the data stored in the

				archive is an exact copy of the original source data.

XML Schema for metadata Column

Name	Type	Min Occurs	Max Occurs	Description
Name	xs:string			Name of the column.
Select	xs:string			Value for column data.

XML Schema for HashValue

Name	Type	Min Occurs	Max Occurs	Description
Algorithm	xs:string			One of the supported hash algorithms.
Hash	xs:string			The hash value.
FileName	xs:string	0		In the case of an asset that comprises multiple files, a separate hash value is specified for each individual file. The FileName specifies the file that the hash value relates to.

XML Schema for "success" response to GetStatus command

Name	Type	Min Occurs	Max Occurs	Description
JobID	xs:string			A string representation of the Job ID.
State	xs:string			One of "Waiting", "Working", or "Complete".
Phase	xs:string	0		One of "Receiving Data", "Writing Data" or "Generating Partial Restore Index".
TotalBytes	xs:long			The total number of bytes discovered in the asset.
BytesProcessed	xs:long			The number of bytes that have been transferred from the remote server.
TotalFiles	xs:int			The total number of files discovered in the asset.
FilesProcessed	xs:int			The number of files that have been transferred from the remote server.
TotalFrames	xs:int	0		Where available, the total number of frames in the asset.
PercentProcessed	xs:int			The percentage of the total data transfer that has completed.

Example 1 (server path form)

```
<Request>
  <Archive>
    <Priority>100</Priority>
    <AssetName>_AAOI0019</AssetName>
    <ServerID>Server1</ServerID>
    <ServerPath></ServerPath>
    <ArchivePath>FirstGroup</ArchivePath>
```

```
  </Archive>
</Request>
```

This request looks in the MediaServers.xml file for a server called “Server1”. If it finds the server then it looks for a file called “_AAAOI0019.mov” in the root directory (because the ServerPath contains an empty string and the MediaServers.xml file specifies mov extensions). If it finds the file then it examines it to determine the list of files that make up the asset. Once that has been done, the request transfers the files to the archive, in a directory called “FirstGroup”.

Example 2 (URL form with FileHash)

```
<Request>
	<Archive>
		<Priority>100</Priority>
		<ArchivePath>FirstGroup</ArchivePath>
		<File>
			<ArchiveGUID>1111-1111-1111-1111</ArchiveGUID>
			<FileUrl>cifs://XenData001/fs/_AAOI0019.mov</FileUrl>
			<FileHash Algorithm="SHA1">
				<Hash>53B55A10D75C838C6607A62EF7DCFECE8C28FA94</Hash>
			</FileHash>
		</File>
	</Archive>
</Request>
```

This request uses a URL to specify a single file which is not treated as a container (because of the “File” tag). A GUID is assigned for future reference to the file and a FileHash was provided to validate the data. Because a GUID was assigned to the file, its name in the archive will be “FirstGroup/1111-1111-1111-1111” and the GUID must be specified when the file is retrieved from the archive.

Example 3 (more complex URL form)

```
<Request>
	<Archive>
		<ServerID>Server</ServerID>
		<Priority>100</Priority>
		<ArchivePath>FirstGroup</ArchivePath>
		<Asset>
			<ArchiveGUID>1111-1111-1111-1111</ArchiveGUID>
			<FileUrl>ftp://user:pw@XenData001/fs/_AAOI0019.mov</FileUrl>
			<FileUrl>cifs://XenData001/fs/_AAOI0019.mov</FileUrl>
			<SearchMetadata>
				<Column Name="displayName">
					<Select>Abc.mxf</Select>
				</Column>
				<Column Name="category">
					<Select>drama</Select>
				</Column>
			</SearchMetadata>
		</Asset>
		<Asset>
			<ArchiveGUID>1111-1111-1111-2222</ArchiveGUID>
			<FileUrl>ftp://user:pw@XenData001/fs/_AAOI0020.mov</FileUrl>
			<FileUrl>cifs://XenData001/fs/_AAOI0020.mov</FileUrl>
		</Asset>
	</Archive>
</Request>
```

This request archives two assets, treating them both as containers (because of the “Asset” tag) and therefore also archiving files referenced by them. GUIDs are assigned for future reference to both

assets and search metadata is associated with one of them. Because GUIDs were assigned to the assets, their names in the archive will be “FirstGroup/1111-1111-1111-1111” and “FirstGroup/1111-1111-1111-2222” and the GUIDs must be specified when the files are to be retrieved from the archive.

Polling with GetStatus shortly after either of these commands starts executing might return the following:

```
<Response>
  <JobID>4D8CE12D-4EC4F822-00-00000006</JobID>
  <State>Working</State>
  <Phase>Receiving Data/Phase>
  <TotalBytes>2787709191</TotalBytes>
  <BytesProcessed>109254436</BytesProcessed>
  <TotalFiles>3</TotalFiles>
  <FilesProcessed>0</FilesProcessed>
  <PercentProcessed>4</PercentProcessed>
</Response>
```

XML Schema for “failure” response

This is a standard “Fault” response described above under “Fault XML schema”.

Example

```
<Fault>
  <JobID>4D8CE12D-4EC4F822-00-00000006</JobID>
  <TotalBytes>0</TotalBytes>
  <BytesProcessed>0</BytesProcessed>
  <TotalFiles>0</TotalFiles>
  <FilesProcessed>0</FilesProcessed>
  <PercentProcessed>0</PercentProcessed>
  <Code>5</Code>
  <Event>3221231476</Event>
  <Message>File 'X:\FirstGroup\_AAOI0019.mov\_AAOI0019.mov' already exists in
the archive.  Overwriting it is not permitted</Message>
  <JobID>4D8CE12D-4EC4F822-00-00000006</JobID>
  <Class>
    <Message Message="3221231476"
Param1="X:\FirstGroup\_AAOI0019.mov\_AAOI0019.mov"/>
  </Class>
  <Module>Transaction.cpp</Module>
  <Line>230</Line>
</Fault>
```

Format of a URL

A URL specifies the transport protocol, server ID, path name and any required login credentials. The form of a URL is “protocol://userid:password@server/path/filename.ext”.

protocol is one of “file”, “ftp” or “cifs”
userid is a plain-text user name, or a string of the form domain\username
password is a plain-text password.

The user id and password are both optional; if the password is omitted then the colon separating the user id and password is also omitted. If both the user id and password are omitted then the ‘@’ sign before the server name is also omitted. The ‘file’ protocol does not require a server name because it refers to a file on the local machine, but if a server name is specified then it will be ignored. To access a file on another server, use the ‘ftp’ or ‘cifs’ protocols.

The following are valid URLs:

file://localhost/C:/Archive/BBBTrailer.mxf
file:///C:/Archive/BBBTrailer.mxf
cifs://XenData001/share/Archive/BBBTrailer.mxf
cifs://username:password@XenData001/share/Archive/BBBTrailer.mxf
cifs://domain\username:password@XenData001/share/Archive/BBBTrailer.mxf
ftp://username:password@XenData002/Archive/BBBTrailer.mxf

Move Command

The Move command moves an asset from a remote server to the archive. This operation is similar to the Archive command, with the exception that that when all the files that comprise the asset have been successfully copied to the archive, they are removed from the remote server. If any part of the request fails, no files are deleted on the remote media server.

XML Schema

The XML schema is the same as that for the Archive command.

XML Schema for "success" response to GetStatus command

The response is the same as that for the Archive command

Example

```
<Request>
  <Move>
    <Priority>100</Priority>
    <AssetName>_AAOI0019</AssetName>
    <ServerID>Server1</ServerID>
    <ServerPath></ServerPath>
    <ArchivePath>FirstGroup</ArchivePath>
  </Move>
</Request>
```

XML Schema for "failure" response

This is a standard "Fault" response described above under "Fault XML schema".

Example

```
<Fault>
      <JobID>4D8CE12D-4EC3AB80-00-00000001</JobID>
      <TotalBytes>0</TotalBytes>
      <BytesProcessed>0</BytesProcessed>
      <TotalFiles>0</TotalFiles>
      <FilesProcessed>0</FilesProcessed>
      <PercentProcessed>0</PercentProcessed>
      <Code>2</Code>
      <Event>3221231480</Event>
      <Message>Cannot archive 'fs/_AAOI0019.mov' because the file could not be
found on the media server</Message>
      <JobID>4D8CE12D-4EC3AB80-00-00000001</JobID>
      <Class>
            <Message Message="3221231480" Param1="fs/_AAOI0019.mov"/>
      </Class>
      <Module>omneonmedia.cpp</Module>
      <Line>293</Line>
</Fault>
```

Restore Command

The Restore command transfers one or more assets from the archive back to a remote media server, leaving then in place in the archive.

XML Schema

Name	Type	Min Occurs	Max Occurs	Description
Priority	xs:int			Priority of this request relative to other requests in the system. Requests with low numbers are processed before requests with high numbers; i.e. Priority 1 is the highest priority.
ArchivePath	xs:string	0		Path to the asset on the archive.
ArchiveFileName	xs:string	0		Provides an optional alternative file name for the asset on the archive.
ServerID	xs:string	0		The name of the remote server that contains the file. This name refers to an entry in the MediaServers.xml configuration file.
AssetName	xs:string	0		The name of the file that contains the asset. This may be the only file in the asset, or it may be a container that references other files. If the file is a container then the files it references will also be restored.
ServerPath	xs:string	0		Path to the asset on the remote server, relative to the predefined root of media on the server.
ServerFileName	xs:string	0		Provides an optional alternative file name for the asset on the remote server. This can be used, for example, to give a different name to a partially restored file.
ArchiveGUID	xs:string	0		GUID to be used to reference the file.
Asset	RemoteObject	0	unbounded	Description of a file that contains an asset. This may

				be the only file in the asset or it may be a container that references other files. If the file is a container then the files it references will also be restored.
File	RemoteObject	0	unbounded	The name of a file. If the file is a container then the files it references will not be restored.
Partial	PartialRequest	0	unbounded	Contains a list of in and out points for performing a partial restore. If this tag is present in this location the partial restore operation will be applied to every file in the request. The maximum number of supported in and out points depends on the partial restore engine being used.

XML Schema for RemoteObject

Name	Type	Min Occurs	Max Occurs	Description
FileURL	xs:anyURI	1	unbounded	URL pointing to the file on the remote server.
ArchivePath	xs:string	0		Path to the asset on the archive.
ArchiveGUID	xs:string	0		GUID to be used instead of the file name for access to the file.
Partial	PartialRequest	0	unbounded	Contains a list of in and out points for performing a partial restore. If this tag is present in this location the partial restore operation will be applied only to the current asset. The maximum number of supported in and out points depends on the partial restore engine being used.

XML Schema for PartialRequest

Name	Type	Min	Max	Description

		Occurs	Occurs	
TimeIn	xs:string			Start timecode formatted as HH:MM:SS:FF
TimeOut	xs:string			End timecode formatted as HH:MM:SS:FF

XML Schema for "success" response to GetStatus command

Name	Type	Min Occurs	Max Occurs	Description
JobID	xs:string			A string representation of the Job ID.
State	xs:string			One of "Waiting", "Working", or "Complete".
Phase	xs:string	0		One of "Processing Partial Restore", "Transmitting Data" or "Completing Operation".
TotalBytes	xs:long			The total number of bytes discovered in the asset.
BytesProcessed	xs:long			The number of bytes that have been transferred from the remote server.
TotalFiles	xs:int			The total number of files discovered in the asset.
FilesProcessed	xs:int			The number of files that have been transferred from the remote server.
TotalFrames	xs:int	0		Where available, the total number of frames in the asset.
PercentProcessed	xs:int			The percentage of the total data transfer that has completed.

Example 1

```
<Request>
  <Restore>
    <Priority>100</Priority>
    <AssetName>_AAOI0019</AssetName>
    <ServerID>Server1</ServerID>
    <ServerPath></ServerPath>
    <ArchivePath>FirstGroup</ArchivePath>
  </Restore>
</Request>
```

This request restores the asset archived in the example under "Archive" above to its original location on the same remote media server.

Example 2 (partial restore)

```
<Request>
  <Restore>
    <Priority>100</Priority>
    <AssetName>_AAOI0019</AssetName>
    <ServerID>Server1</ServerID>
    <ServerPath></ServerPath>
    <ServerFileName>_AAOI0019_01023000</ServerFileName>
    <ArchivePath>FirstGroup</ArchivePath>
    <Partial>
      <TimeIn>01:02:30:00</TimeIn>
      <TimeOut>01:03:05:00</TimeOut>
    </Partial>
```

```
  </Restore>
</Request>
```

This request performs a partial restore of the asset archived in the example under “Archive” above to its original location on the same remote media server but with a different file name on the remote server.

XML Schema for “failure” response

This is a standard “Fault” response described above under “Fault XML schema”.

Example

```
<Fault>
  <Code>4350</Code>
  <Message>
   Cannot restore _AAOI0019 because requested data cartridges
(Barcode:X0005BL4) are all offline
  </Message>
  <Class>
    <Message Message="3221231479" Param1="_AAOI0019"
Param2="Barcode:X0005BL4"/>
  </Class>
  <Module>Transaction.cpp</Module>
  <Line>329</Line>
</Fault>
```

Delete Command

The Delete command removes an asset from the archive by deleting all the files that comprise the asset. The asset to be deleted is specified either by name or by GUID. If a GUID is supplied the asset name is optional.

XML Schema

Name	Type	Min Occurs	Max Occurs	Description
Priority	xs:int			Priority of this request relative to other requests in the system. Requests with low numbers are processed before requests with high numbers; i.e. Priority 1 is the highest priority.
ArchivePath	xs:string	0		Path to the asset on the archive.
AssetName	xs:string	0		The name of the asset in the archive.
ArchiveGUID	xs:string	0		GUID that was supplied when the asset was created, if any.

XML Schema for "success" response to GetStatus command

Name	Type	Min Occurs	Max Occurs	Description
State	xs:string			One of "Waiting", "Working", or "Complete".
TotalBytes	xs:long			The total number of bytes in the asset.
TotalFiles	xs:int			The total number of files in the asset.

Examples

```
<Request>
  <Delete>
    <Priority>100</Priority>
    <AssetName>_AAOI0019</AssetName>
    <ArchivePath>FirstGroup</ArchivePath>
  </Delete>
</Request>

<Request>
  <Delete>
    <Priority>100</Priority>
    <ArchiveGUID>1111-1111-1111-1111</ArchiveGUID>
    <ArchivePath>FirstGroup</ArchivePath>
  </Delete>
</Request>
```

Polling with GetStatus might return the following:

```
<Response>
  <JobID>4D8CE12D-4EC3AB80-00-00000001</JobID>
  <State>Complete</State>
  <TotalBytes>2787709191</TotalBytes>
  <TotalFiles>3</TotalFiles>
</Response>
```

After execution of this command, the restore operation above would fail because the asset no longer exists on the archive.

ListObjects Command

The ListObjects command generates a list of all the objects at a particular level in the archive directory hierarchy. The listing can contain three types of object, which are "Asset", "File" and "Directory". If an object was written as an asset using the Archive command then its type will appear as "Asset" in the listing. Files that were not written as assets and that were put on the archive by some other means are will have a type of "File", whereas subdirectories are listed as objects with type "Directory".

The ListObjects request returns a JobID which is used to retrieve the object listing by means of the GetStatus command. Each invocation of the GetStatus command retrieves up to a maximum number of objects specified by the ObjectsPerRequest parameter (the default value for this parameter is 256).

XML Schema

Name	Type	Min Occurs	Max Occurs	Description
ArchivePath	xs:string			Path to the directory on the archive.
ObjectsPerRequest	xs:int	0		Maximum number of objects to return for each GetStatus request issued for the job.

XML Schema for "success" response to GetStatus command

Name	Type	Min Occurs	Max Occurs	Description
State	xs:string			One of "Working" or "Complete".
Objects	Object	0	unbounded	List of objects this response contains.

XML Schema for an Object returned in the list

Name	Type	Min Occurs	Max Occurs	Description
Type	xs:string			One of "Asset", "Directory", or "File".
Name	xs:string			The name of the object.

Examples

Find the objects at the root of the archive directory:

```
<Request>
  <ListObjects ArchivePath=""/>
</Request>

<Response>
  <JobID>4B45BA54-4CD97C45-00-00000006</JobID>
</Response>

<Request>
  <GetStatus>
    <JobID>4B45BA54-4CD97C45-00-00000006</JobID>
  </GetStatus>
</Request>
```

```
<Response>
  <JobID>4B45BA54-4CD97C45-00-00000006</JobID>
  <State>Complete</State>
  <Object>
    <Type>Directory</Type>
    <Name>FirstGroup</Name>
  </Object>
  <Object>
    <Type>Directory</Type>
    <Name>SecondGroup</Name>
  </Object>
</Response>
```

Find the objects one level down from the root:

```
<Request>
  <ListObjects ArchivePath="FirstGroup"/>
</Request>
...Response (JobID), GetStatus(JobID)...
<Response>
  <JobID>4B45BA54-4CD97C45-00-00000007</JobID>
  <State>Complete</State>
  <Object>
    <Type>Asset</Type>
    <Name>_AAOI0019</Name>
  </Object>
  <Object>
    <Type>Asset</Type>
    <Name>_AAOI0021</Name>
  </Object>
</Response>
```

Find the files that make up the asset:

```
<Request>
  <ListObjects ArchivePath="FirstGroup/_AAOI0019"/>
</Request>
...Response (JobID), GetStatus(JobID)...
<Response>
  <JobID>4B45BA54-4CD97C45-00-00000008</JobID>
  <State>Complete</State>
  <Object>
    <Type>File</Type>
    <Name>_AAOI0019.mov</Name>
  </Object>
</Response>
```

SearchObjects Command

If a search provider has been configured in the system configuration file (Configuration.xml) then the Workflow server will index all the files in the archive. The SearchObjects command is used to search the index.

The SearchObjects request returns a JobID which is used to retrieve the results by means of the GetStatus command. Each invocation of the GetStatus command retrieves up to a maximum number of objects specified by the ObjectsPerRequest parameter (the default value for this parameter is 256). It is also possible to specify an ObjectStartIndex which skips the specified number of results before starting the output, and an ObjectsMaxCount which specifies the maximum number of results that will be supplied.

XML Schema

Name	Type	Min Occurs	Max Occurs	Description
Query	xs:string			Query to be passed to the search provider.
ObjectsPerRequest	xs:int	0		Maximum number of objects to return for each GetStatus request issued for the job.
ObjectsStartIndex	xs:int	0		Skips this many rows in the result before starting the output.
ObjectsMaxCount	xs:int	0		Specifies the maximum number of results to return in total.
Columns	Column	1	unbounded	List of result columns required.

XML Schema for Column

Name	Type	Min Occurs	Max Occurs	Description
Name	xs:string			Column name.

XML Schema for "success" response to GetStatus command

Name	Type	Min Occurs	Max Occurs	Description
State	xs:string			One of "Working" or "Complete".
Result	Result	0	unbounded	List of results this response contains. The schema depends on the columns specified in the request.

Database schema

The default search provider for the Workflow server is XenWFDbSqlCE.DLL. This is an interface to Microsoft SQL Server CE version 4.0 (a free database) and it provides SQL based searching and indexing capabilities. By default, the XenWFDbSqlCE search provider creates a single table called "Objects". This table has four columns: Path (up to 1024 characters) ObjectName (up to 256 characters) Type (up to 8 characters) and AssetID (up to 256 characters). The table is indexed by Path, AssetID and ObjectName. Every file in the archive is included in the table; assets have Type "Asset" while essence files that are referenced by assets that are container files have type "File".

The AssetID is only defined for objects of type “Asset” and it is the ArchiveGUID if one was supplied when the object was archived, otherwise it is the AssetName.

Example search

```
<Request>
  <SearchObjects>
    <Query>SELECT ObjectName, Path FROM Objects WHERE
ObjectName='_AAOI0019.mov'</Query>
    <Column>ObjectName</Column>
    <Column>Path</Column>
  </SearchObjects>
</Request>

<Response>
  <JobID>4B45BA54-4CD97C45-00-00000007</JobID>
</Response>

<Request>
  <GetStatus JobID="4B45BA54-4CD97C45-00-00000007"/>
</Request>

<Response>
      <JobID>4B45BA54-4CD97C45-00-00000007</JobID>
      <State>Complete</State>
      <Result>
            <ObjectName>_AAOI0019.mov</ObjectName>
            <Path>FirstGroup/_AAOI0019</Path>
      </Result>
</Response>
```

ObjectExists Command

The ObjectExists command determines whether a particular object exists in the archive. It does not return a JobID, its result is either a Fault (object does not exist etc.) or a description of the object.

XML Schema

Name	Type	Min Occurs	Max Occurs	Description
ArchivePath	xs:string			Path to the asset on the archive.
AssetName	xs:string	0		The name of the object (asset, file or directory) on the archive.
ArchiveGUID	xs:string	0		GUID that was supplied when the asset was created, if any.

XML Schema for "success" response

Name	Type	Min Occurs	Max Occurs	Description
State	xs:string			"Complete".
Object	Object			Description of the object.

XML Schema for the Object

Name	Type	Min Occurs	Max Occurs	Description
Type	xs:string			One of "Asset", "Directory", or "File".
Name	xs:string			The name of the object.

Example

```
<Request>
  <ObjectExists AssetName="_AAOI0019" ArchivePath="FirstGroup"/>
</Request>

<Response>
  <State>Complete</State>
  <Object>
    <Type>Asset</Type>
    <Name>_AAOI0019</Name>
  </Object>
</Response>
```

XML Schema for "failure" response

This is a standard "Fault" response described above under "Fault XML schema".

ObjectLocation Command

The ObjectLocation command determines the location of an object in the archive (the names of the files that comprise the asset, which tape or set of tapes contain the files) and its status (online, nearline or offline). It does not return a JobID, its result is either a Fault (object does not exist etc.) or a description of the object.

XML Schema

The XML schema for the request is the same as that for the ObjectExists command.

XML Schema for "success" response

Name	Type	Min Occurs	Max Occurs	Description
State	xs:string			"Complete".
Object	Object			Description of the object.

XML Schema for the Object

Name	Type	Min Occurs	Max Occurs	Description
Type	xs:string			One of "Asset", "Directory", or "File".
Name	xs:string			The name of the object.
Volumes	Volume	0	unbounded	List of tape volumes that contain the data for the object.
File	xs:string	0	unbounded	List of file paths that comprise the object
Status	xs:string			Whether the asset is accessible. One of "Online", "Nearline", or "Offline".

XML Schema for a Volume contained within the Object schema

Name	Type	Min Occurs	Max Occurs	Description
Name	xs:string			Volume identity. This will usually contain the barcode labels of the media that comprise the volume.
Status	xs:string			Whether the volume is accessible. One of "Online" or "Offline".

Example

```
<Request>
  <ObjectLocation AssetName="_AAOI0019" ArchivePath="FirstGroup"/>
</Request>

<Response>
  <State>Complete</State>
  <Object>
    <Type>Asset</Type>
    <Name>_AAOI0019</Name>
    <Volume>
      <Name>Barcode:000014L</Name>
      <Status>Online</Status>
    </Volume>
    <Status>Online</Status>
```

```
    <File>X:\FirstGroup\_AAOI0019\media.dir\_AAOI0019.aiff</File>
    <File>X:\FirstGroup\_AAOI0019\media.dir\_AAOI0019.m2v</File>
    <File>X:\FirstGroup\_AAOI0019\_AAOI0019.mov</File>
    <Status>Nearline</Status>
  </Object>
</Response>
```


XML Schema for "failure" response

This is a standard "Fault" response described above under "Fault XML schema".

ListJobs Command

The ListJobs command generates a list of all the jobs known to the system. The ListJobs request returns a JobID which is used to retrieve the job listing by means of the GetStatus command. Each invocation of the GetStatus command retrieves up to a maximum number of objects specified by the ObjectsPerRequest parameter (the default value for this parameter is 256).

XML Schema

Name	Type	Min Occurs	Max Occurs	Description
ObjectsPerRequest	xs:int	0		Maximum number of objects to return for each GetStatus request issued for the job.

Example

```
<Request>
  <ListJobs/>
</Request>
<Response>
  <JobID>4B45BA54-4CD97C59-00-00000006</JobID>
</Response>

<Request>
  <GetStatus>
    <JobID>4B45BA54-4CD97C59-00-00000006</JobID>
  </GetStatus>
</Request>
<Response>
  <JobID>4B45BA54-4CD97C59-00-00000006</JobID>
  <State>Complete</State>
  <Job>
    <ID>4B45BA54-4CD97C59-00-00000002</ID>
    <State>Complete</State>
  </Job>
  <Job>
    <ID>4B45BA54-4CD97C59-00-00000003</ID>
    <State>Complete</State>
  </Job>
  <Job>
    <ID>4B45BA54-4CD97C59-00-00000004</ID>
    <State>Complete</State>
  </Job>
  <Job>
    <ID>4B45BA54-4CD97C59-00-00000005</ID>
    <State>Complete</State>
  </Job>
  <Job>
    <ID>4B45BA54-4CD97C59-00-00000006</ID>
    <State>Working</State>
  </Job>
</Response>
```

ListFileGroups Command

The ListFileGroups command provides information about file groups defined by the underlying XenData archive. It does not return a JobID, its result is either a Fault (file group does not exist etc.) or information about the requested file group(s). When called with no parameters, the command returns information about all file groups; if called with one or more file group names, it returns information about those file groups.

XML Schema

Name	Type	Min Occurs	Max Occurs	Description
FileGroups	FileGroup	0	unbounded	Optional list of file groups of interest.

XML Schema for the FileGroup

Name	Type	Min Occurs	Max Occurs	Description
Name	xs:string			The name of a file group.

Example

```
<Request>
  <ListFileGroups/>
</Request>

<Response>
  <FileGroup>
    <Name>Temporary files</Name>
    <Archive>false</Archive>
    <FragmentSize>0</FragmentSize>
    <IncludePath>*.tmp</IncludePath>
    <ExcludePath/>
  </FileGroup>
  <FileGroup>
    <Name>FirstGroup</Name>
    <Archive>true</Archive>
    <FragmentSize>0</FragmentSize>
    <IncludePath>/FirstGroup/.../*</IncludePath>
    <ExcludePath/>
    <ReadFlushTime>Never</ReadFlushTime>
    <WriteFlushTIme>Never</WriteFlushTIme>
    <VolumeSet>4D8CE12D-00000000</VolumeSet>
    <FreeSpace>453649629184</FreeSpace>
  </FileGroup>
</Response>

<Request>
  <ListFileGroups>
    <FileGroup Name="Default"/>
  </ListFileGroups>
</Request>

<Response>
  <FileGroup>
    <Name>Default</Name>
    <Archive>true</Archive>
    <FragmentSize>0</FragmentSize>
    <IncludePath>*</IncludePath>
    <ExcludePath/>
    <ReadFlushTime>Immediate</ReadFlushTime>
    <WriteFlushTIme>Immediate</WriteFlushTIme>
```

```
    <VolumeSet>4D8CE12D-00000000</VolumeSet>
    <FreeSpace>1291675762688</FreeSpace>
  </FileGroup>
</Response>
```

ListVolumeSets Command

The ListVolumeSets command provides information about volume sets defined by the underlying XenData archive. It does not return a JobID, its result is either a Fault (volume set does not exist etc.) or information about the requested volume set(s). When called with no parameters, the command returns information about all volume sets; if called with one or more volume set names or identities, it returns information about those volume sets.

XML Schema

Name	Type	Min Occurs	Max Occurs	Description
VolumeSet	VolumeSet	0	unbounded	Optional list of volume sets of interest.

XML Schema for the VolumeSet

Name	Type	Min Occurs	Max Occurs	Description
Name	xs:string	0		The name of a volume set.
ID	xs_string	0		A volume set ID.

Example

```
<Request>
  <ListVolumeSets ID="4D8CE12D-00000001"/>
</Request>

<Response>
  <VolumeSet>
    <Name>Archive Tapes</Name>
    <Identity>4D8CE12D-00000001</Identity>
    <MediaReplicas>1</MediaReplicas>
    <ConcurrentWrites>0</ConcurrentWrites>
    <FragmentSize>0</FragmentSize>
    <PhysicalBlockSize>0</PhysicalBlockSize>
    <Compressed>false</Compressed>
    <Deferred>false</Deferred>
    <GroupedReplicas>false</GroupedReplicas>
    <WriteIfNoMediaAvailable>false</WriteIfNoMediaAvailable>
    <UnusedCapacity>0.02</UnusedCapacity>
    <NewVolumeCapacity>0.95</NewVolumeCapacity>
    <WriteOnce>false</WriteOnce>
    <Capacity>0</Capacity>
    <NumVolumes>193</NumVolumes>
 </VolumeSet>
</Response>
```

GetStatus Command

The GetStatus command is used to retrieve the status of a job within the Workflow server. The results returned for this request depend on the job being queried: please see descriptions of individual requests for more information and examples.

XML Schema

Name	Type	Min Occurs	Max Occurs	Description
JobID	xs:string			The ID of the job being queried.

CancelJob Command

The CancelJob command is used to cancel processing of a job. The result returned by the request is what would have been returned by a GetStatus request issued immediately before the job was cancelled.

XML Schema

Name	Type	Min Occurs	Max Occurs	Description
JobID	xs:string			The ID of the job being cancelled.

Example

```
<Request>
  <CancelJob JobID="4D8CE12D-4EC4F822-00-00000004"/>
</Request>

<Response>
  <JobID>4D8CE12D-4EC4F822-00-00000004</JobID>
  <State>Working</State>
  <TotalBytes>2787709191</TotalBytes>
  <BytesProcessed>544622624</BytesProcessed>
  <TotalFiles>3</TotalFiles>
  <FilesProcessed>1</FilesProcessed>
  <PercentProcessed>19</PercentProcessed>
</Response>
```

The job that was cancelled now indicates a fault showing that it was cancelled externally:

```
<Request>
  <GetStatus JobID="4D8CE12D-4EC4F822-00-00000004"/>
</Request>

<Fault>
  <JobID>4D8CE12D-4EC4F822-00-00000004</JobID>
  <TotalBytes>2787709191</TotalBytes>
  <BytesProcessed>544884768</BytesProcessed>
  <TotalFiles>3</TotalFiles>
  <FilesProcessed>1</FilesProcessed>
  <PercentProcessed>19</PercentProcessed>
  <Code>87</Code>
  <Event>3221231484</Event>
  <Message>The request was externally cancelled</Message>
  <JobID>4D8CE12D-4EC4F822-00-00000004</JobID>
  <Class>
    <Message Message="3221231484"/>
  </Class>
  <Module>Request.cpp</Module>
  <Line>1055</Line>
</Fault>
```

SystemStatus Command

The SystemStatus command returns information about the Workflow server, the underlying XenData archive and connection information for the client that made the request. . It does not return a JobID, its result is returned immediately. The ArchiveStatus field in the response may be used to determine if the archive is ready to accept requests.

Examples:

```
<Request>
  <SystemStatus/>
</Request>

<Response>
  <State>Complete</State>
  <ProductName>XenData Workflow API</ProductName>
  <ProductVersion>1.02</ProductVersion>
  <ArchiveStatus>OnLine</ArchiveStatus>
  <ArchiveRoot>X:\</ArchiveRoot>
  <ServerIPAddress>192.168.1.1</ServerIPAddress>
  <ServerPort>3466</ServerPort>
  <ClientIPAddress>192.168.1.100</ClientIPAddress>
  <ClientPort>49350</ClientPort>
</Response>

<Request>
  <SystemStatus/>
</Request>

<Response>
  <State>Complete</State>
  <ProductName>XenData Workflow API</ProductName>
  <ProductVersion>1.01</ProductVersion>
  <ArchiveError>The device is not ready</ArchiveError>
  <ArchiveStatus>Offline</ArchiveStatus>
  <ArchiveRoot>X:\</ArchiveRoot>
  <ServerIPAddress>192.168.1.1</ServerIPAddress>
  <ServerPort>3466</ServerPort>
  <ClientIPAddress>192.168.1.100</ClientIPAddress>
  <ClientPort>49350</ClientPort>
</Response>
```