

Aperio Viewport ActiveX Control (.OCX)

Programmer's Reference



Copyright © 2004-2008 Aperio Technologies, Inc.

Part Number/Revision: MAN-0072, Revision C

Date: December 9, 2008

This document applies to software versions Release 10.0 and later.

All rights reserved. This document may not be copied in whole or in part or reproduced in any other media without the express written permission of Aperio Technologies, Inc. Please note that under copyright law, copying includes translation into another language.

User Resources

For the latest information on Aperio Technologies products and services, please visit the Aperio Technologies website at: <http://www.aperio.com>.

Disclaimers

This manual is not a substitute for the detailed operator training provided by Aperio Technologies, Inc., or for other advanced instruction. Aperio Technologies Field Representatives should be contacted immediately for assistance in the event of any instrument malfunction. Installation of hardware should only be performed by a certified Aperio Technologies Service Engineer.

ImageServer is intended for use with the SVS file format (the native format for digital slides created by scanning glass slides with the ScanScope scanner). Educators will use Aperio software to view and modify digital slides in Composite WebSlide (CWS) format.

Aperio products are FDA cleared for specific clinical applications, and are intended for research use for other applications.

Trademarks and Patents

ScanScope is a registered trademark and ImageServer, TMA Lab, ImageScope, and Spectrum are trademarks of Aperio Technologies, Inc. All other trade names and trademarks are the property of their respective holders.

Aperio products are protected by U.S. Patents: 6,711,283; 6,917,696; 7,035,478; 7,116,440; 7,428,324; and 7,457,446, and licensed under one or more of the following U.S. Patents: 6,101,265; 6,272,235; 6,522,774; 6,775,402; 6,396,941; 6,674,881; 6,226,392; 6,404,906; 6,674,884; and 6,466,690.

Contact Information

Headquarters: Aperio Technologies, Inc.
1360 Park Center Drive
Vista, CA 92081
United States

European Office: Aperio
3 The Sanctuary
Eden Office Park
Ham Green
Bristol BS20 0DD, UK

United States of America

Tel: 866-478-4111 (toll free)

Fax: 760-539-1116

Customer Service Tel: 866-478-4111 (toll free)

Technical Support Tel: 866-478-3999 (toll free)

Email: support@aperio.com

Europe

Tel: +44 (0) 1275 375123

Fax: +44(0) 1275 373501

Customer Service Tel: +44 (0) 1275 375123

Technical Support Tel: +44 (0) 1275 375123

Email: europesupport@aperio.com

Aperio Viewport ActiveX Control (.OCX) Programmer's Reference

The Aperio **viewport** is an ActiveX control (.OCX) that can be included in a variety of programs and configurations. The **viewport** displays image files. Features include:

- Support for TIFF files with various compression schemes (raw, LZW lossless, JPEG lossy, and JPEG2000 lossy), and organizations (raster, stripped, tiled).
- Support for JPEG input files (JFIF, lossy).
- Support for “compound” files. A compound file is a text file (.txt) that describes a mosaic of image files that are composited together to form the source image. Each line in the text file contains an image filename and the X and Y offsets within the compound image at which the image is positioned.
- Support for “Composite WebSlides” (CWS). A composite webslide is a directory that contains many small JPEG files and one or more INI files which contain image information¹.
- Support for large images (up to 250,000+ pixels in width and height).
- Fast panning and zooming. A memory cache is used to save retrieved tiles from disk, and prefetching is performed to pre-populate the cache in anticipation of user actions.
- Support for image adjustments – gamma maps and convolution filters.
- Support for varying zoom levels if stored in file (image “pyramid”). This speeds display of images at high zoom levels.
- Support for annotation display (annotations encoded as XML strings).
- Support for interface to server-side algorithms.
- Support for ICC Aware applications.

Aperio grants to those users of Aperio’s published APIs (TiffComp, Viewport, ImageNav, and other modules that may be made available) a nonexclusive right to use for your personal/internal use. Commercial redistribution is strictly prohibited. Non-commercial redistribution is permitted provided that its original configuration is not modified without Aperio’s express written consent and its origin is not misrepresented.

Contents

Contents	3
Introduction.....	4
Read-only Properties.....	5
Directory Entry Properties (Read-only).....	8

¹A Composite WebSlide, also known as a CWS slide, is a proprietary format created by Bacus Laboratories, Inc. (“Bacus”). WebSlide is a registered trademark of Bacus Laboratories, Inc.

Modifiable Properties.....	9
Methods.....	12
Events.....	16
Local/Remote Files and Directories.....	16
Caching.....	17
Annotations Overview.....	17
Annotations - Collection Object.....	21
Annotation – Object.....	22
Regions - Collection Object.....	22
Region – Object.....	23
Attributes – Collection Object.....	24
Attribute – Object.....	24
AttributeHeaders - Collection Object.....	24
AttributeHeader - Object.....	25
Algorithms.....	25

Introduction

The **viewport** can access files in a Windows filesystem or can connect via TCP/IP to a server running Aperio ImageServer software for remote image access. Facilities are provided to iterate through directory contents and for accessing thumbnails of images for directory display. For remote connections to ImageServer, the viewport supports [optional] authentication; a userid and password may be supplied as authentication credentials which are validated by the server.

The **viewport** renders a rectangular region of the image that fills the control window. There is no border and the control has no UI features other than the image itself. The control may be minimized, maximized, or resized to any proportions at any time. The control implements the ActiveX “simple frame” functionality, allowing it to act as the host for other controls. Mouse messages (clicks, dragging, movement, etc) and keystrokes are passed through to the parent program.

The **viewport** uses the concept of a current *view*. The view is the region of an image that is currently displayed in the control's window. When the zoom setting is 1, the view is the same size as the control window. When zoom is .5, the view is twice the size of the control window (50% of baseline resolution).

The **viewport** has extensive annotation support. Annotations consist of **regions** which are grouped into **layers**. Objects and methods are provided to manage and manipulate annotations. Annotations may be serialized as XML and stored and retrieved in local files or remotely via ImageServer.

The **viewport** provides an interface for running algorithms on a server, using Aperio's Algorithm Framework architecture². Algorithm regions of analysis are defined as annotations, and

² Aperio's Algorithm Framework is documented separately; please contact Aperio for more information.

algorithm results are returned as annotations also. Methods are provided for scheduling algorithms and retrieving algorithm status.

Read-only Properties

Property	Type	Description
CacheUsed	Long	Current amount of memory cache used, in MB.
CacheUsedHWM	Long	High water mark for memory cache, in MB.
CacheAvail	Long	Current available free memory in computer, in MB.
CacheTotal	Long	Total memory in computer, in MB.
CacheMax	Long	Current cache maximum size, in MB. This value is dynamically adjusted (see "Caching" on page17).
CacheBuffers	Long	Current number of buffers stored in cache.
CacheBuffersHWM	Long	High water mark for buffers stored in cache.
CacheSearches	Long	Number of cache searches.
CacheHits	Long	Number of cache hits.
ErrorCode	Short	Error code corresponding to most recent error. This property is set to zero during initialization and at start of Open() and Directory() methods.
ErrorMessage	BSTR	Text string describing most recent error.
HasLabel	Bool	T/F whether image has label
HasMacro	Bool	T/F whether image has macro image
HasThumbnail	Bool	T/F whether image has thumbnail
ImageAppMag	Short	Apparent magnification of image (if known, else 0). Given as optical microscope equivalent, e.g., 20 means "20x", or as if image were viewed through an optical microscope with a 20x objective lens ³ .
ImageBitDepth	Long	Number of bits per pixel (e.g. RGB = 24)
ImageChannels	Long	Number of color channels per pixel (e.g. RGB = 3)
ImageCompCodec	BSTR	Text string giving compression codec info, if any
ImageCompQuality	Short	Image compression quality value (if any, else 0)
ImageCompRatio	Float	Image compression ratio (if any, else 1)

³ ScanScope images are scanned at .46μ/pixel (20x objective) or .23μ/pixel (40x objective).

Property	Type	Description
ImageCompType	Short	Image compression type (and file type): 0 – raw TIFF (uncompressed) – TIF file (TIFF) 1 – LZW (lossless) – TIF file (TIFF) 2 – JPEG (lossy) – SVS file (TIFF) 3 – JPEG2000 (lossy) – SVS file (TIFF) 7 – YUYV – SVS file (TIFF) 8 – JPEG (lossy) – JPG file (JFIF) 10 – JPEG (lossy) – CWS directory (JFIF, INI files) 12 – JPEG 2000 (lossy – JP2 file.
ImageDescription	BSTR	Text string giving image description, if any.
ImageFileSize	Float	Actual file size in bytes
ImageDepth	Long	Current image depth, in pixels.
ImageHeight	Long	Current image height, in pixels.
ImageWidth	Long	Current image width in pixels.
ImagePath	BSTR	Current image file path
ImageServerName	BSTR	Current image server name, returns blank if local file
ImageServerPort	Short	Current image server port, returns zero if local file
ImageServerVersion	Long	ImageServer version returned as major * 1000 + minor (e.g. 9.01 is 9001). If image is not opened through ImageServer, returns 0
ImageSize	Float	Active image size, in bytes. Includes all layers.
ImageTitle	BSTR	Current image title (if any)
Is3D	Bool	T/F whether image has depth > 1
LabelHeight	Long	Height of label image (if any)
LabelWidth	Long	Width of label image (if any)
LabelImage	LPPICTUREDISP	Returns label image as an ActiveX picture, or NULL if image has no associated label.
Layers	Short	Number of layers in image.
LayerDepth (short layer)	Long	Depth of specified layer in pixels.
LayerFocus (short layer)	Long	Value for center of specified layer (-1 -> +1).
LayerZ (short layer)	Long	Z-offset to top of specified layer, a Z value of 0 is “closest.”
LayerLevels (short layer)	Short	Number of pyramid levels in the specified layer.
LayerLevelHeight(short layer, short level)	Long	Returns the height of the specified level in the specified layer.
LayerLevelWidth(short layer, short level)	Long	Returns the width of the specified level in the specified layer.

Property	Type	Description
LayerLevelZoom(short layer, short level)	Double	Returns the zoom of the specified level of the specified layer relative to base.
Levels	Short	Number of pyramid levels in image (if any, else 0)
LevelHeight (short level)	Long	Returns intermediate level image height, in pixels.
LevelWidth (short level)	Long	Returns intermediate level image width, in pixels.
MacroHeight	Long	Height of macro image (if any)
MacroWidth	Long	Width of macro image (if any)
MacroImage	LPPICTUREDISP	Returns macro image as an ActiveX picture, or NULL if image has no associated label.
Seams	Short	Number of seams in image (if any, else 0)
SeamHighlight	Bool	T/F whether seams in composite images are highlighted with 1-pixel black line (default false).
SeamQuality (short seam)	Float	Value indicating seam quality; measure of overlap between files. 0.0 = bad, 1.0 = perfect.
SeamX (short seam)	Long	X-coordinate of specified seam. Zero means seams are horizontal.
SeamY (short seam)	Long	Y-coordinate of specified seam. Zero means seams are vertical.
SourceProfileName	BSTR	Text string giving the name of the ICC profile embedded in the image. If the image does not have an embedded profile, returns null.
TileHeight	Short	Image tile height. If image(s) are stripped [not tiled], returns rows per strip.
TileWidth	Short	Image tile width. If image(s) are stripped [not tiled], returns 0.
Thumbnail (bool MRUmap)	LPPICTUREDISP	Returns thumbnail image corresponding to currently loaded image, as ActiveX picture. Returns NULL if image has no associated thumbnail. The MRUmap parameter should be false .
ThumbData (long X, long Y, short channel)	Short	Returns data value from thumbnail image, at specified X, Y coordinates, for specified channel (0=R, 1=G, 2=B).
ThumbHeight	Long	Returns thumbnail image height, in pixels.
ThumbRegion (long X, long Y, long width, long height)	LPPICTUREDISP	Returns requested region of thumbnail image, as ActiveX picture. Returns NULL if image has no associated thumbnail.
ThumbWidth	Long	Returns thumbnail image width, in pixels.
Version	BSTR	Text string giving OCX version.

Property	Type	Description
ViewHeight	Long	Current "view" height in pixels. At a zoom setting of 1 this is the same as the control's window size.
ViewWidth	Long	Current "view" width in pixels.

Directory Entry Properties (Read-only)

The following read-only properties are returned for the current directory or a specified directory entry (following a successful call to the Directory method to load a directory's contents). Note that properties marked with * are only valid for image files. Directory entries are returned sorted by name [case-insensitive] within type (e.g., subdirectories are returned prior to any files).

Property	Type	Description
DirPath	BSTR	Current directory path
DirServerName	BSTR	Current directory server name (blank if local)
DirServerPort	Short	Current directory server port (or zero)
DirTitle	BSTR	Current directory's title (if any)
DirEntName (short index)	BSTR	Directory entry filename
DirEntType (short index)	Short	Directory entry type: 0=dir, 1=link, 2=file, 3=image file (tiled TIFF or SVS)
DirEntSize (short index)	Float	Directory entry file size in bytes
*DirEntDepth (short index)	Long	Depth of image file (pixels)
*DirEntWidth (short index)	Long	Width of image file (pixels)
*DirEntHeight (short index)	Long	Height of image file (pixels)
*DirEntTileWidth (short index)	Short	Width of image file tiles (pixels).
*DirEntTileHeight (short index)	Short	Height of image file tiles (pixels).
*DirEntTitle (short index)	BSTR	Image file title (if any)
*DirEntDesc (short index)	BSTR	Image file description string
*DirEntThumbnail (short index, long width, long height)	LPPICTUREDISP	Returns thumbnail image for directory entry, as ActiveX picture (or NULL). Width and height specify desired size of image; if height is zero, image aspect ratio is preserved.
*DirEntLabel (short index, long width, long height)	LPPICTUREDISP	Returns label image for directory entry, as ActiveX picture (or NULL). Width and height specify desired size of image; if height is zero, image aspect ratio is preserved.
*DirEntMacro (short index, long width, long height)	LPPICTUREDISP	Returns macro image for directory entry, as ActiveX picture (or NULL). Width and height specify desired size of image; if height is zero, image aspect ratio is preserved.

Modifiable Properties

Property	Type	Description
Annotations	IAnnotations	The annotations collection object (if any). This collection allows you to access and modify the list of annotation layers.
BackColor	OLE_COLOR	Color used to paint control window when no image has been loaded, or for matte around image if visible area smaller than control window.
CacheMemory	Long	Maximum size of in-core memory cache, in MB. The default is the available memory in the machine.
CacheMRU	Bool	Cache priority routine; true = MRU, false = "view" (default).
EnableAuthEvent	Bool	Whether to fire the Authenticate event if an http 401 error is returned for an image request. Default is false .
FalseColor	Bool	T/F whether false coloring is enabled. Only valid for monochrome images where ImageChannels = 1.
FalseHue	Long	The false color hue, range 0-255.
Filter	Bool	Whether to apply filter (convolution matrix) to image data before rendering display. Default is false .
FilterMatrix (int x, int y)	Short	Convolution matrix value, $-2 \leq x \leq 2$ and $-2 \leq y \leq 2$. (0,0) value gives pixel weight.
FilterOffset	Short	Filter offset; sets bias for pixel values. Default is zero.
FilterScale	Short	Filter scale factor; re-computed when a FilterMatrix value is updated to be the sum of all matrix elements.
FilterType	Short	Designate predefined filter, or custom (default=0): 0 – custom filter (FilterMatrix, FilterScale, FilterOffset) 1 – unsharp mask, SD = 0.6 2 – unsharp mask, SD = 0.7 3 – unsharp mask, SD = 0.8 4 – unsharp mask, SD = 0.9 5 – unsharp mark, SD = 1.0 6 – edge enhance K1 7 – edge enhance K2 (softer)
Focus	Double	Current focus level which varies from -1 to +1 (closest to furthest away), along some axis (currently always Z). All views are perpendicular to the focus axis.

Property	Type	Description
ICM	Bool	T/F whether integrated color management is performed on image. Default is FALSE. Only valid when there exists a source ICC profile. See <code>SourceProfileName</code> and <code>SetTargetProfilePath</code> .
Interpolate	Bool	T/F whether interpolation is performed between layers. If not, just use “closest” layer for current focus value. Default is false
LabelTransform	Short	Orthogonal transformation to be applied to label image, as follows: 0 – normal 1 – rotate 90° clockwise 2 – rotate 180° 3 – rotate 90° counter-clockwise Note this value may affects the value of the <code>LabelWidth</code> and <code>LabelHeight</code> properties
Log	Bool	T/F whether logging is enabled. When enabled, processing messages are written to a file named <code>viewport.log</code> , located in the current directory. Default is false .
LogPath	BSTR	Path to <code>viewport.log</code> file; default is “.” (current directory).
OverlayHeight	Long	Gets/sets the height of the overlay image in pixels. Viewport currently only supports one overlay image, so index is ignored.
OverlayHeight	Long	Gets/sets the height of the overlay image in pixels. Viewport currently only supports one overlay image, so index is ignored.
OverlayLeft	Long	Gets/Sets the X offset pixel used to determine where the overlay image should be positioned on the main image. Viewport currently only supports one overlay image, so index is ignored.
OverlayTop	Long	Gets/Sets the Y offset pixel used to determine where the overlay image should be positioned on the main image. Viewport currently only supports one overlay image, so index is ignored.
OverlayWidth	Long	Gets/sets the width of the overlay image in pixels. Viewport currently only supports one overlay image, so index is ignored.
OverlayVisible	Bool	T/F whether the overlay image is visible. Viewport currently only supports one overlay image, so index is ignored.
Password	BSTR	If supplied, gives password to be passed for authentication on remote connections. Must be set prior to calling <code>Open()</code> or <code>Directory()</code> methods.

Property	Type	Description
PixelSmooth	Bool	Use high-fidelity (slow) scaling routine. Default is false .
Prefetch	Bool	T/F whether prefetching is enabled. When enabled, image tiles are prefetched into the cache to improve performance.
PrefetchThreads	Short	Number of prefetch threads (when progressive rendering or prefetching are enabled). Default is 1.
ProgressiveRendering	Bool	T/F whether image data is drawn progressively (as each resolution level is loaded into memory). When enabled, responsiveness of containing application is improved (especially when accessing remote images).
ProxyServerName	BSTR	Host name or IP address of HTTP proxy server. Default is blank , which means don't use a proxy server.
ProxyServerPassword	BSTR	Password passed to HTTP proxy server.
ProxyServerPort	Short	TCP/IP port for HTTP proxy server. Default is 80 .
ProxyServerUserid	BSTR	Userid passed to HTTP proxy server. Default is blank , which means no proxy authentication is performed.
ProxyServerUseWindows	Bool	Whether to use the HTTP proxy settings in the Windows registry (enabled / server / port). Default is true .
SnapCompType	Short	Compression type for SnapShot() images: 0 – raw TIFF (uncompressed) - default 1 – LZW TIFF (lossless) 2 – JPEG (lossy)
SnapCompQuality	Short	Compression quality factor for JPEG SnapShot() images, 0-100. Ignored unless SnapCompType = 2.
Transform	Long	Gets/Sets the transform for the image. The transform values are: 0 – none (0 degrees) 1 - rotate 90 degrees right (clockwise) 2 - rotate 180 degrees 3 - rotate 90 degrees left (counter-clockwise) 4 - flip vertical 5 - rotate 90 degrees right, flip 6 - flip horizontal 7 - rotate 90 degrees left, flip
UserAgent	BSTR	If supplied, gives the User-Agent identification string passed to the server on remote requests. By default, no User-Agent identification is set.
Userid	BSTR	If supplied, gives userid to be passed for authentication on remote connections. Must be set prior to calling Open() or Directory() methods.

Property	Type	Description
ViewClip	Bool	T/F whether image view (X, Y, width, height) is clipped to viewport dimensions. Matte around image area is filled with BackColor.
ViewProgress	Short	Percent complete rendering current view (if progressive rendering, otherwise always 100).
ViewX	Long	Offset from left edge of image to current "view" in pixels. This property is reset when an image is loaded to center the view on the image.
ViewY	Long	Offset from top edge of image to current "view" in pixels. This property is reset when an image is loaded to center the view on the image.
Zoom	Double	Current zoom level. If this value is 1 (default), each pixel in the control window corresponds to one pixel in the image. If this value is .5 (50%), each pixel in the control window corresponds to a square of four pixels from the source image. Zoom values greater than one cause window pixels to be interpolated between image pixels.

Methods

Method Prototype	Description
void Close()	Close current image.
void CloseOverlay (Integer Index)	Close the overlay image. Currently the viewport only supports a single overlay image, so Index is ignored.
bool Delete()	Delete current image (and associated files, if any).
short Directory (LPCTSTR dirpath)	Load directory information. The dirpath may specify a local or remote directory (see "Local/Remote Files and Directories" on page 16). The routine returns the number of entries in the directory, or -1 in case of error.
short DirUpdate()	Update previously loaded directory information. This is more efficient than another Directory() for the same dirpath . Returns the number of entries in the directory, or -1 if error.
BSTR DoDBCall(BSTR callstring)	Used to send a database call to DataServer.
void DoLog(LPCTSTR msg)	Write string msg into viewport log.

Method Prototype	Description
bool GetAlgorithmStatus(long jobQueueId, short *status, short *progress, short *outputLayer)	Returns status of algorithm for the current image, specified by jobQueueId . The returned status parameter has values: 0 = queued - algorithm in queue, not yet started 1 = active - algorithm run in progress 2 = completed - algorithm run completed successfully -1 = error - algorithm run completed in error progress is the %complete in the range 0-100 (0 if the algorithm is queued or if the run completed in error, 100 if the run completed successfully). outputLayer is the annotationId of the annotation layer which contains algorithm results.
BSTR GetAnnotationXML()	Returns annotations' XML string, if any, for the current image. If the image is "local" the <image>.xml file is read, if the image is "remote" the XML string is requested from the server. The Viewport does not cache this string, on each GetAnnotationXML() request the string is re-accessed from the source (enabling cross-updating).
BSTR GetAttribute (LPCTSTR name)	Returns attribute value. Attributes are application information stored in the form "name=value." The first attribute with a given name is returned.
BSTR GetGammaB()	Returns gamma array for Blue channel (256 bytes).
BSTR GetGammaG()	Returns gamma array for Green channel (256 bytes).
BSTR GetGammaR()	Returns gamma array for Red channel (256 bytes).
BSTR GetMacroInfo(BSTR MacroID)	Returns macro information for the macro specified by MacroID
BSTR GetOverlayGammaB()	Returns Blue channel gamma (256 bytes) for overlay images.
BSTR GetOverlayGammaG()	Returns Green channel gamma (256 bytes) for overlay images.
BSTR GetOverlayGammaR()	Returns Red channel gamma (256 bytes) for overlay images.
bool ImageAttributes(LPCTSTR filepath, long *imageWidth, long *imageHeight, long *blockWidth, long *blockHeight, BSTR *title, BSTR *description)	Returns attribute information for an image file. This is independent of any currently loaded directory or opened file. The image dimensions, image block dimensions, image title, and image description string are returned. The method returns true if successful, else false ; in case of errors ErrorCode and ErrorMessage are set appropriately.

Method Prototype	Description
LPPICTUREDISP ImageThumbnail (LPCTSTR filepath)	Returns thumbnail image for an image file, as ActiveX picture (or NULL). Width and height specify desired size of image; if height is zero, image aspect ratio is preserved.
LPPICTUREDISP ImageLabel (LPCTSTR filepath)	Returns label image for an image file, as ActiveX picture (or NULL). Width and height specify desired size of image; if height is zero, image aspect ratio is preserved.
Bool Open (LPCTSTR filepath)	Open image file and display initial view centered on image. The Filepath may specify a local or remote file (see "Local/Remote Files and Directories" on page 16). The routine returns true if successful, else false .
Bool OpenOverlay(Integer Index, LPCTSTR FilePath)	Open the overlay image at the given filepath (local or remote). Currently only a single overlay image is supported, so the Index parameter is ignored. After an overlay image is successfully opened, you should set the OverlayTop, OverlayLeft, OverlayWidth, and OverlayHeight properties.
Bool PutAnnotations (LPCSTR annotations)	Stores annotations' XML string for the current image (replacing any previous string). If the image is "local," the <image>.xml file is created/updated, if the image is "remote," the XML string is stored to the server. This routine returns true if successful, false on error.
Bool RemoteRequest(LPCSTR server, short port, LPCSTR request, BSTR *response, long buflen, long *rsplen)	Perform a generic remote request to an ImageServer. The server parameter gives the host name or IP address of the server machine, and port specifies the TCP/IP port. The request string is passed intact to the server, and the response string is returned with the server's response data. The buflen parameter gives the maximum size of the response, and the rsplen parameter is set with the actual size of the response.
Bool Rename (LPCTSTR newpath)	rename image file (and associated files, if any)
bool RunAlgorithm(LPCTSTR macro, long inputLayer, bool markupImage, long* jobId, BSTR macroXML, long outputLayer)	Causes a request to be made to run an algorithm for the current image. The macro parameter gives the name of the algorithm to be run. The markupImage parameter specifies whether a markup image will be generated. The inputLayer parameter specifies the annotationId of the annotation layer to be used as the input region of analysis by the algorithms; if zero, the entire virtual slide is used as the region of analysis. The jobId parameter receives the job queue id. The optional macroXML parameter overrides the existing XML in the macro. The optional outputLayer specifies the output layer for the algorithm's results.

Method Prototype	Description
void SetGamma(LPCTSTR R, LPCTSTR G, LPCTSTR B)	Set gamma arrays for Red, Green, and Blue channels. Each array contains 256 bytes that describe the output value for each possible input value in the channel. The default array values map each input value to the identical output value. DEPRECATED, use SetGammaBytes instead.
void SetGammaBytes(long R, long G, long B)	Same as SetGamma, but modified to work for MBCS systems, such as Japanese windows.
void SetOverlayGamma(LPCTSTR R, LPCTSTR G, LPCTSTR B)	Set gamma mapping as with SetGamma but for overlay images (algorithm markup images, z-stacks, etc.). DEPRECATED, use SetOverlayGammaBytes instead.
void SetOverlayGammaBytes(long R, long G, long B)	Same as SetOverlayGamma, but modified to work for MBCS systems, such as Japanese windows.
void SetRulerEndPos(long EndY, long EndX)	No longer supported. Rulers are stored as annotations.
void SetRulerStartPos(long StartY, long StartX)	No longer supported. Rulers are stored as annotations.
BOOL SetTargetProfilePath (LPCTSTR path)	Specify path & filename of target monitor profile for ICM processing. See ICM and SourceProfilePath.
void SetViewXY (long X, long Y)	Set current values of ViewX and ViewY properties to position current view on image. Using this method is better than setting the ViewX and ViewY properties separately because it enables smooth diagonal scrolling.
BSTR Snapdata (bool doAnnotations)	Return bitmap of current control window, encoded as string. Each pixel has three values corresponding to R, G, and B. The dimensions of the bitmap are the same as the current window. The doAnnotations parameter determines whether annotations are rendered into the bitmap.
bool SnapShot (LPCTSTR filepath)	Take snapshot of current control window and write as TIFF or JPEG image into specified file. The file type and compression type are given by SnapCompType property, and the JPEG quality is given by the SnapCompQuality property.
bool TransformPoint (double* X, double* Y)	Transforms an X/Y point from zero degrees to the Viewport's current transform (see Transform property).

Method Prototype	Description
bool UnTransformPoint (double* X, double* Y)	Transforms an X/Y point from Viewport's current transform to a zero degree transform (see Transform property).

Events

Event Handler Prototype	Description
void AlgorithmStatus (BSTR macro, long inputLayer, short status, short progress, long outputLayer)	This event is fired each time an update is available for algorithm status. After a status of "completed" or "error" the event will no longer be fired (until the RunAlgorithm() method is called again).
void ViewUpdate (short progress)	Fired whenever percent rendering complete for the current view changes (if progressive rendering). Progress is reported as percent complete (0-100).
void Authenticate (short unused)	Fired whenever an http 401 error is encountered. The event handler should prompt the user for credentials, and then set Viewport's Userid and Password properties. This event is fired only if the EnableAuthEvent has been set to TRUE.

Local/Remote Files and Directories

The **viewport** may be used to access both local and remote files. "Local" files are accessed through a Windows filesystem (this includes mapped network shares and drives), while "remote" files are accessed via a TCP/IP connection to the Aperio Web Server.

When a directory or file path is specified (to the Open or Directory methods, described above), the path for a remote file or directory is given as:

```
//<server_name>:<server_port>/<file_path>
```

The <server_name> must be resolvable through DNS or may be specified as an IP address in dotted format. The <server_port> is optional; if omitted, port 80 is assumed (default for HTTP access). If the path specified does not conform to this syntax, it is assumed to be for a local file. Note that the leading slashes should be forward slashes, backward slashes may be used to indicate a file or directory on a Windows share, which may be "remote" in the sense that it is on another computer, but it is "local" from a standpoint of the **viewport**.

When accessing local files, the **viewport** remains open in shared mode on a file while it is open. In some cases more than one handle may be obtained for a file, such as when there are multiple "pyramid" images within the file.

When accessing remote files, the **viewport** uses the HTTP/1.1 protocol over a TCP/IP socket connection. Multiple sockets may be connected concurrently to improve performance. Under HTTP/1.1, socket connections are persistent across multiple requests, but may be closed by either side for any reason. When a socket connection is closed by the server (such as for a connection timeout), the **viewport** will reconnect automatically as required. The Close method causes all active socket connections to be closed.

For remote connections, the **viewport** communicates with the Aperio Web Server (AWS). This interface provides support for authentication on each request, on a per-connection, per-directory, per-image basis. Authentication credentials are established with the Userid and Password properties, which are validated by the server. Typically, an application program calls Open() and/or Directory() without knowing whether authentication is required. If these methods fail (return false) with a status of "authentication required" (ErrorCode set to 121), the application program prompts the user for entry of [new] authentication credentials. The program then [re]sets the Userid and Password properties, and calls Open() and/or Directory() again with the [new] credentials.

Caching

The Viewport uses a memory cache to improve performance. The size of the cache is dynamically adjusted to match the available memory of the computer. Initially all available memory will be used, but as other programs are run and/or the memory of the computer is used for other purposes, the cache size will dynamically adjust to match. As the cache is adjusted, lower priority blocks from images are released.

All instances of the Viewport within one program share one cache (that is, multiple images may be open simultaneously). If several programs are using the Viewport the cache sizes between the programs will be dynamically balanced to share the available memory.

The **CacheMemory** property may be set to establish an upper limit on the size of the cache. If this property is not set, all available memory will be used. There are a number of read-only properties which enable caching to be displayed/monitored, such as **CacheMax** (the current maximum size of the cache) and **CacheAvail** (the current available memory of the computer).

Annotations Overview

The Viewport supports image Annotations which are drawn on top of an image in a non-destructive manner. Images annotations consist of **regions** which are grouped into **layers**. Each layer consists of a collection of regions which can be one of four types (poly-line, rectangle, ellipse, or arrow). Regions are defined by a list of vertices, which are used in various ways depending upon the region type (see below):

Region Type	Description
0-PolyLine	Any number of vertices defines a series of connected lines. The last vertex does <i>not</i> reconnect back to the first vertex.
1-Polygon	Any number of vertices defines a series of connected lines and the last vertex connects back to first vertex .
2-Ellipse	Two vertices define opposite corners of a bounding box in which an Ellipse is drawn.
4-Arrow	Two vertices define the endpoints of an arrow. The first is the arrow head, and the second is the tail.
5-Ruler	Two vertices define the endpoints of the ruler.

Each region of each layer is drawn on top of the image with the pen color you specify. Annotation layers and Regions can be programmatically selected. The selected Region of the selected annotation is drawn with an inner black line as well as the thick colored line.

Annotations and Regions can also have descriptive text associated with them. These text values are stores in collections of attributes (i.e. each region has a list of attributes, and each layer has a list of attributes). You can optionally populate and use the AttributeHeaders collection that belongs to each Regions collection. AttributeHeaders are useful for displaying attributes in a grid format. Each AttributeHeader object can define a column in the grid, and each region attribute can correspond to cell data.

The annotations for an image may be stored and retrieved as an XML string, using the GetAnnotationXML and PutAnnotationXML methods. For local images, the XML is stored in a file named **<image>.xml** (where <image> is the path to the image file, less its file extension), and for remote images the XML is stored on the server.

Following is the XML format used to store annotations (closing tags are omitted for brevity):

```
<Annotations>
  <Annotation      ReadOnly=rOnly      Visible=annVis      LineColor=lineCol
Selected=annSel>
    <ModifiedDate>modDateTime
    <Author>authorName
    <Attributes>
      <Attribute>
        <Name>attrName
        <Value>attrValue
    <Regions>
      <RegionAttributeHeaders>
        <AttributeHeader Id=headerId Name=headerName>
      <Region Type=regType Zoom=regZoom Selected=regSelected>
        <Attributes>
          <Attribute>
            <Name>headerId
            <Value>attrValue
        <Vertices>
          <Vertex X=verX Y=verY>
```

<i>readOnly</i>	[optional] Either 0 or 1, to indicate whether this layer can be edited in ImageScope. Algorithm results are usually read only.
<i>annVis</i>	[optional] Either 0 or 1, to indicate whether this layer will be painted on the image.
<i>lineCol</i>	[optional] The color suggested for outlining the annotation regions; the 32-bit color value 00RRGGBB is formatted as a decimal value.
<i>annSel</i>	[optional] Either 0 or 1, to indicate whether this layer is “selected.” Only one annotation layer may be selected at any time. The selected layer is typically highlighted in some way when displayed.
<i>modDateTime</i>	The date/time of creation or last modification of the annotation layer, given as “yyyy-mm-dd hh:mm:ss.ttt”
<i>authorName</i>	[optional] The person who created the annotation layer, or if created by a program, the name and version of the program.

<i>attrName</i>	Attribute name – any valid string.
<i>attrValue</i>	Attribute value – any valid string.
<i>headerId</i>	Integer giving index to attribute header (typically used as column index)
<i>headerName</i>	Name of attribute header
<i>regType</i>	Region type: 0=PolyLine, 1=Polygon, 2=Ellipse, 3=Arrow
<i>regZoom</i>	[optional] A decimal value specifying the zoom level associated with the region. 1.0 = baseline, 0.5 = 50% (zoomed out). Each region may be rendered at any zoom level, this value gives the zoom level used when the region was created.
<i>regSelected</i>	[optional] Either 0 or 1, to indicate whether this region is “selected.” Only one region may be selected within a layer at any time. The selected region is typically highlighted in some way when displayed.
<i>verX</i>	The X-coordinate of the vertex (across the image, with 0 at left)
<i>verY</i>	The Y-coordinate of the vertex (down the image, with 0 at top)

The Viewport control defines a number of objects to simplify management and manipulation of annotations and regions:

Annotation	- Object which encapsulations a single annotation layer
Annotations	- Collections object for Annotation objects
Region	- Object which encapsulates a single region within an annotation layer
Regions	- Collections object for Region objects
Attribute	- Object which encapsulates a single attribute (name=value) for an Annotation or Region
Attributes	- Collections object for Attribute objects
AttributeHeader	- Object which encapsulates a single attribute header for a Regions collection
AttributeHeaders	- Collections object for AttributeHeader objects

The following pages describe these objects in detail.

Annotations – Collection Object

Property	Type	Description
<i>Count</i>	Integer	The number of annotation objects in this collection.
<i>Dirty</i>	Bool	This property gets set to true whenever a changes is made to the annotations collection or any of its children.
<i>Item</i> (long Index)	Annotation	Returns the annotation object whose position in the list is specified by index.
<i>SelectedAnnotation</i>	Annotation	Returns the annotation that is currently “selected.” The selected region of the selected annotation is drawn with a black inner line.
<i>XML</i>	BSTR	Returns an XML string containing all annotations data (i.e., regions, vertices, attributes, attributeheaders, etc.). Getting this string is useful for saving annotations. Setting this string is useful for loading annotations.

Method Prototype	Description
void <i>Add</i> (Annotation Ann)	Adds an existing annotation to the collection.
Annotation <i>AddNew</i> ()	Creates a new annotation and adds it to the collection. The newly added annotation is returned.
void <i>ClearAnnotations</i> ()	Removes all annotations from the collection.
void <i>LoadAnnotationsFile</i> (BSTR fn)	Opens an XML annotations file specified by the string parameter.
void <i>Remove</i> (integer index)	Removes the annotation specified by index from the list.
void <i>SaveAnnotationsFile</i> (BSTR fn)	Saves the annotations to the specified XML file.

Annotation - Object

Property	Type	Description
<i>Attributes</i>	IAttributes	Returns/Sets the attributes collection object.
<i>Exported</i>	BOOL	Set true if an annotation has been exported.
<i>LineColor</i>	OLE_COLOR	Color used to draw the regions in this annotation layer.
<i>Regions</i>	IRegions	Returns/Sets the regions collection object.
<i>Type</i>	BSTR	Returns/Sets a string. Many Aperio apps use the integer values 3, 4, and 5 to indicate Analysis Result Annotation, Man-Made Annotation, and Z-Stack Annotation respectively.
<i>Visible</i>	Bool	Returns/Sets whether a layer will be painted on image
<i>ReadOnly</i>	Bool	Has no effect within viewport but used by ImageScope.
<i>NameReadOnly</i>	Bool	Has no effect within viewport but used by ImageScope.
<i>LineColorReadOnly</i>	Bool	Has no effect within viewport but used by ImageScope.
<i>Incremental</i>	Bool	Set to 1 if layer supports incremental AAF processing.
<i>MarkupImagePath</i>	BSTR	Returns/sets a string used by ImageScope for markup image location
<i>MacroName</i>	BSTR	Returns/sets the name of the macro that was run on this annotation. Valid for type 3 annotations only.

Method Prototype	Description
void <i>Select</i> ()	Makes this the selected annotation.

Regions - Collection Object

Property	Type	Description
<i>AttributeHeaders</i>	IAttributeHeaders	Get/Sets the attributeHeaders collection. AttributeHeaders can be useful for displaying regions and attributes in a grid (see Annotations Overview).
<i>Count</i>	Integer	The number of Region objects in this collection.
<i>Item</i> (long Index)	IRegion	Returns the Region object whose position in the list is specified by index.
<i>SelectedRegion</i>	IRegion	Returns the Region that is currently "selected." The selected region of the selected annotation is drawn with a black inner line.

Method Prototype	Description
void <i>Add</i> (IRegion Reg)	Adds an existing region to the collection.
IRegion <i>AddNew</i> ()	Creates a new region and adds it to the collection. The newly added region is returned.
void <i>Remove</i> (integer Index)	Removes the region specified by index from the list.

Region - Object

Property	Type	Description
<i>Attributes</i>	IAttributes	Gets/Sets an attributes collection object for this region.
<i>Bottom</i>	Long	Gets the bottom edge of this region's bounding box (in base level image pixels).
<i>HitTest</i> (Long X, Long Y)	Bool	Returns Boolean indicating whether or not the point specified by X and Y falls inside the region. (X,Y in base level image pixels).
<i>ImageLocation</i>	BSTR	Returns/sets the path to an image associated with this region. Used by ImageScope to locate Z-Stacks
<i>Left</i>	Long	Gets the left edge of this region's bounding box (in base level image pixels).
<i>Right</i>	Long	Gets the right edge of this region's bounding box (in base level image pixels).
<i>Top</i>	Long	Gets the top edge of this region's bounding box (in base level image pixels).
<i>Type</i>	Integer	Gets/Sets the region type (see Annotations Overview).
<i>VertexCount</i>	Long	Gets the total number of vertices in the region.
<i>VertexX</i> (long Index)	Long	Gets/Sets X value of the vertex at the specified index (in base level image pixels).
<i>VertexY</i> (long Index)	Long	Gets/Sets Y value of the vertex at the specified index (in base level image pixels).
<i>Analyze</i>	Bool	If True (or not present) AAF will analyze this region. If False, AAF will ignore this region when analyzing.
<i>NegativeROA</i>	Bool	If True, region is a negative ROA.

Method Prototype	Description
void <i>AddVertex</i> (Long X, Long Y)	Adds a vertex to the end of the list of vertices.
void <i>ClearVertices</i> ()	Clears the list of vertices.
void <i>Select</i> ()	Makes this region the selected region in its annotation layer.

Attributes - Collection Object

Property	Type	Description
<i>Count</i>	Integer	The number of attribute objects in this collection.
<i>Item</i> (long Index)	Attribute	Returns the attribute object whose position in the list is specified by index.

Method Prototype	Description
void <i>Add</i> (Attribute Att)	Adds an existing attribute to the collection.
Attribute <i>AddNew</i> ()	Creates a new attribute and adds it to the collection. The newly added attribute is returned.
void <i>Clear</i> ()	Removes all attributes from the collection.
void <i>Remove</i> (integer Index)	Removes the attribute specified by index from the collection.

Attribute - Object

Property	Type	Description
<i>Id</i>	BSTR	ID string for the attribute.
<i>Value</i>	BSTR	Value string for the attribute.
<i>NameReadOnly</i>	Bool	Has no effect within viewport but used by ImageScope.
<i>ValueReadOnly</i>	Bool	Has no effect within viewport but used by ImageScope.
<i>DisplayColor</i>	Long	Has no effect within viewport but used by ImageScope.

AttributeHeaders - Collection Object

Property	Type	Description
<i>Count</i>	Integer	The number of AttributeHeaders objects in this collection.
<i>Item</i> (long Index)	IAttributeHeader	Returns the AttributeHeader object whose position in the list is specified by index.

Method Prototype	Description
void <i>Add</i> (IAttributeHeader AttHdr)	Adds an existing AttributeHeader object to the collection.
IAttributeHeader <i>AddNew</i> ()	Creates a new AttributeHeader object and adds it to the collection. The newly added annotation is returned.
void <i>Clear</i> ()	Removes all AttributeHeader objects from the collection.
void <i>Remove</i> (integer index)	Removes the AttributeHeader object specified by index from the collection.

AttributeHeader - Object

Property	Type	Description
<i>Id</i>	BSTR	ID string for the AttributeHeader. Can be used for column id.
<i>Name</i>	BSTR	Name string for the AttributeHeader. Can be used for column caption.

Algorithms

The Viewport supports an interface to running algorithms server-side via Aperio's Algorithm Framework architecture. This capability is only available for "remote" image files.

Algorithm processing is accessed via the `RunAlgorithm()` and `GetAlgorithmStatus()` methods, and the `AlgorithmStatus` event (see API description above for details). Annotation layers are used to communicate input regions of analysis to algorithms, and to communicate output result regions from algorithms. Annotations may be accessed via the `GetAnnotationXML()` and `PutAnnotationXML()` methods, and manipulated using the Annotation objects (previous section, and described in the API above).

Algorithms are referenced by **macros**. A macro is a named set of algorithms and associated parameters. Algorithm execution is initiated with the `RunAlgorithm()` method, which creates an entry in a queue of algorithms to be run. The queue is "worked" by one or more computers running Aperio's Algorithm Framework. In the simplest configuration, the server itself is the computer which performs algorithm processing. Along with specifying the macro to be executed, a `RunAlgorithm()` request may specify an annotation layer for an image to be used as the input region of analysis (ROA). If no layer is specified, the entire virtual slide is processed. Annotation layers may be created and updated with the `PutAnnotations()` method.

As algorithms run they periodically update their progress and status on the server. These data may be queried with the `GetAlgorithmStatus()` method to display progress on the client side. Alternatively, the Viewport provides the `AlgorithmStatus` event, which is fired whenever status for a scheduled algorithm changes. When algorithm processing is complete the results are stored as a new annotation layer for the image. Annotations may be retrieved using the `GetAnnotationXML()` method.

**Aperio Viewport ActiveX Control (.OCX) Programmer's Reference
MAN-0072. Revision C**