AmpLibNet and COM Object Manual – V3.4.8.X

In older versions of AmpLib, .NET languages like Visual Basic and C# accessed the native DLL APIs through a COM object. That COM object only supported 32-bit .NET applications. Now that AmpLib64.dll is available, All My Papers now provides two .NET DLLs (AmpLibNet.dll and AmpLib64.dll) so that both 32 and 64-bit .NET applications can access the powerful AmpLib API. The methods and properties of both DLLs are largely compatible with the COM object interface. This manual describes these methods and properties in sufficient detail so that the demo programs provided in the SDK can be understood and expanded upon.

The VB and C# demo programs provided in the AmpLibNet SDK are intended to provide examples on how to use the core functionality of AmpLib with a minimal user interface. Before you start coding, it is recommended that the example program source code and project files be copied to a new folder so that your development efforts won't be hampered by the administrative restrictions of C:\Program Files.

The IampLib **IampImage**, **IampBarcode**, **and IampMICR** interfaces provide a powerful yet simple paradigm for manipulating images and reading barcodes without the need to know all of the data structures and APIs associated with the AMPLIB DLL. Technically speaking, these interfaces are a COM object for the AMPLIB DLL. They put a "friendly face" on the AMPLIB Application Program Interface (API) making it more convenient to use in programming environments such as C#, C++, Visual Basic, and Java that support COM objects.

This document is constructed as a series of hyperlinks to the various methods and properties of IampImage, IampBarcode, and IampMICR. Use the IDispatch tables for each in the following sections to quickly access the particular method or property of interest. Click on the Method or Property highlighted in blue to jump to the area in the document that has the information.

AMPLIB DLL has the ability to create and free unlimited image buffers of arbitrary dimensions using the ampCreateWorkImage, ampCreateGrayWorkImage, ampCreateColorWorkImage, ampFreeImage, and ampFreeAllImages API calls. The AMPLib COM object's IampImage interface removes the necessity of manual image buffer management by providing an internal workimage object and an array of 10 image buffers which are always allocated. Buffer index is specified as a parameter in API calls which use image buffers (see CopyImageToBuffer, LoadImageBuffer, PasteImageFromBuffer, RotateImageToBuffer, ScaleImageToBuffer). For example, CopyImageToBuffer copies the image contents of the main AmpLibNet image property to one of the 10 auxillary image buffers. Some AmpLibNet methods like ReadCamera use 2 image buffers for front/back image input and another 2 image buffers for front/back image output.

The AmpLibNet DLL is a .Net interface to the AMPLIB DLL which closely resembles the COM interface. It provides the methods and properties of the IampImage, IampBarcode, and IampMICR interfaces in a single object, and can be used in .Net languages like Visual Basic .NET and C#.

Appendix A contains a complete list of the error codes that are used within the AMPLib COM object as well as the AMPLIB DLL. Appendix B contains a mapping of how each AMPLib COM object method corresponds to a particular AMPLIB API function or group of functions. Appendix C contains a list of AMPLIB DLL calls which are not used by the AMPLib COM Object. Appendix D contains a description of AMPLib COM functionality which is not present in AMPLIB DLL. Appendix E is a discussion of which AMPLib code modules are needed for redistribution along with applications that reference IampLib.

In order to start developing your own application based on the AmpLibNet VB or CSharp demo projects, follow these steps:

1. Install the SDK
2. Copy the demo project folder out of the Program Files area
3. Launch VS2010 by double clicking on the .SLN file
4. Compile in debug mode
5. Copy all of the support files from the AmpLibNet bin folder to the debug folder created in step 4.
6. Start debugging the demo program.

IampImage: IDispatch

The **IampImage** interface provides a method for manipulating images.

Quick Info

Header file:	ampImage.h
Interface identifier:	IID_IampImage
Pointer type:	ampImage*

Vtable

Methods	Descriptions
AnnotateImage	Draws text on the image object.
CopyImageToBuffer	Copies the image object to an image buffer.
CountImages	Counts the number of images in a TIFF file.
DynamicThresholdGrayImage	Converts a grayscale image to a bilevel image.
FilterImage	Performs a filter operation on the image object.
GetAmplibVersion	Retrieves the current version of AMPLIB.DLL.
GetCOMVersion	Retrieves the COM object version.
GetLicenseInfo	Retrieves AMPLib license bit information

GetRunsInfo	Gets information on the image runlengths
GetMessage	Translates an error code into a text message.
GetScaledImageAddress	Creates a scaled bitmap of the image object.
GetWindow	Gets the current region of interest.
InterpolateGrayImage	Doubles the height and width of the image.
LoadBlankImage	Loads a blank image into the image object.
LoadClipboardImage	Loads the clipboard DIB into the image object.
LoadHBitmapImage	Loads an HBitmap into the image object.
LoadImage	Loads an image file into the image object.
LoadImageBuffer	Loads an image file from a memory buffer.
LoadImageFromMemory	Loads an image file from a memory buffer.
PasteImageFromBuffer	Pastes an image buffer into the image object.
ProcessGrayImage	Performs image processing on image object.
PromoteBilevelImage	Upgrades bilevel image object to grayscale.
RotateImageToBuffer	Creates a rotated version of the current image in an image buffer.
SaveImage	Saves the current image to disk.
SaveImageToClipboard	Saves the current image to the clipboard.
SaveImageToMemory	Saves the image to memory as G4 TIFF
SaveImageToMemoryTest	Saves the image to memory then to disk
ScaleImageToBuffer	Creates a scaled version of the current image in an image buffer.
SecurityEnableAppsFile	Activates OEM License from a file
SetWindow	Sets the region of interest.
ThresholdGrayImage	Thresholds gray image object to bilevel.

Properties	**Access**
AutoPrepEnable	Read/Write
debugTraceEnable	Read/Write
debugTraceFile	Read/Write
GrayImageEnable	Read/Write
imgHeight	Read only.
imgMaxHeight	Read only.
imgMaxWidth	Read only.
imgWidth	Read only.
pixelBitDepth	Read only.
traceEnable	Read/Write
traceFile	Read/Write
xResolution	Read/Write
yResolution	Read/Write

Remarks

The **IampImage** interface inherits directly from **IDispatch**.

1.2. IampImage::AnnotateImage

The **IampImage::AnnotateImage** method draws text within a stamp sized image that is then merged with the image object.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT AnnotateImage (** **BSTR** *outputText*, **BSTR** *fontStyle*, **BSTR** *fontOptions*, **LONG** *rectTop*, **LONG** *rectLeft*, **LONG** *rectRight*, **LONG** *rectBottom*, **LONG*** *ampResult* **);**
C#	**int AnnotateImage (** **string** *outputText*, **string** *fontStyle*, **string** *fontOptions*, **int** *rectTop*, **int** *rectLeft*, **int** *rectRight*, **int** *rectBottom*, **);**
C	**HRESULT IampImage_AnnotateImage (** **BSTR** *outputText*, **BSTR** *fontStyle*, **BSTR** *fontOptions*, **LONG** *rectTop*, **LONG** *rectLeft*, **LONG** *rectRight*, **LONG** *rectBottom*, **LONG*** *ampResult* **);**
JAVA	**int AnnotateImage (** **String** *outputText*, **String** *fontStyle*, **String** *fontOptions*, **int** *rectTop*, **int** *rectLeft*, **int** *rectRight*, **int** *rectBottom*, **);**
VB	**AnnotateImage (** *outputText* **as String,** *fontStyle* **as String,** *fontOptions* **as String,** *rectTop* **as Integer** *rectLeft* **as Integer** *rectRight* **as Integer** *rectBottom* **as Integer** **)** **as Integer;**

Parameters

outputText
The text that will be written on the image object.

fontStyle

The name of the font to be used when writing the text (e.g. Times New Roman).

fontOptions

Specifies options to be used when the font is being rasterized to the image object. Options are concatenated within the text string (e.g. TFQ=4). The supported font options are:

B Text written with reverse background

F Draws a frame around the annotated region

I Text is drawn italicized

J Specifies the justification: center J=C, left J=L, right, J=R

P Defines the pitch of the text in the annotation and is in the Following format:
P=0 - default
P=1 - variable
P=2 - fixed

O Defines the orientation or slant of the annotation text baseline and and is in the format "O=n" where n can be a positive or negative number in tenths of degrees. Positive numbers rotate the text counter-clockwise and negative numbers clockwise as in the following examples:
O=0 - default left-to-right horizontal text
O=300 – text rises up and to the left at 30 degrees
O=-900 – top-down vertical text

Q Specifies the rectangle corner the text is drawn in:
Q=0 - anywhere on the image
Q=1 - upper right
Q=2 - upper left
Q=3 - lower left
Q=4 - lower right

S Specifies the point size of the annotation text and is in the format "S=n". A 72 point font has upper case letters one inch tall.

T Text is written with transparent background

U Text is underlined

W Specifies the weight of the strokes used in the annotation text and is in the format "W=n". A value of 400 produces normal character stroke widths while smaller values are lighter/thinner and larger values are bolder/thicker. A value of 0 selects the current value which is typically normal.

X Specifies the starting horizontal pixel location in the annotation region where text will be begun in left justify mode. The default value for left justified text is 8. The center justify and right justify options automatically change this value as required.
X=8 – default value for left justified
X=64 – when O=-900, XY need to be moved away from the edge of the annotation stamp region

Y Specifies the starting vertical pixel location in the annotation region where text will be begun. The default value is 0 which is the top of the region.
Y=0 - default
Y=64 – when O=-900, XY needs to be moved away from the edge of the annotation stamp region

rectTop

This value along with the Q option specifies the vertical offset inward from the corner where the annotation stamp will be placed. Value is specified in units of 1/100 inch.

rectLeft

This value along with the Q option specifies the horizontal offset inward from the corner where the annotation stamp will be placed. Value is specified in units of 1/100 inch.

rectRight

This value specifies the width of the stamp region in which the outputText will be written. Value is specified in units of 1/100 inch.

rectBottom

This value specifies the height of the stamp region in which the outputText will be written. Value is specified in units of 1/100 inch.

ampResult

The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code. An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating variables within this method. An exception will be thrown (E_INVALIDARG) if the fontOptions parameters are incorrect.

Remarks

This method writes the specified text into a temporary image using the requested options and font. That temporary image is then merged on to the image object according to the Q parameter. If the text will not fit within the temporary region, it is clipped with no notification given if clipping occurs. Note that the requested annotation region size and position may be adjusted by the program to maintain pixel alignment rules. The size of the temporary image is specified in units of 1/100 inch so it is important that the resolution property of the image object is accurate.

See Also

1.3. IampImage::CopyImageToBuffer

The **IampImage::CopyImageToBuffer** method copies the image object to an image buffer.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT CopyImageToBuffer (** **LONG** *nBufferIndex*, **LONG*** *ampResult* **);**
C#	**int CopyImageToBuffer (** **int** *nBufferIndex* **);**
C	**HRESULT IampImage_ CopyImageToBuffer (** **LONG** *nBufferIndex*, **LONG*** *ampResult* **);**
JAVA	**int CopyImageToBuffer (** **int** *nBufferIndex* **);**
VB	**CopyImageToBuffer (** *nBufferIndex* **as Integer** **)** **as Integer;**

Parameters

nBufferIndex
The index number of the buffer into which the image object contents will be written. Valid indices are 0 through 9.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code. AMPLib error code 47 will be returned if the nBufferIndex parameter is outside of (0,9). An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating variables within this method.

Remarks

This method stores the contents of the image object in a buffer for future retrieval.

See Also

IampImage::PasteImageFromBuffer

1.4. IampImage::DynamicThresholdGrayImage

The **IampImage::DynamicThresholdGrayImage** method converts a grayscale image to a bilevel image.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT DynamicThresholdGrayImage (** **double** *dblPCS*, **LONG** *nAbsoluteBlackThreshold*, **LONG** *nHistogramEnable*, **LONG** *nLowPassEnable*, **LONG*** *ampResult* **);**
C#	**int DynamicThresholdGrayImage (** **double** *dblPCS*, **int** *nAbsoluteBlackThreshold*, **int** *nHistogramEnable*, **int** *nLowPassEnable* **);**
C	**HRESULT IampImage_DynamicThresholdGrayImage (** **double** *dblPCS*, **LONG** *nAbsoluteBlackThreshold*, **LONG** *nHistogramEnable*, **LONG** *nLowPassEnable*, **LONG*** *ampResult* **);**
JAVA	**int DynamicThresholdGrayImage (** **double** *dblPCS*, **int** *nAbsoluteBlackThreshold*, **int** *nHistogramEnable*, **int** *nLowPassEnable* **);**
VB	**DynamicThresholdGrayImage (** *dblPCS* **as Double,** *nAbsoluteBlackThreshold* **as Integer,** *nHistogramEnable* **as Integer,** *nLowPassEnable* **as Integer** **)** **as Integer;**

Parameters

dblPCS
The contrast ratio threshold (range: 0-1). If dblPCS is 0.0 then the system will use the default threshold of 0.15, which is known to produce optimal bilevel images for CAR/LAR recognition. Use a lower setting to darken the image, or a higher setting to lighten the image.

nAbsoluteBlackThreshold
The absolute black threshold. Any grayscale pixel below this threshold will be converted to a black pixel in the binary image. The threshold is set to 55 if nAbsoluteBlackThreshold = 0.

nHistogramEnable

If nHistogramEnable is set to 1, the system will determine the optimal threshold curve based on histogram analysis. Use nHistogramEnable = 0 only if you know that the images have good dynamic range for contrast. The histogram analysis will require extra processing time. Good quality check scanners produce images with good dynamic range. Use nHistogramEnable for page scanner or unknown scanning devices.

nLowPassEnable

If nLowPassEnable = 1, the system will filter out high frequency noise in the grayscale image producing a very clean image with a low compressed file size. Set nLowPassEnable = 0 if this operation is not desired.

Return Values

If the function succeeds, the return value is zero. If the image is already bilevel, the method will exit promptly with success. An exception will be thrown (e_NoImage) if this method is called before an image is loaded.

Remarks

This method thresholds the 8-bit grayscale workimage property to bilevel. Input parameters are used to adjust the algorithm for optimal results on different classes of source grayscale images.

This system requires a high quality grayscale image (> 80 DPI and 256 levels of gray). The function will return error code 202 if the quality of the grayscale image is below these levels.

The pitch of the bilevel image will be increased to ensure that the image rows end on a byte boundary with white pixels added for padding.

The dynamic threshold compares the grayscale value of a centre pixel to the average grayscale of the surrounding area (1/8 sq in.). If the grayscale value is below the PCS threshold then it is turned black. This produces very good binary images on scenic background checks.

See Also

IampImage::ThresholdGrayImage

1.5. IampImage::AutoPrepEnable

The **IampImage::AutoPrepEnable** property enhances the performance of LoadImage when loading MICR images.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT put_AutoPrepEnable (** **BOOL** *newVal* **);** **HRESULT get_AutoPrepEnable (** **BOOL *** *pVal* **);**
C#	**int AutoPrepEnable (** **boolean** *newVal* **};** **boolean AutoPrepEnable**
C	**HRESULT Iax9Lib_put_AutoPrepEnable (** **BOOL** *newVal* **);** **HRESULT Iax9Lib_get_AutoPrepEnable (** **BOOL *** *pVal* **);**
JAVA	**void set_AutoPrepEnable (** **boolean** *newVal* **);** **void get_AutoPrepEnable (** **boolean*** *pVal* **);**

VB	**AutoPrepEnable (** *NewVal* **As Boolean** **) as Integer;** **AutoPrepEnable () as Boolean;**

Parameters

newVal, pVal
A flag that turns on or off AutoPrepEnable.

Remarks

LoadImage checks the value of AutoPrepEnable and if true, assumes that a check image is being loaded and proceeds through several combinations of image processing, thresholding, and MICR reading operations with different parameters each time. Once an optimal set of parameters are determined, the check image will be loaded into the image object and scaled to 200 dpi. AutoPrep is best used on grayscale images of unknown resolution.

See IampImage::LoadImage

1.6. IampImage::CountImages

The **IampImage::CountImages** method returns the number of images in a multi-page TIFF image file.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT CountImages (** **BSTR** *fileName*, **BSTR** *options*, **BSTR** *fileType*, **LONG*** *pnImageCount*, **LONG*** *ampResult* **);**
C#	**int CountImages (** **string** *fileName*, **string** *options*, **string** *fileType*, **int** *pnImageCount*, **);**
C	**HRESULT IampImage_CountImages (** **BSTR** *fileName*, **BSTR** *options*, **BSTR** *fileType*, **LONG*** *pnImageCount*, **LONG*** *ampResult* **);**
JAVA	**int CountImages (** **String** *fileName*, **String** *options*, **String** *fileType*, **int** *pnImageCount*, **);**

VB	**CountImages (** *fileName* **as String,** *options* **as String,** *fileType* **as String,** *pnImageCount* **as Integer** **)** **as Integer;**

Parameters

fileName
The name of the file that will be loaded.

options
A string of characters that describe optional processing for the image data as it is loaded.

B Bit-byte-reverses the image data stream.
T Allows color components to be suppressed when loading color TIFF and JPEG files.
T=-1 Suppress red
T=-2 Suppress green
T=-3 Suppress blue

fileType
Specifies the file organization. It is only needed when the type cannot be determined by the file extension or by reading the header information in the file. The supported file types are:

TIFF	TIFF file, including ViewStar TIFF format
PCX	PCX
DCX	Multi-page PCX
PDF	Adobe®-compatible PDF (output only)
NOHEADER	Data only, no header
BMP	Bitmap
JPG	JPEG

pnImageCount
Returns the number of images in the multi-image source TIFF file.

ampResult
The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code. An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating variables within this method. An exception will be thrown (e_FileNotFound) if this method cannot find the file to load. An exception will be thrown (E_INVALIDARG) if the options, fileType or imageIndex parameters incorrect.

Remarks

This method return the number of individual image pages in a multi-image TIFF source file.

See Also

1.7. IampImage::debugTraceEnable

The **IampImage::debugTraceEnable** property determines whether or not the AMPLIB COM Object debug trace feature is enabled.

Quick Info

See IampImage: IDispatch.

C++	**HRESULT put_debugTraceEnable (** **int** *debugTraceEnable* **);** **HRESULT get_debugTraceEnable (** **int*** *debugTraceEnable* **);**
C#	**int** *debugTraceEnable*
C	**HRESULT IampImage_put_debugTraceEnable (** **int*** *debugTraceEnable* **);** **HRESULT IampImage_get_debugTraceEnable (** **int*** *debugTraceEnable* **);**
JAVA	**void put_debugTraceEnable (** **int*** *debugTraceEnable* **);** **void get_debugTraceEnable (** **int*** *debugTraceEnable* **);**
VB	**debugTraceEnable as Integer;**

Parameters

debugTraceEnable
An integer value. If set to 1, the debug trace feature will be enabled. If set to 0, the debug trace feature will be disabled.

Remarks

The AMPLIB COM Object debug trace feature records a log of significant events to a text file. This log may be useful to All My Papers Customer Support during troubleshooting.

See Also

IampImage::debugTraceFile

1.8. IampImage::debugTraceFile

The **IampImage::debugTraceFile** property contains the destination filename of the AMPLIB COM Object debug trace file.

Quick Info

See IampImage: IDispatch.

C++	**HRESULT put_debugTraceFile (** **BSTR** *debugTraceFile* **);** **HRESULT get_debugTraceFile (** **BSTR*** *debugTraceFile* **);**
C#	**string** *debugTraceFile*
C	**HRESULT IampImage_put_debugTraceFile (** **BSTR** *debugTraceFile* **);** **HRESULT IampImage_get_debugTraceFile (** **BSTR*** *debugTraceFile* **);**

JAVA	**void put_debugTraceFile (** **String** *debugTraceFile* **);** **void get_debugTraceFile (** **String** *debugTraceFile* **);**
VB	**debugTraceFile as String;**

Parameters

debugTraceFile
A string that contains the path to the debug trace text file.

Remarks

The AMPLIB COM Object debug trace feature records a log of significant events to a text file. This log may be useful to All My Papers Customer Support during troubleshooting.

See Also

IampImage::debugTraceEnable

1.9. IampImage::FilterImage

The **IampImage::FilterImage** method performs image processing on the image object.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT FilterImage (** **BSTR** *filterStyle*, **BSTR** *filterSubcode*, **LONG** *threshold*, **LONG*** *ampResult* **);**
C#	**int FilterImage (** **string** *filterStyle*, **string** *filterSubcode*, **int** *threshold*, **);**
C	**HRESULT IampImage_LoadImage (** **BSTR** *filterStyle*, **BSTR** *filterSubcode*, **LONG** *threshold*, **LONG*** *ampResult* **);**
JAVA	**int LoadImage (** **String** *filterStyle*, **String** *filterSubcode*, **int** *threshold*, **);**

VB	**LoadImage (** *filterStyle* **as String,** *filterSubcode* **as String,** *threshold* **as Integer** **)** **as Integer;**

Parameters

filterStyle

Defines the class of image processing filter operation to perform. The following filters are available:

majority	A directional majority filter that can preserve certain structures within the image.
erode	An erosion filter useful for thinning image elements
dilate	A dilation filter useful for fattening image elements.
spot	A spot removal (de-speckel) filter.

For erode, dilate, and spot filters, the *filterSubcode* can be given as '**weak**' (4-neighbor) or '**strong**' (8-neighbor) to control the strength of the effect.

4-neighbor (or 4-connected) means the adjacent pixels to the North, East, South, and West. 8-neighbor, (or 8-connected), means any adjacent pixel in a surrounding 3x3 box.

For the majority filter, a variety of *filterSubcode* values can be used.

Note: the majority filters, which control preservation of lines, act only on single pixel width lines. These filters can be useful in removing fine line screens, as appear in some negotiable documents, but they are *not* generalized line removal functions.

filterSubcode

A string of characters that describe a numeric (e.g. “70” or “201”) or character-based (e.g. “WEAK” or “STRONG”) modifier to the basic filterStyle.

Filter Type	Sub Code	Definition
ERODE	WEAK	4- neighbor erosion. Can be used to lighted lines. If any of 4-connected neighbors are white, the pixel will be set white. This filter can remove single pixel lines.
ERODE	STRONG	8- neighbor erosion. A more aggressive filter than 4-neighbor, this will set the pixel white if any of the eight-connected neighbors are white.
DILATE	WEAK	4- neighbor dilation. This filter darkens an object by setting a pixel black if any of its 4-connected neighbors are black.
DILATE	STRONG	8- neighbor dilation The filter darkens an object by setting a pixel black if any of its 8 neighbors are black.
SPOT	WEAK	4x4 neighbor spot removal. A more aggressive spot removal filter. It will remove any black pixels in a 2x2 region if all the pixels in the surrounding 4x4 region are white.

SPOT	STRONG	6x6- neighbor spot removal. An even more aggressive spot removal filter. It will remove any black pixels in a 2x2 region if all the pixels in the surrounding 6x6 region are white.
SPOT	SINGLE	Single pixel spot removal. If all 8 neighbors of a pixel are white, the pixel is set to white.
MAJORITY	NORMAL	Standard majority filter. The number of black pixels in a 3x3 region is compared to a threshold. If the count >= the threshold, sets the center pixel to black, else sets the center pixel to white. Preserves features in image.
MAJORITY	ERODE	A weighted erosion filter. The number of white pixels in a 3x3 region is compared to a threshold. If the count >= the threshold, sets the center pixel to white. Preserves features in image.
MAJORITY	DILATE	A weighted dilation filter. The number of black pixels in a 3x3 region is compared to a threshold. If the count >= the threshold, sets the center pixel to black. Preserves features in image.
MAJORITY	SPUR	Removes single-pixel growths from vertical line edges
MAJORITY	BUMP1	Removes one pixel growths from vertical line edges
MAJORITY	BUMP2	Removes two pixel growths from vertical line edges
MAJORITY	NORMAL_NP	A weighted filter that does not try to preserve single pixel lines in horizontal, vertical, and diagonal directions.
MAJORITY	NORMAL_NPH	Like NORMAL, except that it preserves all but horizontal lines.
MAJORITY	NORMAL_NPV	Like NORMAL, except that it preserves all but vertical lines.
MAJORITY	NORMAL_NPD	Like NORMAL, except that it preserves all but diagonal lines.
MAJORITY	NORMAL_NPNE	Like NORMAL, except that it preserves all but NorthEast to southwest diagonal lines.
MAJORITY	NORMAL_NPNW	Like NORMAL, except that it preserves all but Northwest to Southeast diagonal lines.
MAJORITY	ERODE_NPH	Like ERODE, except that it preserves all but horizontal lines.
MAJORITY	ERODE_NPV	Like ERODE, except that it preserves all but vertical lines.
MAJORITY	ERODE_NPD	Like ERODE, except that it preserves all but diagonal lines.
MAJORITY	ERODE_NPNE	Like ERODE, except that it preserves all but Northeast to Southwest diagonal lines.
MAJORITY	ERODE_NPNW	Like ERODE, except that it preserves all but Northwest to Southeast diagonal lines.

MAJORITY	DILATE_NPH	Like DILATE, except that it preserves all but horizontal lines.
MAJORITY	DILATE_NPV	Like DILATE, except that it preserves all but vertical lines.
MAJORITY	DILATE_NPD	Like DILATE, except that it preserves all but diagonal lines.
MAJORITY	DILATE_NPNE	Like DILATE, except that it preserves all but Northeast to Southwest diagonal lines.
MAJORITY	DILATE_NPNW	Like DILATE, except that it preserves all but Northwest to Southeast diagonal lines.

threshold
This value is used for the majority filter, and ranges from 0 to 9. A threshold of 5 is considered neutral. Lower values will give darker looking images; higher values will tend to lighten the image.

ampResult
The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code. An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating variables within this method. An exception will be thrown (E_INVALIDARG) if the options, filterStyle or filterSubcode parameters incorrect.

Remarks

This method loads an image file into the image object.

See Also

IampImage::LoadImage

1.10. IampImage::GetAmplibVersion

The **IampImage::GetAmplibVersion** method gets the current version of AMPLIB.DLL.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT GetAmplibVersion (** **LONG*** *major1*, **LONG*** *major2*, **LONG*** *minor1*, **LONG*** *minor2*, **LONG*** *ampResult* **);**
C#	**int GetAmplibVersion (** **out int** *major1*, **out int** *major2*, **out int** *minor1*, **out int** *minor2*, **);**

C	**HRESULT IampImage_GetAmplibVersion (** **LONG*** *major1*, **LONG*** *major2*, **LONG*** *minor1*, **LONG*** *minor2*, **LONG*** *ampResult* **);**
JAVA	**int GetAmplibVersion (** **int** *major1*, **int** *major2*, **int** *minor1*, **int** *minor2*, **);**
VB	**GetAmplibVersion (** *major1* **as Integer,** *major2* **as Integer,** *minor1* **as Integer,** *minor2* **as Integer** **)** **as Integer;**

Parameters

major1
Contains the left most digit of the AMPLIB.DLL version number (e.g. 6 of 6.1.1.4).

major2
Contains the second most significant digit of the version number (e.g. 1 of 6.1.1.4).

minor1
Contains the third most significant digit of the version number (e.g. 1 of 6.1.1.4).

minor2
Contains the rightmost digit in the version number (e.g. 4 of 6.1.1.4).

ampResult
The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

This method returns the version number associated with the AMPLIB.DLL that the COM object is currently connected to in the format major1.major2.minor1.minor2 (e.g. 6.1.1.4).

1.11. IampImage::GetCOMVersion

The **IampImage::GetCOMVersion** method returns the version of this COM object DLL.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT GetVersion (** **BSTR*** *strVersion*, **LONG*** *major1*, **LONG*** *major2*, **LONG*** *minor1*,

	LONG* *minor2*, **LONG*** *ampResult* **);**
C#	**int GetVersion (** **out string** *strVersion*, **out int** *major1*, **out int** *major2*, **out int** *minor1*, **out int** *minor2* **);**
C	**HRESULT IampImage _GetVersion (** **BSTR*** *strVersion*, **LONG*** *major1*, **LONG*** *major2*, **LONG*** *minor1*, **LONG*** *minor2*, **LONG*** *ampResult* **);**
JAVA	**int GetVersion (** **string** *strVersion*, **int** *major1*, **int** *major2*, **int** *minor1*, **int** *minor2* **);**
VB	**GetVersion (** *strVersion* **as String,** *major1* **as Integer,** *major2* **as Integer,** *minor1* **as Integer,** *minor2* **as Integer** **)** **as Integer;**

Parameters

strVersion
The returned version information as character data.

major1
The returned first major number.

major2
The returned second major number.

minor1
The returned first minor number.

minor2
The returned second minor number.

ax9Result
The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code. An exception will be thrown (E_OUTOFMEMORY) if the system runs out of memory while allocating variables within this method.

Remarks

This method returns the version level and sub levels of the COM object DLL. The strVersion value contains the formatted name and revision level information. The four integer variables return the Major and Minor levels of the DLL.

See Also

1.12. IampImage::GetLicenseInfo

The **IampImage::GetLicenseInfo** method returns AMPLib licensing information residing on the current PC.

Quick Info

See Iax9Lib: IDispatch.

C++	**HRESULT GetLicenseInfo (** **BSTR*** *strExpDate,* **LONG*** *AMPLibLicenseBits,* **LONG*** *ampResult* **);**
C#	**int GetLicenseInfo (** **out string** *strExpDate,* **out int** *AMPLibLicenseBits,* **);**
C	**HRESULT IampImage_ GetLicenseInfo (** **BSTR*** *strExpDate,* **LONG*** *AMPLibLicenseBits,* **LONG*** *ampResult* **);**
JAVA	**int GetLicenseInfo (** **string** *strExpDate,* **int** *AMPLibLicenseBits,* **);**
VB	**GetLicenseInfo (** *strExpDate* **as String,** *AMPLibLicenseBits* **as Integer,** **)** **as Integer**

Parameters

strExpDate
The returned license expiration date of AMPLib in mm/dd/yy format. Returned strings can also include “Does not expire” based on the current license.

AMPLibLicensingBits
The returned 32 AMPLib licensing bits from the registry.

ampResult
The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code. An exception will be thrown (E_OUTOFMEMORY) if the system runs out of memory while allocating variables within this method.

Remarks

This method returns the AMPLib expiration date and various licensing bits associated with AMPLib from the registry.

See Also

1.13. IampImage::GetRunsInfo

The **IampImage::GetRunsInfo** method returns runlength information on the current work image property. If the work image is gray enabled, a temporary bilevel version of the image is created for runlength analysis.

Quick Info

See Iax9Lib: IDispatch.

C++	**HRESULT GetRunsInfo (** **LONG** *nMinWidth*, **LONG** *nMaxWidth*, **LONG*** *nTotalBlackPixels* **LONG*** *nTotalBlackRuns* **LONG*** *ampResult* **);**
C#	**int GetRunsInfo (** **int** *nMinWidth*, **int** *nMaxWidth*, **out int** *nTotalBlackPixels*, **out int** *nTotalBlackRuns*, **);**
C	**HRESULT IampImage_ GetRunsInfo (** **LONG** *nMinWidth*, **LONG** *nMaxWidth*, **LONG*** *nTotalBlackPixels* **LONG*** *nTotalBlackRuns* **LONG*** *ampResult* **);**
JAVA	**int GetRunsInfo (** **int** *nMinWidth*, **int** *nMaxWidth*, **int** *nTotalBlackPixels*, **int** *nTotalBlackRuns*, **);**
VB	**GetRunsInfo (** *nMinWidth* **as Integer**, *nMaxWidth* **as Integer**, *nTotalBlackPixels* **as Integer**, *nTotalBlackRuns* **as Integer**, **)** **as Integer**

Parameters

nMinWidth
The minimum length run that will be analyzed.

nMaxWidth
The maximum length run that will be analyzed.

nTotalBlackPixels
The returned total number of black pixels in the image that are a part of runlengths that vary in size from nMinWidth to nMaxWidth.

nTotalBlackRuns
The returned total number of runlengths in the image that vary in size from nMinWidth to nMaxWidth.

ampResult
The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code. An exception will be thrown (E_OUTOFMEMORY) if the system runs out of memory while allocating variables within this method.

Remarks

This method analyzes the black runlengths in the current work image that vary in size between nMinWidth and nMaxWidth. The total number of pixels in these runs is returned along with the total number of runs. If the work image is gray enabled, a temporary bilevel version of the image is created prior to the runlength analysis.

2.

See Also

2.2. IampImage::GetMessage

The **IampImage::GetMessage** translates an error code into an appropriate text string.

Quick Info

See IampImage: IDispatch.

C++	**HRESULT GetMessage (** **LONG** *rcVal,* **BSTR*** *strMessage,* **LONG*** *ampResult* **);**
C#	**int GetMessage (** **int** *rcVal,* **out string** *strMessage,* **);**
C	**HRESULT Iax9Lib_ GetMessage (** **LONG** *rcVal,* **BSTR*** *strMessage,* **LONG*** *ampResult* **);**
JAVA	**int GetMessage (** **int** *rcVal,* **string** *strMessage,* **);**
VB	**GetMessage (** *rcVal* **as Integer,** *strMessage* **as String,** **)** **as Integer**

Parameters

rcVal
The input return code value.

strMessage
The returned message that corresponds to the input AMPLib return code value.

ampResult
The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code. An exception will be thrown (E_OUTOFMEMORY) if the system runs out of memory while allocating variables within this method.

Remarks

This method returns an error code string that matches the input integer value. For example, if the input value is 107, the value passed back through strMessage will be “License required for this feature.”

See Also

2.3. IampImage::GetWindow

The **IampImage::GetWindow** method gets the current region of interest.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT GetWindow (** **LONG*** *left*, **LONG*** *top*, **LONG*** *width*, **LONG*** *height*, **LONG*** *ampResult* **);**
C#	**int GetWindow (** **out int** *left*, **out int** *top*, **out int** *width*, **out int** *height*, **);**
C	**HRESULT IampImage_GetWindow (** **LONG*** *left*, **LONG*** *top*, **LONG*** *width*, **LONG*** *height*, **LONG*** *ampResult* **);**
JAVA	**int GetWindow (** **int** *left*, **int** *top*, **int** *width*, **int** *height*, **);**
VB	**GetWindow (** *left* **as Integer,** *top* **as Integer,** *width* **as Integer,** *height* **as Integer** **)** **as Integer;**

Parameters

left

Contains the number of pixels for the left margin of the region.

top
Contains the number of pixels for the top margin of the region.

width
Contains the number of pixels for the horizontal length of the region.

height
Contains the number of pixels for the vertical length of the region.

ampResult
The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

This method gets the region of interest. An exception will be thrown (e_NoImage) if this method is called before an image is loaded.

See Also

IampImage::SetWindow

2.4. IampImage::GetScaledImageAddress

The **IampImage::GetScaledImageAddress** method creates a new working image in the AMPLib COM Object that can be used in a .NET PictureBox.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT GetAmplibVersion (** **LONG** *nWidth*, **LONG** *nHeight*, **LONG** *nFormat*, **LONG*** *pdwBytePitch*, **LONG*** *pdwAddress*, **LONG*** *ampResult* **);**
C#	**int GetAmplibVersion (** **int** *nWidth*, **int** *nHeight*, **int** *nFormat*, **out int** *pdwBytePitch*, **out int** *pdwAddress* **);**
C	**HRESULT IampImage_GetAmplibVersion (** **LONG** *nWidth*, **LONG** *nHeight*, **LONG** *nFormat*, **LONG*** *pdwBytePitch*, **LONG*** *pdwAddress*, **LONG*** *ampResult* **);**

JAVA	**int GetAmplibVersion (** **Integer** *nWidth*, **Integer** *nHeight*, **Integer** *nFormat*, **Integer*** *pdwBytePitch*, **Integer*** *pdwAddress* **);**
VB	**GetAmplibVersion (** *nWidth* **as Integer,** *nHeight* **as Integer,** *nFormat* **as Integer,** *pdwBytePitch* **as Integer,** *pdwAddress* **as Integer** **)** **as Integer;**

Parameters

nWidth
Sets the width, in pixels, of the result image.

nHeight
Sets the height, in pixels, of the result image.

nFormat
0 sets the image to bilevel (black & white) format.
1 sets the image to bilevel format with colors inverted.

pdwBytePitch
Contains the number of bytes per raster in the scaled image.

pdwAddress
Contains the address of the beginning of the image data.

ampResult
The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

This method creates a scaled bitmap of the AMPLib image object designed to be used in a .NET PictureBox, and provides information necessary to use it.

2.5. IampImage::GrayImageEnable

The **IampImage::GrayImageEnable** property allows the workimage property to contain grayscale images.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT put_GrayImageEnable (** **BOOL** *newVal* **);** **HRESULT get_GrayImageEnable (** **BOOL *** *pVal* **);**

C#	**int GrayImageEnable (** **boolean** *newVal* **};** **boolean AutoPrepEnable**
C	**HRESULT IampImage_put_GrayImageEnable (** **BOOL** *newVal* **);** **HRESULT IampImage_get_GrayImageEnable (** **BOOL *** *pVal* **);**
JAVA	**void set_GrayImageEnable (** **boolean** *newVal* **);** **void get_GrayImageEnable (** **boolean*** *pVal* **);**
VB	**GrayImageEnable (** *NewVal* **As Boolean** **) as Integer;** **GrayImageEnable () as Boolean;**

Parameters

newVal, pVal
A flag that turns on or off AutoPrepEnable.

Remarks

LoadImage checks the value of GrayImageEnable and if true, changes the workimage property to have a grayscale attribute before the image is loaded. If the image loaded is bilevel, the workimage property will be bilevel. If the image loaded is grayscale or color, then the workimage property will be 8-bit grayscale. The method GetPixelBitDepth can be used after LoadImage to determine whether the image was grayscale or bilevel.

See IampImage::LoadImage

2.6. IampImage::imgHeight

The **IampImage::imgHeight** property is the height of the loaded image.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT get_imgHeight (** **LONG*** *pVal* **);**
C#	**Long imgHeight**
C	**HRESULT IampImage_get_imgHeight (** **LONG*** p*Val* **);**
JAVA	**void get_imgHeight (** **int*** *pVal* **);**
VB	**imgHeight () as Long;**

Parameters

pVal
A pointer to a variable that will receive the value.

Remarks

The height of the image remains unchanged until a new image is loaded or the SetWindow method is called. This value is the same as the height parameter returned with GetWindow. An exception will be thrown (e_NoImage) if this property is accessed before an image is loaded.

See Also

IampImage::imgWidth, IampImage::GetWindow, IampImage::SetWindow

2.7. IampImage::imgMaxHeight

The **IampImage::imgMaxHeight** property is the maximum height of the loaded image.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT get_imgMaxHeight (** **LONG*** *pVal* **);**
C#	**Long imgMaxHeight**
C	**HRESULT IampImage_get_imgMaxHeight (** **LONG*** p*Val* **);**
JAVA	**void get_imgMaxHeight (** **int*** *pVal* **);**
VB	**imgMaxHeight () as Long;**

Parameters

pVal
A pointer to a variable that will receive the value.

Remarks

The maximum height of the image is established when a new image is loaded. The SetWindow method does not change the maximum height. An exception will be thrown (e_NoImage) if this property is accessed before an image is loaded.

See Also

IampImage::LoadImage

2.8. IampImage::imgMaxWidth

The **IampImage::imgMaxWidth** property is the maximum width of the loaded image.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT get_imgMaxWidth (** **LONG*** *pVal* **);**

C#	**Long imgMaxWidth**
C	**HRESULT IampImage_get_imgMaxWidth (** **LONG*** p*Val* **);**
JAVA	**void get_imgMaxWidth (** **int*** *pVal* **);**
VB	**imgMaxWidth () as Long;**

Parameters

pVal
A pointer to a variable that will receive the value.

Remarks

The maximum width of the image is established when a new image is loaded. The SetWindow method does not change the maximum width. An exception will be thrown (e_NoImage) if this property is accessed before an image is loaded.

See Also

IampImage::LoadImage

2.9. IampImage::imgWidth

The **IampImage::imgWidth** property is the width of the loaded image.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT get_imgWidth (** **LONG*** *pVal* **);**
C#	**Long imgWidth**
C	**HRESULT IampImage_get_imgWidth (** **LONG*** *pVal* **);**
JAVA	**void get_imgWidth (** **int*** *pVal* **);**
VB	**imgWidth () as Long;**

Parameters

pVal
A pointer to a variable that will receive the value.

Remarks

The width of the image remains unchanged until a new image is loaded. This value is the same as the width parameter returned with GetWindow. An exception will be thrown (e_NoImage) if this property is accessed before an image is loaded.

See Also

IampImage::imgHeight, IampImage::GetWindow, IampImage::SetWindow

2.10. IampImage::InterpolateGrayImage

The **IampImage::InterpolateGrayImage** method doubles the height and width of the grayscale workimage property.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT InterpolateGrayImage (** **LONG*** *ampResult* **);**
C#	**int InterpolateGrayImage (** **);**
C	**HRESULT IampImage_InterpolateGrayImage (** **LONG*** *ampResult* **);**
JAVA	**int InterpolateGrayImage (** **);**
VB	**InterpolateGrayImage (** **)** **as Integer;**

Parameters

ampResult
The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

This method method doubles the height and width of the workimage property filling in the new pixels with the average value of neighboring pixels. If the workimage was bilevel, it will be promoted to grayscale before the interpolation operation. An exception will be thrown (e_NoImage) if this method is called before an image is loaded.

See Also

2.11. IampImage::LoadBlankImage

The **IampImage::LoadBlankImage** method fills the image object with a blank (white) bitmap.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT LoadBlankImage (** **LONG** *width*, **LONG** *height*, **LONG*** *ampResult* **);**
C#	**int LoadBlankImage (** **int** *width*, **int** *height*, **);**
C	**HRESULT IampImage_LoadBlankImage (** **LONG** *width*, **LONG** *height*, **LONG*** *ampResult* **);**
JAVA	**int LoadBlankImage (** **int** *width*, **int** *height*, **);**
VB	**LoadBlankImage (** *width* **as Integer,** *height* **as Integer** **)** **as Integer;**

Parameters

width
Specifies the number of pixels for the horizontal width of the blank image. If this value is less than 32 pixels the width and height will be forced to 1200 and 600.

height
Specifies the number of pixels for the vertical height of the blank image. If this value is less than 32 pixels the width and height will be forced to 1200 and 600.

ampResult
The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

This method erases the current image and replaces it with a blank image (white) of the specified size.

See Also

2.12. IampImage::LoadClipboardImage

The **IampImage::LoadClipboardImage** method loads the image property with the contents of the clipboard.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT LoadClipboardImage (** **LONG** *hWnd*, **LONG*** *ampResult* **);**
C#	**int LoadClipboardImage (** **int** *hWnd*, **);**
C	**HRESULT IampImage_ LoadClipboardImage (** **LONG** *hWnd*, **LONG*** *ampResult* **);**
JAVA	**int LoadClipboardImage (** **int** *hWnd*, **);**
VB	**LoadClipboardImage (** *hWnd* **as Integer** **)** **as Integer;**

Parameters

hWnd
Specifies the windows handle for the current application.

ampResult
The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

This method loads the image property with any DIB image residing in the Windows Clipboard.

See Also

IampImage::LoadImage

2.13. IampImage::LoadHBitmapImage

The **IampImage::LoadHBitmapImage** method loads the image property with a Windows HBitmap.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT LoadHBitmapImage (** **LONG** *hBitmap*, **LONG** *nStyle*, **LONG** *nThreshold*, **LONG*** *ampResult* **);**

C#	**int LoadHBitmapImage (** **int** *hBitmap*, **int** *nStyle*, **int** *nThreshold* **);**
C	**HRESULT IampImage_LoadHBitmapImage (** **LONG** *hBitmap*, **LONG** *nStyle*, **LONG** *nThreshold*, **LONG*** *ampResult* **);**
JAVA	**int LoadHBitmapImage (** **int** *hBitmap*, **int** *nStyle*, **int** *nThreshold* **);**
VB	**LoadHBitmapImage (** *hBitmap* **as IntPtr,** *nStyle* **as Integer,** *nThreshold* **as Integer** **)** **as Integer;**

Parameters

hBitmap
Specifies the windows HBitmap to use.

nStyle
This should always be set to 0.

nThreshold
This should always be set to 0.

ampResult
The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

This method loads the image property with the contents of a windows HBitmap. The HBitmap to use is specified by a windows handle, such as the return value of the .NET System.Drawing.Bitmap.GetHbitmap() call.

See Also

IampImage::LoadImage

2.14. IampImage::LoadImage

The **IampImage::LoadImage** method loads an image file from disk into the image object.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT LoadImage (** **BSTR** *fileName*, **BSTR** *options*, **BSTR** *fileType*, **LONG** *imageIndex*, **LONG*** *ampResult* **);**
C#	**int LoadImage (** **string** *fileName*, **string** *options*, **string** *fileType*, **int** *imageIndex*, **);**
C	**HRESULT IampImage_LoadImage (** **BSTR** *fileName*, **BSTR** *options*, **BSTR** *fileType*, **LONG** *imageIndex*, **LONG*** *ampResult* **);**
JAVA	**int LoadImage (** **String** *fileName*, **String** *options*, **String** *fileType*, **int** *imageIndex*, **);**
VB	**LoadImage (** *fileName* **as String,** *options* **as String,** *fileType* **as String,** *tifByteOrder* **as Integer** **)** **as Integer;**

Parameters

fileName
The name of the file that will be loaded.

options
A string of characters that describe optional processing for the image data as it is loaded.

B Bit-byte-reverses the image data stream.
T Allows color components to be suppressed when loading color TIFF and JPEG files.
T=-1 Suppress red
T=-2 Suppress green
T=-3 Suppress blue

fileType
Specifies the file organization. It is only needed when the type cannot be determined by the file extension or by reading the header information in the file. The supported file types are:

TIFF	TIFF file, including ViewStar TIFF format
PCX	PCX
DCX	Multi-page PCX
PDF	Adobe®-compatible PDF (output only)
NOHEADER	Data only, no header
BMP	Bitmap
JPG	JPEG

imageIndex
Specifies which image to load in a multi-image TIFF file. The range is 1-n.

ampResult
The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code. An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating variables within this method. An exception will be thrown (e_FileNotFound) if this method cannot find the file to load. An exception will be thrown (E_INVALIDARG) if the options, fileType or imageIndex parameters incorrect.

Remarks

This method loads an image file into the image object. If AutoPrepEnable is active, then several combinations of image processing, thresholding, and MICR reading operations will be performed in order to find the optimal settings for the best MICR read. Once these optimal settings are found, the processed image will be scaled to 200 dpi. AutoPrep is best used on grayscale images of unknown resolution.

See Also

IampImage::SetWindow, IampImage::AutoPrepEnable

2.15. IampImage::LoadImageBuffer

The **IampImage::LoadImageBuffer** method loads an image file located in a memory buffer into the image object.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT LoadImageBuffer (** **BYTE *** *pbyBuffer,* **BSTR** *options,* **BSTR** *fileType,* **LONG** *imageIndex,* **LONG** *bufferSize,* **LONG*** *ampResult* **);**
C#	**int LoadImageBuffer (** **BYTE** **pbyBuffer,* **string** *options,* **string** *fileType,* **int** *imageIndex,* **int** *bufferSize,* **);**
C	**HRESULT IampImage_LoadImageBuffer (** **BYTE *** *pbyBuffer,* **BSTR** *options,* **BSTR** *fileType,* **LONG** *imageIndex,* **LONG** *bufferSize,* **LONG*** *ampResult* **);**
JAVA	**int LoadImageBuffer (** **BYTE *** *pbyBuffer,* **String** *options,* **String** *fileType,* **int** *imageIndex,* **int** *bufferSize,* **);**

VB	**LoadImageBuffer (** *pbyBuffer* **as INTPTR,** *options* **as String,** *fileType* **as String,** *tifByteOrder* **as Integer,** *bufferSize* **as Integer,** **)** **as Integer;**

Parameters

pbyBuffer
The location of the memory buffer that holds the file that will be loaded.

options
A string of characters that describe optional processing for the image data as it is loaded.

B Bit-byte-reverses the image data stream.
T Allows color components to be suppressed when loading color TIFF and JPEG files.
T=-1 Suppress red
T=-2 Suppress green
T=-3 Suppress blue

fileType
Specifies the file organization. It is only needed when the type cannot be determined by the file extension or by reading the header information in the file. The supported file types are:

TIFF	TIFF file, including ViewStar TIFF format
PCX	PCX
DCX	Multi-page PCX
PDF	Adobe®-compatible PDF (output only)
NOHEADER	Data only, no header
BMP	Bitmap
JPG	JPEG

imageIndex
Specifies which image to load in a multi-image TIFF file. The range is 1-n.

bufferSize
The size of the file in bytes that has been loaded into the memory buffer.

ampResult
The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code. An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating variables within this method. An exception will be thrown (e_FileNotFound) if this method cannot find the file to load. An exception will be thrown (E_INVALIDARG) if the options, fileType or imageIndex parameters incorrect.

Remarks

This method loads an image file that has been read into a memory buffer into the image object.

See Also

IampImage::SetWindow, IampImage::LoadImage

2.16. IampImage::LoadImageFromMemory

The **IampImage::LoadImageFromMemory** method loads an image file located in a memory buffer into the image object.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT LoadImageFromMemory (** **LONG** *dwBufferAddress,* **BSTR** *options,* **BSTR** *fileType,* **LONG** *imageIndex,* **LONG** *bufferSize,* **LONG*** *ampResult* **);**
C#	**int LoadImageFromMemory (** **int** *dwBufferAddress,* **string** *options,* **string** *fileType,* **int** *imageIndex,* **int** *bufferSize,* **);**
C	**HRESULT IampImage_LoadImageFromMemory (** **LONG** *dwBufferAddress,* **BSTR** *options,* **BSTR** *fileType,* **LONG** *imageIndex,* **LONG** *bufferSize,* **LONG*** *ampResult* **);**
JAVA	**int LoadImageFromMemory (** **int** *dwBufferAddress,* **String** *options,* **String** *fileType,* **int** *imageIndex,* **int** *bufferSize,* **);**
VB	**LoadImageFromMemory (** *dwBufferAddress* **as INTPTR,** *options* **as String,** *fileType* **as String,** *tifByteOrder* **as Integer,** *bufferSize* **as Integer,** **)** **as Integer;**

Parameters

dwBufferAddress
The location of the memory buffer that holds the file that will be loaded.

options
A string of characters that describe optional processing for the image data as it is loaded.

B Bit-byte-reverses the image data stream.
T Allows color components to be suppressed when loading color TIFF and JPEG files.
T=-1 Suppress red
T=-2 Suppress green
T=-3 Suppress blue

fileType
Specifies the file organization. It is only needed when the type cannot be determined by the file extension or by reading the header information in the file. The supported file types are:

TIFF TIFF file, including ViewStar TIFF format

PCX	PCX
DCX	Multi-page PCX
PDF	Adobe®-compatible PDF (output only)
NOHEADER	Data only, no header
BMP	Bitmap
JPG	JPEG

imageIndex
Specifies which image to load in a multi-image TIFF file. The range is 1-n.

bufferSize
The size of the file in bytes that has been loaded into the memory buffer.

ampResult
The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code. An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating variables within this method. An exception will be thrown (e_FileNotFound) if this method cannot find the file to load. An exception will be thrown (E_INVALIDARG) if the options, fileType or imageIndex parameters incorrect.

Remarks

This method loads an image file that has been read into a memory buffer into the image object. This method is identical to LoadImageBuffer except for the type of the first parameter dwBufferAddress/pbyBuffer.

See Also

IampImage::LoadImage, IampImage::LoadImageBuffer

2.17. IampImage::PasteImageFromBuffer

The **IampImage::PasteImageFromBuffer** method pastes an image buffer into the image object.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT PasteImageFromBuffer (** **LONG** *nBufferIndex*, **LONG*** *ampResult* **);**
C#	**int PasteImageFromBuffer (** **int** *nBufferIndex* **);**
C	**HRESULT IampImage_PasteImageFromBuffer (** **LONG** *nBufferIndex*, **LONG*** *ampResult* **);**
JAVA	**int PasteImageFromBuffer (** **int** *nBufferIndex* **);**

VB	**PasteImageFromBuffer (** *nBufferIndex* **as Integer** **)** **as Integer;**

Parameters

nBufferIndex
The index number of the buffer that will be pasted into the image object. Valid indices are 0 through 9.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code. AMPLib error code 47 will be returned if the nBufferIndex parameter is outside of (0,9). An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating variables within this method.

Remarks

This method restores the copied contents of the image object from a buffer.

See Also

IampImage::CopyImageToBuffer

2.18. IampImage::pixelBitDepth

The **IampImage::pixelBitDepth** property is the number of bits used for each pixel in the loaded image.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT get_pixelBitDepth (** **LONG*** *pVal* **);**
C#	**Long pixelBitDepth**
C	**HRESULT IampImage_get_pixelBitDepth (** **LONG*** p*Val* **);**
JAVA	**void get_pixelBitDepth (** **int*** *pVal* **);**
VB	**pixelBitDepth () as Long;**

Parameters

pVal
A pointer to a variable that will receive the value.

Remarks

This value remains at 1 unless GrayImageEnable is active. If GrayImageEnable is active then the value will be 8 if a grayscale or color image is loaded with LoadImage or if LoadBlankImage is called. An exception will be thrown (e_NoImage) if this property is accessed before an image is loaded.

See Also

IampImage::LoadImage, IampImage::LoadBlankImage

2.19. IampImage::traceEnable

The **IampImage::traceEnable** property determines whether or not the AMPLIB DLL trace feature is enabled.

Quick Info

See IampImage: IDispatch.

C++	**HRESULT put_traceEnable (** **int** *traceEnable* **);** **HRESULT get_traceEnable (** **int*** *traceEnable* **);**
C#	**int** *traceEnable*
C	**HRESULT IampImage_put_traceEnable (** **int*** *traceEnable* **);** **HRESULT IampImage_get_traceEnable (** **int*** *traceEnable* **);**
JAVA	**void put_traceEnable (** **int*** *traceEnable* **);** **void get_traceEnable (** **int*** *traceEnable* **);**
VB	**traceEnable as Integer;**

Parameters

traceEnable
An integer value. If set to 1, the trace feature will be enabled. If set to 0, the trace feature will be disabled.

Remarks

The AMPLIB DLL trace feature records a log of significant events to a text file. This log may be useful to All My Papers Customer Support during troubleshooting.

See Also

IampImage::traceFile

2.20. IampImage::traceFile

The **IampImage::traceFile** property contains the destination filename of the AMPLIB DLL trace file.

Quick Info

See IampImage: IDispatch.

C++	**HRESULT put_traceFile (** **BSTR** *traceFile* **);** **HRESULT get_traceFile (** **BSTR*** *traceFile* **);**
C#	**string** *traceFile*
C	**HRESULT IampImage_put_traceFile (** **BSTR** *traceFile* **);** **HRESULT IampImage_get_traceFile (** **BSTR*** *traceFile* **);**
JAVA	**void put_traceFile (** **String** *traceFile* **);** **void get_traceFile (** **String** *traceFile* **);**
VB	**traceFile as String;**

Parameters

traceFile
A string that contains the path to the trace text file.

Remarks

The AMPLIB DLL trace feature records a log of significant events to a text file. This log may be useful to All My Papers Customer Support during troubleshooting.

See Also

IampImage::traceEnable

2.21. IampImage::ProcessGrayImage

The **IampImage::ProcessGrayImage** method performs the selected image processing operation on the grayscale workimage object.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT ProcessGrayImage (** **LONG** *lOperation*, **LONG*** *ampResult* **);**
C#	**int ProcessGrayImage (** **int** *lOperation*, **);**
C	**HRESULT IampImage_ ProcessGrayImage (** **LONG** *lOperation*, **LONG*** *ampResult* **);**
JAVA	**int ProcessGrayImage (** **int** *lOperation*, **);**

VB	**ProcessGrayImage (** *lOperation* **as Integer,** **)** **as Integer;**

Parameters

lOperation
Specifies the image processing operation to be used. This set of values is identical to the set used for ampGrayProcesses and is shown below:

0 - GRAYLIGHTEN - lightens the image 8 greyscale steps
1 - GRAYDARKEN - darkens the image 8 greyscale steps
2 - GRAYREVERSE - 0-255 becomes 255-0
3 - GRAYSTRETCH - stretches greyscales from 32-224 to 0-255
4 - GRAYCOMPRESS - 0-255 becomes 32-224
5 - GRAYNORMALIZE - the current dynamic range becomes 0-255
6 - GRAYTHRESHOLD - 0-255 is thresholded at 64
7 - GRAYMILDSHARPEN - Mildly enhance the image edges
8 - GRAYSTRONGSHARPEN - Strongly enhance the image edges
9 - GRAYMILDBLEND - Mildly enhance the image edges
10 - GRAYSTRONGBLEND - Strongly enhance the image edges
11 - GRAYGAMMALIGHTEN1 – First style of gamma lightening
12 - GRAYGAMMALIGHTEN2 – Second style of gamma lightening
13 - GRAYGAMMALIGHTEN3 – Third style of gamma lightening
14 - GRAYGAMMAEQUALIZE – Gamma dynamic range equalization

ampResult
The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

This method performs an operation on the 8-bit grayscale workimage object. If the image is bilevel, it will be promoted to 8-bit grayscale first. An exception will be thrown (e_NoImage) if this method is called before an image is loaded.

See Also

2.22. IampImage::PromoteBilevelImage

The **IampImage::PromoteBilevelImage** method converts a bilevel workimage property to 8-bit grayscale.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT PromoteBilevelImage (** **LONG*** *ampResult* **);**
C#	**int PromoteBilevelImage (** **);**
C	**HRESULT IampImage_PromoteBilevelImage (** **LONG*** *ampResult* **);**

JAVA	**int PromoteBilevelImage (** **);**
VB	**PromoteBilevelImage (** **)** **as Integer;**

Parameters

ampResult
The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

This method promotes 1-bit per pixel workimage property to 8-bit per pixel grayscale leaving the visual appearance unchanged. If the workimage is already grayscale then the method immediately returns with success. An exception will be thrown (e_NoImage) if this method is called before an image is loaded.

See Also

2.23. IampImage::RotateImageToBuffer

The **IampImage::RotateImageToBuffer** method rotates the current image object selection and stores the result in an image buffer.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT RotateImageToBuffer (** **LONG** *nDegrees*, **LONG** *nBufferIndex*, **LONG*** *ampResult* **);**
C#	**int RotateImageToBuffer (** **int** *nDegrees*, **int** *nBufferIndex* **);**
C	**HRESULT IampImage_RotateImageToBuffer (** **LONG** *nDegrees*, **LONG** *nBufferIndex*, **LONG*** *ampResult* **);**
JAVA	**int RotateImageToBuffer (** **int** *nDegrees*, **int** *nBufferIndex* **);**

VB	**RotateImageToBuffer (** *nDegrees* **as Integer,** *nBufferIndex* **as Integer** **)** **as Integer;**

Parameters

nDegrees
The amount of degrees the source image will be rotated: 90, 180, or 270.

nBufferIndex
The index number of the buffer into which the scaled image will be written. Valid indices are 0 through 9.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code. AMPLib error code 47 will be returned if the nBufferIndex parameter is outside of (0,9). An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating variables within this method.

Remarks

This method rotates the current image object counter-clockwise the amount of degrees selected and then stores the result image in the selected buffer for future retrieval. The image object is not affected by this operation.

The image object selection can be set with the IampImage::SetWindow command. If no selection is specified, the entire image object is rotated.

See Also

IampImage::SetWindow
IampImage::PasteImageFromBuffer

2.24. IampImage::SaveImage

The **IampImage::SaveImage** method saves an image file from the image object to disk.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT SaveImage (** **BSTR** *fileName*, **BSTR** *options*, **BSTR** *fileType*, **LONG*** *ampResult* **);**
C#	**int SaveImage (** **string** *fileName*, **string** *options*, **string** *fileType* **);**
C	**HRESULT IampImage_SaveImage (** **BSTR** *fileName*, **BSTR** *options*, **BSTR** *fileType*, **LONG*** *ampResult* **);**

JAVA	**int SaveImage (** **String** *fileName*, **String** *options*, **String** *fileType* **);**
VB	**SaveImage (** *fileName* **as String,** *options* **as String,** *fileType* **as String** **)** **as Integer;**

Parameters

fileName

The name of the file that will be saved. Some extensions are automatically recognized:

TIF	File is a TIFF file
PCX	File is a Paintbrush PCX file
BMP	File is a Bitmap
JPG	File is a JPEG

options

A string of characters that describe optional processing for the image data during save:

A Appends image data to file; used for creating multi-image format files.

B Bit-byte-reverses the image data stream. See note above. Without this option, TIFF files will be created with Fill Order 2. With the **B** option, the Fill Order is set to 1.

Q Format is "Q=n", which sets the quality level for JPEG files.

R Format is "R=*n*", which sets the X and Y resolution tags in the TIFF file to *n*; used when the resolution information is not already known internally or you need to override the value.

X Format is "X=*n*", which sets the X-resolution tag in the TIFF file to *n*.

Y Format is "Y=*n*", which sets the Y-resolution tag in the TIFF file to *n*.

U Format is "U=*n*", which sets the ResolutionUnits tag in the TIFF file to *n*; the default value is 2.

fileType

Specifies the file organization. It is only needed when the type cannot be determined by the file extension or by reading the header information in the file. The supported file types are:

TIFF	TIFF file, including ViewStar TIFF format
PCX	PCX
DCX	Multi-page PCX
PDF	Adobe®-compatible PDF (output only)
NOHEADER	Data only, no header
BMP	Bitmap
JPG	JPEG

ampResult

The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code. An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating variables within this method. An exception will be thrown (e_FileNotFound) if this method cannot find the directory in which to save the file. An exception will be thrown (E_INVALIDARG) if the options, fileType or imageIndex parameters incorrect.

Remarks

This method save image object to a compressed disk file.

See Also

IampImage::LoadImage

2.25. IampImage::SaveImageEx

The **IampImage::SaveImageEx** method saves an image file from the image object to disk with enhanced control over the ASCII tags in the output file TIFF header.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT SaveImage (** **BSTR** *fileName*, **BSTR** *options*, **BSTR** *fileType*, **LONG** *nNoAscii*, **LONG*** *ampResult* **);**
C#	**int SaveImage (** **string** *fileName*, **string** *options*, **string** *fileType*, **int** *nNoAscii* **);**
C	**HRESULT IampImage_SaveImage (** **BSTR** *fileName*, **BSTR** *options*, **BSTR** *fileType*, **LONG** *nNoAscii*, **LONG*** *ampResult* **);**
JAVA	**int SaveImage (** **String** *fileName*, **String** *options*, **String** *fileType*, **int** *nNoAscii*, **);**
VB	**SaveImage (** *fileName* **as String,** *options* **as String,** *fileType* **as String,** *nNoAscii* **as Integer,** **)** **as Integer;**

Parameters

fileName

The name of the file that will be saved. Some extensions are automatically recognized:

TIF	File is a TIFF file
PCX	File is a Paintbrush PCX file
BMP	File is a Bitmap

JPG	File is a JPEG

options
A string of characters that describe optional processing for the image data during save:

- **A** Appends image data to file; used for creating multi-image format files.
- **B** Bit-byte-reverses the image data stream. See note above. Without this option, TIFF files will be created with Fill Order 2. With the **B** option, the Fill Order is set to 1.
- **Q** Format is "Q=n", which sets the quality level for JPEG files.
- **R** Format is "R=*n*", which sets the X and Y resolution tags in the TIFF file to *n*; used when the resolution information is not already known internally or you need to override the value.
- **X** Format is "X=*n*", which sets the X-resolution tag in the TIFF file to *n*.
- **Y** Format is "Y=*n*", which sets the Y-resolution tag in the TIFF file to *n*.
- **U** Format is "U=*n*", which sets the ResolutionUnits tag in the TIFF file to *n*; the default value is 2.

fileType
Specifies the file organization. It is only needed when the type cannot be determined by the file extension or by reading the header information in the file. The supported file types are:

TIFF	TIFF file, including ViewStar TIFF format
PCX	PCX
DCX	Multi-page PCX
PDF	Adobe®-compatible PDF (output only)
NOHEADER	Data only, no header
BMP	Bitmap
JPG	JPEG

nNoAscii
When this value is nonzero, ASCII tags in the output TIFF file (if specified) will be suppressed. This option is useful for removing the Time Stamp tag from the output file.

ampResult
The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code. An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating variables within this method. An exception will be thrown (e_FileNotFound) if this method cannot find the directory in which to save the file. An exception will be thrown (E_INVALIDARG) if the options, fileType or imageIndex parameters incorrect.

Remarks

This method save image object to a compressed disk file.

See Also

IampImage::LoadImage

2.26. IampImage::SaveImageToClipboard

The **IampImage::SaveImageToClipboard** method saves the current image to the clipboard.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT SaveImageToClipboard (** **LONG** *hWnd*, **LONG*** *ampResult* **);**
C#	**int SaveImageToClipboard (** **int** *hWnd*, **);**
C	**HRESULT IampImage_ SaveImageToClipboard (** **LONG** *hWnd*, **LONG*** *ampResult* **);**
JAVA	**int SaveImageToClipboard (** **int** *hWnd*, **);**
VB	**SaveImageToClipboard (** *hWnd* **as Integer** **)** **as Integer;**

Parameters

hWnd
Specifies the windows handle for the current application.

ampResult
The error code returned.

Return Values

Remarks

This method saves the image contained within the image object to the clipboard in order that the image might be used by another Windows application or used within a VB application for display in a PictureBox.

See Also

IampImage::SetWindow

2.27. IampImage::SaveImageToMemory

The **IampImage::SaveImageToMemory** method saves the image object to memory in a compressed G4 TIFF data format.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT SaveImageToMemory (** **LONG** *dwBufferAddress*, **LONG** *dwBufferSize*, **BSTR** *options*, **BSTR** *fileType*, **LONG*** *pdwBufferUsed*, **LONG*** *ampResult* **);**

C#	**int SaveImageToMemory (** **int** *dwBufferAddress*, **int** *dwBufferSize*, **string** *options*, **string** *fileType*, **out int** *pdwBufferUsed* **);**
C	**HRESULT lampImage_SaveImageToMemory (** **LONG** *dwBufferAddress*, **LONG** *dwBufferSize*, **BSTR** *options*, **BSTR** *fileType*, **LONG*** *pdwBufferUsed*, **LONG*** *ampResult* **);**
JAVA	**int SaveImageToMemory (** **integer** *dwBufferAddress*, **integer** *dwBufferSize*, **String** *options*, **String** *fileType*, **integer*** *pdwBufferUsed*, **);**
VB	**SaveImageToMemory (** *dwBufferAddress* **as LONG,** *dwBufferSize* **as LONG,** *options* **as String,** *fileType* **as String,** *pdwBufferUsed* **as Integer,** **)** **as Integer;**

Parameters

dwBufferAddress

The starting address of the memory buffer in memory.

dwBufferSize

The size of the memory buffer in bytes.

options

A string of characters that describe optional processing for the image data during save:

- **B** Bit-byte-reverses the image data stream. See note above. Without this option, TIFF files will be created with Fill Order 2. With the **B** option, the Fill Order is set to 1.
- **R** Format is "R=*n*", which sets the X and Y resolution tags in the TIFF file to *n*; used when the resolution information is not already known internally or you need to override the value.
- **X** Format is "X=*n*", which sets the X-resolution tag in the TIFF file to *n*.
- **Y** Format is "Y=*n*", which sets the Y-resolution tag in the TIFF file to *n*.
- **U** Format is "U=*n*", which sets the ResolutionUnits tag in the TIFF file to *n*; the default value is 2.

fileType

Specifies the file organization. It is only needed when the type cannot be determined by the file extension or by reading the header information in the file. The supported file types are:

TIFF Tagged Image File Format file

pdwBufferUsed

The size of the compressed G4 data in bytes.

ampResult

The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code. An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating variables within this method. An exception will be thrown (e_FileNotFound) if this method cannot find the directory in which to save the file. An exception will be thrown (E_INVALIDARG) if the options, fileType or imageIndex parameters incorrect.

Remarks

This method saves the image object to an internal memory buffer in TIFF G4 encoding. This buffer is supplied by the caller and is recommended to be at least 512K bytes so that even complex G4 images will fit with room to spare. If the image object is grayscale, a temporary bilevel image will be created and the image object will be thresholded before saving as TIFF G4.

See Also

IampImage::LoadImageFromMemory, IampImage::LoadImage

2.28. IampImage::SaveImageToMemoryTest

The **IampImage::SaveImageToMemoryTest** method saves the image object to a 65K memory buffer using SaveImageToMemory and then to disk.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT SaveImage (** **BSTR** *fileName*, **BSTR** *options*, **BSTR** *fileType*, **LONG*** *ampResult* **);**
C#	**int SaveImage (** **string** *fileName*, **string** *options*, **int** *imageIndex*, **);**
C	**HRESULT IampImage_SaveImage (** **BSTR** *fileName*, **BSTR** *options*, **BSTR** *fileType*, **LONG*** *ampResult* **);**
JAVA	**int SaveImage (** **String** *fileName*, **String** *options*, **String** *fileType*, **);**
VB	**SaveImage (** *fileName* **as String,** *options* **as String,** *fileType* **as String,** **)** **as Integer;**

Parameters

fileName

The name of the file that will be saved. Since only single-page bilevel images are supported, a TIF extension is recommended.

options

A string of characters that describe optional processing for the image data during save:

B Bit-byte-reverses the image data stream. See note above. Without this option, TIFF files will be created with Fill Order 2. With the **B** option, the Fill Order is set to 1.

R Format is "R=*n*", which sets the X and Y resolution tags in the TIFF file to *n*; used when the resolution information is not already known internally or you need to override the value.

X Format is "X=*n*", which sets the X-resolution tag in the TIFF file to *n*.

Y Format is "Y=*n*", which sets the Y-resolution tag in the TIFF file to *n*.

U Format is "U=*n*", which sets the ResolutionUnits tag in the TIFF file to *n*; the default value is 2.

fileType

Specifies the file organization. It is only needed when the type cannot be determined by the file extension or by reading the header information in the file. The supported file types are:

TIFF Tagged Image File Format file

ampResult

The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code. An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating variables within this method. An exception will be thrown (e_FileNotFound) if this method cannot find the directory in which to save the file. An exception will be thrown (E_INVALIDARG) if the options, fileType or imageIndex parameters incorrect.

Remarks

This method saves the image object to an internal 65K memory buffer in TIFF G4 encoding and then writes the buffer to disk using the specified filename.

See Also

IampImage::SaveImageToMemory

2.29. IampImage::ScaleImageToBuffer

The **IampImage::ScaleImageToBuffer** method scales the current image object selection and stores the result in an image buffer.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT ScaleImageToBuffer (** **LONG** *nWidth*, **LONG** *nHeight*, **LONG** *nBufferIndex*, **LONG*** *ampResult* **);**
C#	**int ScaleImageToBuffer (** **int** *nWidth*, **int** *nHeight*, **int** *nBufferIndex* **);**

C	**HRESULT IampImage_ScaleImageToBuffer (** **LONG** *nWidth*, **LONG** *nHeight*, **LONG** *nBufferIndex*, **LONG*** *ampResult* **);**
JAVA	**int ScaleImageToBuffer (** **int** *nWidth*, **int** *nHeight*, **int** *nBufferIndex* **);**
VB	**ScaleImageToBuffer (** *nWidth* **as Integer,** *nHeight* **as Integer,** *nBufferIndex* **as Integer** **)** **as Integer;**

Parameters

nWidth
The width, in pixels, of the scaled image.

nHeight
The height, in pixels, of the scaled image.

nBufferIndex
The index number of the buffer into which the scaled image will be written. Valid indices are 0 through 9.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code. AMPLib error code 47 will be returned if the nBufferIndex parameter is outside of (0,9). An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating variables within this method.

Remarks

This method scales the current image object selection to the supplied height and width, then stores the result image in a buffer for future retrieval. The image object is not affected by this operation.

The image object selection can be set with the IampImage::SetWindow command. If no selection is specified, the entire image object is scaled.

See Also

IampImage::CopyImageToBuffer
IampImage::SetWindow
IampImage::PasteImageFromBuffer

2.30. IampImage::SecurityEnableAppsFile

The **IampImage::SecurityEnableAppsFile** method will access an OEM license file and enable licensing for the application using the COM object.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT SecurityEnableAppsFile (** **BSTR** *LicenseFileName*, **LONG** *ProductNumber*, **BSTR** *UserDefName*, **UINT** *P1*, **UINT** *P2*, **LONG*** *ax9Result* **);**
C#	**int SecurityEnableAppsFile (** **string** *LicenseFileName*, **int** *ProductNumber*, **string** *UserDefName*, **uint** *P1*, **uint** *P2*, **);**
C	**HRESULT lampImage_SecurityEnableAppsFile (** **BSTR** *LicenseFileName*, **LONG** *ProductNumber*, **BSTR** *UserDefName*, **UINT** *P1*, **UINT** *P2*, **LONG*** *ax9Result* **);**
JAVA	**int SecurityEnableAppsFile (** **String** *LicenseFileName*, **int** *ProductNumber*, **String** *UserDefName*, **uint** *P1*, **uint** *P2*, **);**
VB	**SecurityEnableAppsFile (** *LicenseFileName* **as String**, *ProductNumber* **as Integer**, *UserDefName* **as String**, *P1* **as UInteger**, *P2* **as UInteger**, **)** **as Integer;**

Parameters

LicenseFileName
The folder or pathname of the license file (e.g. “C:\WINDOWS\AMP\LICENSES\”)

ProductNumber
The decimal number that matches the hex suffix in the license file name (e.g. 106 for SIPC006A).

UserDefName
The kind of license file (e.g. “AMPLIB OEM”)

P1
The first 32-bit hexadecimal code that is used to unlock the license.

P2
The second 32-bit hexadecimal code that is used to unlock the license.

ax9Result
The error/success code returned.

Return Values

If the function succeeds, the return code is 1, otherwise the return code is 0. An exception will be thrown (E_OUTOFMEMORY) if the system runs out of memory while allocating variables within this method.

Remarks

This method will read the OEM license contained in the license file specified by LicenseFileName and ProductNumber and if its contents match the UserDefName, P1, and P2 values, then the specified AMPLib licenses are activated for the application using the COM object. This function is very similar to the AMPLib function ampSecurityEnableAppsFile.

See Also

2.31. IampImage::SetRegion

The **IampImage::SetRegion** method sets the region of interest used for OCR-A and OCR-B recognition.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT SetRegion (** **LONG** *nRegionEnable*, **LONG** *nTopOffset*, **LONG** *nLeftOffset*, **LONG** *nRoiX*, **LONG** *nRoiY*, **LONG** *nRoiDX*, **LONG** *nRoiDY*, **LONG*** *ampResult* **);**
C#	**int SetRegion (** **int** *nRegionEnable*, **int** *nTopOffset*, **int** *nLeftOffset*, **int** *nRoiX*, **int** *nRoiY*, **int** *nRoiDX*, **int** *nRoiDY*, **);**
C	**HRESULT IampImage_SetRegion (** **LONG** *nRegionEnablet*, **LONG** *nTopOffset*, **LONG** *nLeftOffset*, **LONG** *nRoiX*, **LONG** *nRoiY*, **LONG** *nRoiDX*, **LONG** *nRoiDY*, **LONG*** *ampResult* **);**
JAVA	**int SetRegion (** **int** *nRegionEnablet*, **int** *nTopOffset*, **int** *nLeftOffset*, **int** *nRoiX*, **int** *nRoiY*, **int** *nRoiDX*, **int** *nRoiDY*, **);**

VB	**SetRegion (** *nRegionEnable* **as Integer,** *nTopOffset* **as Integer,** *nLeftOffset* **as Integer,** *nRoiX* **as Integer,** *nRoiY* **as Integer,** *nRoiDX* **as Integer,** *nRoiDY* **as Integer** **)** **as Integer;**

Parameters

nRegionEnable
If nonzero, the following parameters are used in conjunction with the image dimensions to Establish a region of interest for OCR-A and OCR-B recognition using ReadMICR.

nTopOffset
If nonzero the Roi references the top edge of the image. Otherwise the bottom edge is used.

nLeftOffset
If nonzero the Roi references the left edge of the image. Otherwise the right edge is used.

nRoiX
ROI Pixel distance from the left or right image edge.

nRoiY
ROI Pixel distance from the top or bottom image edge.

nRoiDX
Pixel width of the region of interest.

nRoiDY
Pixel height of the region of interest.

ampResult
The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

This method sets the region of interest used for OCR-A and OCR-B recognition when ReadMICR is called. If the nRegionEnable value is 0, then ROI will default to the MICR region at the bottom of the check. Otherwise, the DX and DY values set the width and the height of the region. The input values are used in the same way their companion variables in the ampOCRINFO structure as described in the AMPLib manual.

See Also

IampMICR::ReadMICR

2.32. IampImage::SetWindow

The **IampImage::SetWindow** method sets the region of interest.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT SetWindow (** **LONG** *left*, **LONG** *top*, **LONG** *width*, **LONG** *height*, **LONG*** *ampResult* **);**
C#	**int SetWindow (** **int** *left*, **int** *top*, **int** *width*, **int** *height*, **);**
C	**HRESULT IampImage_SetWindow (** **LONG** *left*, **LONG** *top*, **LONG** *width*, **LONG** *height*, **LONG*** *ampResult* **);**
JAVA	**int SetWindow (** **int** *left*, **int** *top*, **int** *width*, **int** *height*, **);**
VB	**SetWindow (** *left* **as Integer,** *top* **as Integer,** *width* **as Integer,** *height* **as Integer** **)** **as Integer;**

Parameters

left
Specifies the number of pixels for the left margin of the region.

top
Specifies the number of pixels for the top margin of the region.

width
Specifies the number of pixels for the horizontal length of the region. Any negative value will set this value to the maximum possible.

height
Specifies the number of pixels for the vertical length of the region. Any negative value will set this value to the maximum possible.

ampResult
The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

This method sets the region of interest. If all the values are zero, then the width and height of the region will be set to its maximum value with a left and top value of 0. If the left and width value exceed the total width of the image, left will be set to 0 and width to the total width of the image. If the top and height value exceed the total height of the image,

top will be set to 0 and height to the total height of the image. An exception will be thrown (e_NoImage) if this method is called before an image is loaded.

See Also

IampImage::GetWindow

2.33. IampImage::ThresholdGrayImage

The **IampImage::ThresholdGrayImage** method converts a grayscale workimage into bilevel.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT ThresholdGrayImage (** **LONG** *lStyle*, **LONG** *lValue1*, **LONG** *lValue2*, **LONG*** *ampResult* **);**
C#	**int ThresholdGrayImage (** **int** *lStyle*, **int** *lValue1*, **int** *lValue2*, **);**
C	**HRESULT IampImage_ ThresholdGrayImage (** **LONG** *lStyle*, **LONG** *lValue1*, **LONG** *lValue2*, **LONG*** *ampResult* **);**
JAVA	**int ThresholdGrayImage (** **int** *lStyle*, **int** *lValue1*, **int** *lValue2*, **);**
VB	**ThresholdGrayImage (** *lStyle* **as Integer,** *lValue1* **as Integer,** *lValue2* **as Integer,** *height* **as Integer** **)** **as Integer;**

Parameters

lStyle
Specifies the overall style of thresholding that will be used: 0 - dynamic, 1 – adaptive, 3- fixed slice value.

lValue1
If lStyle is set for dynamic thresholding (zero), then a 0 for this value should be used for optimal recognition of finer details and 1 for more filtering of single pixel noise. Values of 2 and 3 are similar to 0 and 1 with exception that low-pass filtering is turned off. lValue2 is used for the AdjustableBlackThreshold parameter.

If lStyle is adaptive thresholding (one), then a 0 for this value is the default and a 1 can be used to enhance fine detail.

If lStyle is set to a fixed slice value (three) then this parameter has the following values:
0 - thresholds all gray values less than 64 to black.
1 – thresholds all gray values less than 128 to black.
2 – thresholds all gray values less than 192 to black.

lValue2
This parameter should generally be set to zero.

ampResult
The error code returned.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

This method thresholds an 8-bit grayscale workimage into bilevel using a number of selectable algorithms. If the image is already bilevel, the method will exit promptly with success. An exception will be thrown (e_NoImage) if this method is called before an image is loaded.

See Also

IampImage::DynamicThresholdGrayImage, IampImage::GetWindow

2.34. IampImage::xResolution

The **IampImage::xResolution** property is the horizontal resolution of the loaded image.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT get_xResolution (** **LONG*** *pVal* **);**
C#	**Long xResolution**
C	**HRESULT IampImage_get_ xResolution (** **LONG*** p*Val* **);**
JAVA	**void get_ xResolution (** **int*** *pVal* **);**
VB	**xResolution () as Long;**

Parameters

pVal
A pointer to a variable that will receive the value.

Remarks

The xResolution of the image remains unchanged until a new image is loaded. An exception will be thrown (e_NoImage) if the image is not loaded before accessing this property.

See Also

IampImage::yResolution

2.35. IampImage::yResolution

The **IampImage::yResolution** property is the vertical resolution of the loaded image.

Quick Info

See IampImage : IDispatch.

C++	**HRESULT get_yResolution (** **LONG*** *pVal* **);**
C#	**Long yResolution**
C	**HRESULT IampImage_get_ yResolution (** **LONG*** p*Val* **);**
JAVA	**void get_ yResolution (** **int*** *pVal* **);**
VB	**yResolution () as Long;**

Parameters

pVal
A pointer to a variable that will receive the value.

Remarks

The yResolution of the image remains unchanged until a new image is loaded. An exception will be thrown (e_NoImage) if the image is not loaded before accessing this property.

See Also

IampImage::xResolution

3. IampBarcode: IDispatch

The **IampBarcode** interface provides a method for scanning barcodes in an image. It is implemented within the ampImage class and is provided as a separate interface.

Quick Info

Header file:	ampImage.h
Interface identifier:	IID_IampBarcode
Pointer type:	ampImage*

Vtable

Methods	Descriptions
ScanBarCodes	Scans the image for barcodes.

Properties	Access
barHeight	Read only.
barWidth	Read only.
checkSum1	Read only.
checkSum2	Read only.
confidence	Read only.
count	Read only.
doCheckSum	Write only.
filter	Write only.
fixedLength	Write only.
forceResolution	Write only.
height	Write only.
index	Write only.
maxCharCount	Write only.
maxScanTime	Write only.
maxToScan	Write only.
minLength	Write only.
minPartialLength	Write only.
numberTested	Read only.
numberToFind	Write only.
orientation	Write only.
partial	Read only.
partialCount	Read only.
prLarge	Write only.
prMedium	Write only.
prSmall	Write only.
quality	Write only.
reversed	Read only.

rotated	Read only.
symbology	Read only.
symbologyMask	Write only.
text	Read only.
width	Write only.
xOrigin	Read only.
yOrigin	Read only.

Remarks

The **IampBarcode** interface inherits directly from **IDispatch**. The symbologyMask property must be set before calling ScanBarCodes. The other properties will use default values specified in the following sections unless modified prior to calling **ScanBarCodes**. The read only properties are only valid after calling **ScanBarCodes** and setting the index property.

3.2. IampBarcode::ScanBarCodes

The **IampBarcode::ScanBarCodes** method scans the loaded image for barcodes.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT ScanBarCodes (** **LONG*** *ampResult* **);**
C#	**int ScanBarCodes (** **);**
C	**HRESULT IampBarCodes_ScanBarCodes (** **LONG*** *ampResult* **;**
JAVA	**int ScanBarCodes (** **);**
VB	**ScanBarCodes (** **)** **as Integer;**

Parameters

ampResult
The error code returned. Error codes are defined in Appendix A.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

The image must be loaded and the **symbologyMask** property must be set prior to scanning the barcodes in an image. An exception will be thrown (e_SymbologyMaskNotSet) if this method is called prior to setting the mask. All other properties have default values as described in the respective sections. Once **ScanBarCodes** has been called, the number of barcodes found can be accessed through the **count** property. The information for each barcode can then be accessed after setting the **index** property. An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating variables within this method.

See Also

IampBarcode::symbologyMask, IampBarcode::count, IampBarcode::index, Appendix A

3.3. IampBarcode::barHeight

The **IampBarcode::barHeight** property is the height in pixels of the scanned barcode.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT get_barHeight (** **LONG*** *pVal* **);**
C#	**int barHeight**
C	**HRESULT IampBarcode_get_barHeight (** **LONG*** *pVal* **);**
JAVA	**void get_barHeight (** **int*** *pVal* **);**
VB	**barHeight () as Integer;**

Parameters

pVal
A pointer to a variable that will receive the value.

Remarks

This property is only available after calling **IampBarcode::ScanBarCodes** and setting the **IampBarcode::index** property. An exception will be thrown (e_BarcodesNotScanned) if this property is accessed before calling **ScanBarCodes**.

See Also

IampBarcode::ScanBarCodes, IampBarcode::index

3.4. IampBarcode::barWidth

The **IampBarcode::barWidth** property is the width in pixels of the scanned barcode.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT get_barWidth (** **LONG*** *pVal* **);**
C#	**int barWidth**
C	**HRESULT IampBarcode_get_barWidth (** **LONG*** *pVal* **);**
JAVA	**void get_barWidth (** **int*** *pVal* **);**

VB	**barWidth () as Integer;**

Parameters

pVal
A pointer to a variable that will receive the value.

Remarks

This property is only available after calling **IampBarcode::ScanBarCodes** and setting the **IampBarcode::index** property. An exception will be thrown (e_BarcodesNotScanned) if this property is accessed before calling **ScanBarCodes**.

See Also

IampBarcode::ScanBarCodes, IampBarcode::index

3.5. IampBarcode::checkSum1

The **IampBarcode::checkSum1** property is the checksum of the scanned barcode.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT get_checkSum1 (** **LONG*** *pVal* **);**
C#	**int checkSum1**
C	**HRESULT IampBarcode_get_checkSum1 (** **LONG*** *pVal* **);**
JAVA	**void get_checkSum1 (** **int*** *pVal* **);**
VB	**checkSum1 () as Integer;**

Parameters

pVal
A pointer to a variable that will receive the value.

Remarks

This property is only available after calling **IampBarcode::ScanBarCodes** and setting the **IampBarcode::index** property. An exception will be thrown (e_BarcodesNotScanned) if this property is accessed before calling **ScanBarCodes**.

See Also

IampBarcode::ScanBarCodes, IampBarcode::index

3.6. IampBarcode::checkSum2

The **IampBarcode::checkSum2** property is the second checksum of the scanned barcode. Some barcodes have two checksum values.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT get_checkSum2 (** **LONG*** *pVal* **);**
C#	**int checkSum2**
C	**HRESULT IampBarcode_get_checkSum2 (** **LONG*** *pVal* **);**
JAVA	**void get_checkSum2 (** **int*** *pVal* **);**
VB	**CheckSum2 () as Integer;**

Parameters

pVal
A pointer to a variable that will receive the value.

Remarks

This property is only available after calling **IampBarcode::ScanBarCodes** and setting the **IampBarcode::index** property. An exception will be thrown (e_BarcodesNotScanned) if this property is accessed before calling **ScanBarCodes**.

See Also

IampBarcode::ScanBarCodes, IampBarcode::index

3.7. IampBarcode::confidence

The **IampBarcode::confidence** property is a value between 0 (low) and 100 (high) which indicates the possible accuracy of the scanned barcode. Barcodes can be successfully scanned even though the confidence value is low.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT get_confidence (** **LONG*** *pVal* **);**
C#	**int confidence**
C	**HRESULT IampBarcode_get_confidence (** **LONG*** *pVal* **);**
JAVA	**void get_confidence (** **int*** *pVal* **);**
VB	**confidence () as Integer;**

Parameters

pVal
A pointer to a variable that will receive the value.

Remarks

This property is only available after calling **IampBarcode::ScanBarCodes** and setting the **IampBarcode::index** property. An exception will be thrown (e_BarcodesNotScanned) if this property is accessed before calling **ScanBarCodes**.

See Also

IampBarcode::ScanBarCodes, IampBarcode::index

3.8. IampBarcode::count

The **IampBarcode::count** property is the number of valid and partial barcodes that were scanned.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT get_count (** **LONG*** *pVal* **);**
C#	**int count**
C	**HRESULT IampBarcode_get_count (** **LONG*** *pVal* **);**
JAVA	**void get_count (** **int*** *pVal* **);**
VB	**count () as Integer;**

Parameters

pVal
A pointer to a variable that will receive the value.

Remarks

This property is only available after calling **IampBarcode::ScanBarCodes**. An exception will be thrown (e_BarcodesNotScanned) if this property is accessed before calling **ScanBarCodes**.

See Also

IampBarcode::ScanBarCodes, IampBarcode::index, IampBarcode::minPartialLength

3.9. IampBarcode::doCheckSum

The **IampBarcode::doCheckSum** property enables checksum verification for barcode data when such verification is optional for the symbology.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT put_doCheckSum (** **VARIANT_BOOL** *newVal* **);**
C#	**bool doCheckSum**

C	**HRESULT IampBarcode_put_doCheckSum (** **VARIANT_BOOL** *newVal* **);**
JAVA	**void set_doCheckSum (** **boolean** *newVal* **);**
VB	**doCheckSum () as Boolean;**

Parameters

newVal
True or False. If newVal set to **true**, the AIM standard checksum algorithm is performed. If the checksum fails validation, the barcode will not be reported as a successful scan.

Remarks

The default value for this property is **false**. If checksum verification is enabled, the resulting data will not include the checksum character(s). If the **minLength** property is too low, the entire barcode will not be reported.

See Also

IampBarcode::ScanBarCodes, IampBarcode::minLength

3.10. IampBarcode::filter

The **IampBarcode::filter** property specifies whether spot filtering should be applied to the barcode image before scanning.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT put_filter (** **LONG** *newVal* **);**
C#	**int filter**
C	**HRESULT IampBarcode_put_filter (** **LONG** *newVal* **);**
JAVA	**void set_filter (** **int** *newVal* **);**
VB	**filter () as Integer;**

Parameters

newVal
If newVal = 0, no filtering will occur. If newVal = 1, a 1x1 spot filter will be applied. If newVal = 2, a 2x1 spot filter will be used. For bilevel Data Matrix, Aztec, and QR images: newVal = 1 is a MAJOR 70 filter, newVal = 2 is a DILATE WEAK filter, newVal = 3 is an ERODE WEAK filter. For bilevel and grayscale QR images a newVal = 4 is a median filter. For grayscale QR images any newVal from 1 through 3 will perform a median filter.

Remarks

The default value for this property is 0 (no filtering). An exception will be thrown (E_INVALIDARG) if the new value is not one of these five.

See Also

IampBarcode::ScanBarCodes

3.11. IampBarcode::fixedLength

The **IampBarcode::fixedLength** property forces the scanned barcodes to be the exact number of characters specified by the **IampBarcode::minLength** property.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT put_fixedLength (** **VARIANT_BOOL** *newVal* **);**
C#	**bool fixedLength**
C	**HRESULT IampBarcode_put_fixedLength (** **VARIANT_BOOL** *newVal* **);**
JAVA	**void set_fixedLength (** **boolean** *newVal* **);**
VB	**fixedLength () as Boolean;**

Parameters

newVal
True or False. If newVal is **false**, variable length barcodes are allowed and must be greater in length than the **IampBarcode::minLength** property. If newVal is **true**, only barcodes which are exactly **IampBarcode::minLength** characters will be reported as successfully scanned.

Remarks

The default value for this property is **false**.

See Also

IampBarcode::ScanBarCodes, IampBarcode::minLength, IampBarcode::doCheckSum

3.12. IampBarcode::forceResolution

The **IampBarcode::forceResolution** property allows the application to override the image resolution. In **AmpLibNet**, this property is called **forceBarcodeResolution**.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT put_forceResolution (** **LONG** *newVal* **);**
C#	**int forceResolution**
C	**HRESULT IampBarcode_put_forceResolution (** **LONG** *newVal* **);**

JAVA	**void set_forceResolution (** **int** *newVal* **);**
VB	**forceResolution () as Integer;**

Parameters

newVal
If newVal = 0, the resolution of the image is used; provided the image file contains this information. If newVal > 0, the value specified by this property will be used as the image resolution for scanning the barcodes. If newVal < 0, an estimated resolution will be calculated and used for scanning.

Remarks

The default value for this property is 0.

See Also

IampBarcode::ScanBarCodes

3.13. IampBarcode::height

The **IampBarcode::height** property specifies the height of barcodes to be scanned in thousandths of an inch.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT put_height (** **LONG** *newVal* **);**
C#	**int height**
C	**HRESULT IampBarcode_put_height (** **LONG** *newVal* **);**
JAVA	**void set_height (** **int** *newVal* **);**
VB	**height () as Integer;**

Parameters

newVal
If newVal = 0, this property is ignored. If this property is > 0, the specified value is thousandths of an inch.

Remarks

The default value for this property is 0. Only use this property if the size of the barcodes to be scanned can be approximated. When used with the **prLarge**, **prMedium** and **prSmall** properties, a variety of barcode sizes can be scanned.

See Also

IampBarcode::ScanBarCodes, IampBarcode::width, IampBarcode::prLarge, IampBarcode::prMedium, IampBarcode::prSmall

3.14. IampBarcode::index

The **IampBarcode::index** property specifies which barcode, of the scanned set of barcodes, will be accessed through the read-only properties.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT put_index (** **LONG** *newVal* **);**
C#	**int index**
C	**HRESULT IampBarcode_put_index (** **LONG** *newVal* **);**
JAVA	**void set_index (** **int** *newVal* **);**
VB	**index () as Integer;**

Parameters

newVal
The range of values for newVal are 0 to n, where n is less than the **IampBarcode::count** property.

Remarks

In cases where barcode scanning may report more than one barcode in a single image, use this property to “select” which barcode will be accessed. This property must be set after calling the **IampBarcode::ScanBarCodes** method and before any properties relating to individual barcodes may be used. An exception will be thrown (e_BarcodesNotScanned) if **ScanBarCodes** has not been called before setting this property. An exception will be thrown (E_INVALIDARG) if the new value is out of range. An exception will be thrown (E_ABORT) if the barcode data specified by the **index** property cannot be accessed. The message reported in the exception will contain the internal error code.

See Also

IampBarcode::ScanBarCodes, IampBarcode::count

3.15. IampBarcode::maxCharCount

The **IampBarcode::maxCharCount** property specifies the maximum number of characters to expect as a result of scanning all the barcodes in a single image.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT put_maxCharCount (** **LONG** *newVal* **);**
C#	**int maxCharCount**
C	**HRESULT IampBarcode_put_maxCharCount (** **LONG** *newVal* **);**

JAVA	**void set_maxCharCount (** **int** *newVal* **);**
VB	**maxCharCount () as Integer;**

Parameters

newVal
This property

Remarks

The default value for this property is 200 and must be larger than the expected total number of characters of all the scanned barcodes in a single image. Some barcode symbologies (PDF 417) are very dense and may contain up to 20,000 characters. As a rule, set this property at least as big as **numberToFind * minLength** plus some additional space for partial barcode scans.

See Also

IampBarcode::ScanBarCodes, IampBarcode::numberToFind, IampBarcode::minLength

3.16. IampBarcode::maxScanTime

The **IampBarcode::maxScanTime** property specifies an upper time limit in hundredths of seconds which may be used when processing barcodes on the image.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT put_maxScanTime (** **LONG** *newVal* **);**
C#	**int maxScanTime**
C	**HRESULT IampBarcode_put_maxScanTime (** **LONG** *newVal* **);**
JAVA	**void set_maxScanTime (** **int** *newVal* **);**
VB	**maxScanTime () as Integer;**

Parameters

newVal
This property establishes the maximum amount of processing time which the computer will spend processing barcodes on the image. The units for this property are hundredths of a second, so for example, the value 400 would be treated as 4.00 seconds.

Remarks

The default value for this property is 200.

See Also

IampBarcode::ScanBarCodes

3.17. IampBarcode::maxToScan

The **IampBarcode::maxToScan** property specifies the maximum number of valid and partial barcodes which may be scanned and reported in a single image.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT put_maxToScan (** **LONG** *newVal* **);**
C#	**int maxToScan**
C	**HRESULT IampBarcode_put_maxToScan (** **LONG** *newVal* **);**
JAVA	**void set_maxToScan (** **int** *newVal* **);**
VB	**maxToScan () as Integer;**

Parameters

newVal
This property must be larger than the **IampBarcode::numberToFind** property.

Remarks

The default value for this property is 100.

See Also

IampBarcode::ScanBarCodes, IampBarcode::numberToFind, IampBarcode::minPartialLength, IampBarcode::count

3.18. IampBarcode::minLength

The **IampBarcode::minLength** property specifies the minimum length barcodes must be before being considered valid.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT put_minLength (** **LONG** *newVal* **);**
C#	**int minLength**
C	**HRESULT IampBarcode_put_minLength (** **LONG** *newVal* **);**
JAVA	**void set_minLength (** **int** *newVal* **);**
VB	**minLength () as Integer;**

Parameters

newVal
This property must be greater than zero.

Remarks

The default value for this property is 1. If the **IampBarcode::fixedLength** property is set to true, then the barcode must contain exactly the number of characters specified by this property to be considered valid. If the property **IampBarcode::doCheckSum** is set to true, the checksum character(s) will be removed from the data and will not be counted in the length of the barcode. An exception will be thrown (E_INVALIDARG) if the new value is < 1.

See Also

IampBarcode::ScanBarCodes, IampBarcode::doChecksum, IampBarcode::fixedLength

3.19. IampBarcode::minPartialLength

The **IampBarcode::minPartialLength** property specifies the minimum number of characters that must be valid to report a partial read of a barcode.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT put_minPartialLength (** **LONG** *newVal* **);**
C#	**int minPartialLength**
C	**HRESULT IampBarcode_put_minPartialLength (** **LONG** *newVal* **);**
JAVA	**void set_minPartialLength (** **int** *newVal* **);**
VB	**minPartialLength () as Integer;**

Parameters

newVal
If newVal = 0, no partial reads will be reported. If newVal > 0, partial reads will be considered valid if the number of scanned characters is greater or equal to this property.

Remarks

The default value for this property is 0. Partial reads do not count towards meeting the **IampBarcode::numberToFind** requirement. They do count against the **IampBarcode::maxToScan** property. When analyzing the barcode properties for scanned barcodes, the **IampBarcode::partial** property indicates if the barcode data is from a partial scan. An exception will be thrown (E_INVALIDARG) if the new value is < 0.

See Also

IampBarcode::ScanBarCodes, IampBarcode::numberToFind, IampBarcode::maxToScan, IampBarcode::partial

3.20. IampBarcode::numberTested

The **IampBarcode::numberTested** property is a value which indicates the number of objects in the image that were scanned as possible barcodes.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT get_numberTested (** **LONG*** *pVal* **);**
C#	**int numberTested**
C	**HRESULT IampBarcode_get_numberTested (** **LONG*** *pVal* **);**
JAVA	**void get_numberTested (** **int*** *pVal* **);**
VB	**numberTested () as Integer;**

Parameters

pVal
A pointer to a variable that will receive the value.

Remarks

This property is only available after calling **IampBarcode::ScanBarCodes**. An exception will be thrown (e_BarcodesNotScanned) if this property is accessed before calling **ScanBarCodes**.

See Also

IampBarcode::ScanBarCodes

3.21. IampBarcode::numberToFind

The **IampBarcode::numberTofind** property specifies how many valid barcodes to report.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT put_numberToFind (** **LONG** *newVal* **);**
C#	**int numberToFind**
C	**HRESULT IampBarcode_put_numberToFind (** **LONG** *newVal* **);**
JAVA	**void set_numberToFind (** **int** *newVal* **);**
VB	**numberToFind () as Integer;**

Parameters

newVal
Valid values for newVal are >= 0.

Remarks

The default value for this property is 0. If the property is set to 0, all valid barcodes on the image will be reported. If the property is > 0, scanning will continue until the specified number of valid barcodes is found or no more barcodes can be found. An exception will be thrown (E_INVALIDARG) if the new value is < 0.

See Also

IampBarcode::ScanBarCodes

3.22. IampBarcode::orientation

The **IampBarcode::orientation** property specifies how the barcodes to be scanned are arranged on the image.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT put_orientation (** **LONG** *newVal* **);**
C#	**int orientation**
C	**HRESULT IampBarcode_put_orientation (** **LONG** *newVal* **);**
JAVA	**void set_orientation (** **int** *newVal* **);**
VB	**orientation () as Integer;**

Parameters

newVal
- 0 Scan only horizontal barcodes
- 1 Scan only vertical barcodes
- 2 Scan both horizontal and vertical barcodes
- 3 Scan only horizontal barcodes with significant skew
- 4 Scan only vertical barcodes with significant skew
- 5 Scan both horizontal and vertical barcodes with significant skew

Remarks

The default value for this property is 0. Horizontal is defined as the vertical bars being perpendicular to the x-axis of the image. Values 0-2 will scan for barcodes with less than a normal amount of skew; generally 10 to 15 degrees. Values 3-5 scan for all barcodes. Knowing the orientation of the barcodes to be scanned can reduce execution times of the scans. An exception will be thrown (E_INVALIDARG) if the new value is not one of these six.

See Also

IampBarcode::ScanBarCodes

3.23. IampBarcode::partial

The **IampBarcode::partial** property indicates the barcode data is an incomplete scan.
In **AmpLibNet**, this property is called **partial_**.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT get_partial (** **VARIANT_BOOL*** *pVal* **);**
C#	**bool partial**
C	**HRESULT IampBarcode_get_partial (** **VARIANT_BOOL*** *pVal* **);**
JAVA	**void get_partial (** **Boolean*** *pVal* **);**
VB	**partial () as Boolean;**

Parameters

pVal
A pointer to a variable that will receive the value.

Remarks

This property is only available after calling **IampBarcode::ScanBarCodes** and setting the **IampBarcode::index** property. An exception will be thrown (e_BarCodesNotScanned) if this property is access prior to calling **ScanBarcodes**.

See Also

IampBarcode::ScanBarCodes, IampBarcode::index

3.24. IampBarcode::partialCount

The **IampBarcode::partialCount** property is a value which indicates the number of objects in the image that were partially decoded.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT get_partialCount (** **LONG*** *pVal* **);**
C#	**int partialCount**
C	**HRESULT IampBarcode_get_partialCount (** **LONG*** *pVal* **);**
JAVA	**void get_partialCount (** **int*** *pVal* **);**
VB	**partialCount () as Integer;**

Parameters

pVal
A pointer to a variable that will receive the value.

Remarks

This property is only available after calling **IampBarcode::ScanBarCodes** and setting the **IampBarcode::index** property. Text and other objects that are not barcodes may partially decode and return unexpected results. Use this property to decide whether the data is valid for the application. An exception will be thrown (e_BarcodesNotScanned) if this property is accessed before calling **ScanBarCodes**.

See Also

IampBarcode::ScanBarCodes, IampBarcode::index

3.25. IampBarcode::prLarge

The **IampBarcode::prLarge** property specifies the priority of scanning for large barcodes on the image. Large barcodes are defined as a 4.0-inch wide by 0.9-inch high barcode.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT put_prLarge (** **LONG** *newVal* **);**
C#	**int prLarge**
C	**HRESULT IampBarcode_put_prLarge (** **LONG** *newVal* **);**
JAVA	**void set_prLarge (** **int** *newVal* **);**
VB	**prLarge () as Integer;**

Parameters

newVal
The valid values are 0-3.

Remarks

The default value for this property is 3. If non-zero values are given for properties **IampBarcode::width** and **IampBarcode::height**, barcodes with the size specified by those two properties will be scanned first. If this property is set to zero, barcodes of this size will not be scanned. Each priority setting (**prLarge**, **prMedium** and **prSmall**) should be unique unless set to zero. Do not set all the priority and **IampBarcode::width** and **IampBarcode::height** properties to zero. An invalid argument error (47) will occur when **ScanBarcodes** is called. An exception will be thrown (E_INVALIDARG) if the new value is not 0-3 or not unique.

See Also

IampBarcode::ScanBarCodes, IampBarcode::prMedium, IampBarcode::prSmall, IampBarcode::width, IampBarcode::height

3.26. IampBarcode::prMedium

The **IampBarcode::prMedium** property specifies the priority of scanning for medium barcodes on the image. Medium barcodes are defined as a 1.7-inch wide by 0.5-inch high barcode.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT put_prMedium (** **LONG** *newVal* **);**
C#	**int prMedium**
C	**HRESULT IampBarcode_put_prMedium (** **LONG** *newVal* **);**
JAVA	**void set_prMedium (** **int** *newVal* **);**
VB	**prMedium () as Integer;**

Parameters

newVal
The valid values are 0-3.

Remarks

The default value for this property is 2. If non-zero values are given for properties **IampBarcode::width** and **IampBarcode::height**, barcodes with the size specified by those two properties will be scanned first. If this property is set to zero, barcodes of this size will not be scanned. Each priority setting (**prLarge**, **prMedium** and **prSmall**) should be unique unless set to zero. Do not set all the priority and **IampBarcode::width** and **IampBarcode::height** properties to zero. An invalid argument error (47) will occur. An exception will be thrown (E_INVALIDARG) if the new value is not 0-3 or not unique.

See Also

IampBarcode::ScanBarCodes, IampBarcode::prLarge, IampBarcode::prSmall, IampBarcode::width, IampBarcode::height

3.27. IampBarcode::prSmall

The **IampBarcode::prSmall** property specifies the priority of scanning for small barcodes on the image. Small barcodes are defined as a 0.9-inch wide by 0.25-inch high barcode.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT put_prSmall (** **LONG** *newVal* **);**
C#	**int prSmall**
C	**HRESULT IampBarcode_put_prSmall (** **LONG** *newVal* **);**
JAVA	**void set_prSmall (** **int** *newVal* **);**
VB	**prSmall () as Integer;**

Parameters

newVal
The valid values are 0-3.

Remarks

The default value for this property is 1. If non-zero values are given for properties **IampBarcode::width** and **IampBarcode::height**, barcodes with the size specified by those two properties will be scanned first. If this property is set to zero, barcodes of this size will not be scanned. Each priority setting (**prLarge**, **prMedium** and **prSmall**) should be unique unless set to zero. Do not set all the priority and **IampBarcode::width** and **IampBarcode::height** properties to zero. An invalid argument error (47) will occur. An exception will be thrown (E_INVALIDARG) if the new value is not 0-3 or not unique.

See Also

IampBarcode::ScanBarCodes, IampBarcode::prLarge, IampBarcode::prMedium, IampBarcode::width, IampBarcode::height

3.28. IampBarcode::quality

The **IampBarcode::quality** property specifies how hard the scanning process will try to decode barcodes.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT put_quality (** **LONG** *newVal* **);**
C#	**int quality**
C	**HRESULT IampBarcode_put_quality (** **LONG** *newVal* **);**
JAVA	**void set_quality (** **int** *newVal* **);**
VB	**quality () as Integer;**

Parameters

newVal
The valid values are 0-10.

Remarks

The default value for this property is 1. This property indicates, to the scanning process, the quality of the image to be scanned. Low numbers are for low quality images and the scanning process will try harder to decode valid barcodes. Higher numbers are for better quality images and the scanning process will tighten the criteria for reporting valid barcodes. These are some guidelines for image quality:

1 FAX input
4 Microfilm scan
7 Second generation copy
10 First generation print

An exception will be thrown (E_INVALIDARG) if the new value is not 0-10.

See Also

IampBarcode::ScanBarCodes

3.29. IampBarcode::reversed

The **IampBarcode::reversed** property indicates the direction the barcode was scanned.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT get_reversed (** **VARIANT_BOOL*** *pVal* **);**
C#	**bool reversed**
C	**HRESULT IampBarcode_get_reversed (** **VARIANT_BOOL*** *pVal* **);**
JAVA	**void get_reversed (** **Boolean*** *pVal* **);**
VB	**reversed () as Boolean;**

Parameters

pVal
A pointer to a variable that will receive the value.

Remarks

This property is only available after calling **IampBarcode::ScanBarCodes** and setting the **IampBarcode::index** property. An exception will be thrown (e_BarcodesNotScanned) if this property is accessed before calling **ScanBarCodes**.

See Also

IampBarcode::ScanBarCodes, IampBarcode::index

3.30. IampBarcode::rotated

The **IampBarcode::rotated** property indicates the direction the barcode was scanned.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT get_rotated (** **VARIANT_BOOL*** *pVal* **);**
C#	**bool rotated**
C	**HRESULT IampBarcode_get_rotated (** **VARIANT_BOOL*** *pVal* **);**
JAVA	**void get_rotated (** **Boolean*** *pVal* **);**
VB	**rotated () as Boolean;**

Parameters

pVal
A pointer to a variable that will receive the value.

Remarks

This property is only available after calling **IampBarcode::ScanBarCodes** and setting the **IampBarcode::index** property. An exception will be thrown (e_BarcodesNotScanned) if this property is accessed before calling **ScanBarCodes**.

See Also

IampBarcode::ScanBarCodes, IampBarcode::index

3.31. IampBarcode::symbology

The **IampBarcode::symbology** property is a value which indicates the type of barcode that was scanned.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT get_symbology (** **LONG*** *pVal* **);**
C#	**int symbology**
C	**HRESULT IampBarcode_get_symbology (** **LONG*** *pVal* **);**
JAVA	**void get_symbology (** **int*** *pVal* **);**
VB	**symbology () as Integer;**

Parameters

pVal
A pointer to a variable that will receive the value.

Remarks

This property is only available after calling **IampBarcode::ScanBarCodes** and setting the **IampBarcode::index** property. The property will be one of the enumerated values in Appendix B. ????? Need list of the symbologies in Appendix B and a link there. An exception will be thrown (e_BarcodesNotScanned) if this property is accessed before calling **ScanBarCodes**.

See Also

IampBarcode::ScanBarCodes, IampBarcode::index

3.32. IampBarcode::symbologyMask

The **IampBarcode::symbologyMask** property specifies which barcode symbologies the scanning process will attempt to decode.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT put_symbologyMask (** **LONG** *newVal* **);**
C#	**int symbologyMask**
C	**HRESULT IampBarcode_put_symbologyMask (** **LONG** *newVal* **);**
JAVA	**void set_symbologyMask (** **int** *newVal* **);**
VB	**symbologyMask () as Integer;**

Parameters

newVal
The valid values are in Appendix B. To select more than one barcode symbology, use a sum of the values. ????? Need to link to Appendix B.

Remarks

The default value for this property is 0 and is invalid. This property **must** be set prior to calling the **IampBarcode::ScanBarCodes** method. Some symbologies are incompatible when selected at the same time. EAN_13 is a superset of UPC_A and barcodes will always decode as EAN_13 instead of UPC_A. An exception will be thrown (e_InvalidSymbology) if the new mask is invalid.

See Also

IampBarcode::ScanBarCodes

3.33. IampBarcode::text

The **IampBarcode::text** property is a string property which contains the characters of barcode that was scanned. ????? Do I need a charct property in case barcodes CONTAIN a NULL character?

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT get_text (** **BSTR*** *pVal* **);**
C#	**String text**
C	**HRESULT IampBarcode_get_text (** **BSTR*** *pVal* **);**
JAVA	**void get_text (** **java.lang.String*** *pVal* **);**
VB	**text () as Integer;**

Parameters

pVal
A pointer to a variable that will receive the value.

Remarks

This property is only available after calling **IampBarcode::ScanBarCodes** and setting the **IampBarcode::index** property. An exception will be thrown (e_BarcodesNotScanned) if this property is accessed before calling **ScanBarCodes**. An exception will be thrown (e_BarcodeTextEmpty) if there is no barcode text to return. An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating memory for variables within this method.

See Also

IampBarcode::ScanBarCodes, IampBarcode::index

3.34. IampBarcode::width

The **IampBarcode::width** property specifies the width of barcodes to be scanned in thousandths of an inch.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT put_width (** **LONG** *newVal* **);**
C#	**int width**
C	**HRESULT IampBarcode_put_width (** **LONG** *newVal* **);**
JAVA	**void set_width (** **int** *newVal* **);**
VB	**width () as Integer;**

Parameters

newVal
If newVal = 0, this property is ignored. If this property is > 0, the specified value is thousandths of an inch.

Remarks

The default value for this property is 0. Only use this property if the size of the barcodes to be scanned can be approximated. When used with the **prLarge**, **prMedium** and **prSmall** properties, a variety of barcode sizes can be scanned.

See Also

IampBarcode::ScanBarCodes, IampBarcode::height, IampBarcode::prLarge, IampBarcode::prMedium, IampBarcode::prSmall

3.35. IampBarcode::xOrigin

The **IampBarcode::symbology** property is a value in pixels which indicates the left point of origin of the scanned barcode on the image.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT get_xOrigin (** **LONG*** *pVal* **);**

C#	**int xOrigin**
C	**HRESULT IampBarcode_get_xOrigin (** **LONG*** *pVal* **);**
JAVA	**void get_xOrigin (** **int*** *pVal* **);**
VB	**xOrigin () as Integer;**

Parameters

pVal
A pointer to a variable that will receive the value.

Remarks

This property is only available after calling **IampBarcode::ScanBarCodes** and setting the **IampBarcode::index** property. xOrigin and yOrigin can be used to identify where on the scanned image the barcode is located and can be used to determine the meaning of the data when multiple barcodes are expected to be scanned on a single image. If a region of interest is specified using the **IampImage::SetWindow** method, then the values are relative to the region and not to the entire image. An exception will be thrown (e_BarcodesNotScanned) if this property is accessed before calling **ScanBarCodes**.

See Also

IampBarcode::ScanBarCodes, IampBarcode::index, IampBarcode::yOrigin, IampImage::SetWindow

3.36. IampBarcode::yOrigin

The **IampBarcode::yOrigin** property is a value in pixels which indicates the top point of origin of the scanned barcode on the image.

Quick Info

See IampBarcode : IDispatch.

C++	**HRESULT get_yOrigin (** **LONG*** *pVal* **);**
C#	**int yOrigin**
C	**HRESULT IampBarcode_get_yOrigin (** **LONG*** *pVal* **);**
JAVA	**void get_yOrigin (** **int*** *pVal* **);**
VB	**yOrigin () as Integer;**

Parameters

pVal
A pointer to a variable that will receive the value.

Remarks

This property is only available after calling **IampBarcode::ScanBarCodes** and setting the **IampBarcode::index** property. xOrigin and yOrigin can be used to identify where on the scanned image the barcode is located and can be

used to determine the meaning of the data when multiple barcodes are expected to be scanned on a single image. If a region of interest is specified using the **IampImage::SetWindow** method, then the values are relative to the region and not to the entire image. An exception will be thrown (e_BarcodesNotScanned) if this property is accessed before calling **ScanBarCodes**.

See Also

IampBarcode::ScanBarCodes, IampBarcode::index, IampBarcode::xOrigin, IampImage::SetWindow

4. IampMICR: IDispatch

The **IampMICR** interface provides a method for reading the MICR line on scanned check images. It is implemented within the ampImage class and is provided as a separate interface.

Quick Info

Header file:	ampImage.h
Interface identifier:	IID_IampMICR
Pointer type:	ampImage*

Vtable

Methods	Descriptions
AssembleMICR	Assembles a MICR string from components.
DetectCoupon	Detects and reads remittance coupons.
ReadCamera	Read the MICR line from camera check images with output of formatted thresholded images.
ReadCameraEx	Read the MICR line from camera check images with output of formatted thresholded images and formatted grayscale images.
ReadMICR	Reads the MICR, OCR-A, or OCR-B line in the image.
ReadMICRPage	Reads the MICR line in the image and returns the region where it is located.
ReadScanner	Read the MICR line from full-page scanned check images and isolates the check image.
FormatMICRFields	Formats a MICR string.
GetMoreReadCameraResults	Gets additional ReadCamera status results
GetReadCameraResults	Gets ReadCamera status results.
GetReadScannerResults	Gets ReadScanner status results.
GetFieldVerifyResultData	Gets engine information after a Verify operation.
GetFieldVerifyResultMinConf	Gets MICR field confidence minima after a Verify operation.
getMICRData	Gets character-specific information after a MICR read operation.
getMICRFields	Extracts components from a MICR string.
getMICRRegion	Returns MICR region pixel boundaries.
getRemitAlternateAmount	Returns a low-confidence alternative remittance amount.
getRemitAlternateCheckNumber	Returns a low-confidence alternative remittance check number.
getRemitAlternateDate	Returns a low-confidence alternative remittance date.
getRemitAmountConfidence	Returns the remittance amount confidence value.

getRemitAmountRegion	Returns remittance amount region pixel boundaries.
getRemitCheckNumberConfidence	Returns the remittance check number confidence value.
getRemitCheckNumberRegion	Returns remittance check number region pixel boundaries.
getRemitCheckRegion	Returns remittance check region pixel boundaries.
getRemitDateConfidence	Returns the remittance date confidence value.
getRemitDateRegion	Returns remittance date region pixel boundaries.
getVerifyMICRData	Gets character-specific information after a MICR Verify operation.
PrepareCouponImage	Crops and deskews coupon, check, and other various types of images.
PrepareMICRImage	Crops and deskews a check shaped image.
ReadMICRRemit	Reads a remittance document.
SetAppDirectory	Specifies the install directory containing “storage.dat”.
SetBottomCropDistance	Distance, in inches, from the bottom of the image to ignore during MICR reading.
SetComboMode	Runs MICR reading without enhancements, then reads with enhancements only if fewer than comboChars characters are scanned successfully.
SetDo180	Specifies whether to check for upside down MICR images.
SetDoImagePrep	Sets image deskewing and border removal.
SetDoImageRepair	Sets special image enhancements; usually off.
SetDoImageUpdate	Sets whether to update the image object with user-selected image enhancements.
SetDoPrescale	Sets whether to prescale images.
SetDoRepair	Sets whether to repair degraded characters.
SetFieldVerifyMinConf	Sets the minimum successful confidence value for MICR fields during a Verify operation.
SetImageFilter	Sets the image filter to use.
SetMICRCode	Sets the MICR code type.
SetMICRMinConf	Sets the minimum acceptable character confidence score.
SetMICRRules	Sets the type of MICR rules.
SetNoBlanks	Sets whether to report MICR blanks.
SetNoRules	Sets whether to ignore MICR rules.
SetOCRAEnable	Configures OCR-A reading settings.
SetOCRBEnable	Configures OCR-B reading settings.
SetResolution	Sets image resolution determination method.
SetRotate	Sets image rotation.
SetTimeout	Sets maximum allowed scan time, in seconds.

VerifyMICR	Verifies characters from a previously-read MICR line.
VerifyMICRField	Verifies fields from a previously-read MICR line.

Properties	Access
enableCameraMode	Read/Write.
enableFullPage	Read/Write.
enableScannerMode	Read/Write.
done180	Read only.
doneCombo	Read only.
doneImageUpdate	Read only.
doneRepair	Read only.
enableReadRemit	Read/Write.
OCRCode	Read/Write.
skew	Read only.

Remarks

The **IampMICR** interface inherits directly from **IDispatch**. The symbologyMask property must be set before calling **ReadMICR**. The other properties will use default values specified in the following sections unless modified prior to calling **ReadMICR**. The read only properties are only valid after calling **ReadMICR** and setting the index property.

4.2. IampMICR::ReadCamera

The **IampMICR::ReadCamera** method uses special grayscale image processing to read the MICR line in check images that come from cameras and produces bilevel 200 dpi output images.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT ReadCamera (** **LONG** *nFrontInputImage,* **LONG** *nBackInputImage,* **LONG** *nFrontOutputImage,* **LONG** *nBackOutputImage,* **BSTR** *options,* **BSTR*** *pszMICR,* **LONG*** *nCount,* **LONG*** *ampResult* **);**
C#	**int ReadCamera (** **int** *nFrontInputImage,* **int** *nBackInputImage,* **int** *nFrontOutputImage,* **int** *nBackOutputImage* **string** *options,* **out string** *pszMICR,* **out int** *nCount* **);**

C	**HRESULT IampMICR_ReadCamera (** **LONG** *nFrontInputImage,* **LONG** *nBackInputImage,* **LONG** *nFrontOutputImage,* **LONG** *nBackOutputImage,* **BSTR** *options,* **BSTR*** *pszMICR,* **LONG*** *nCount,* **LONG*** *ampResult* **);**
JAVA	**int ReadCamera (** **int** *nFrontInputImage,* **int** *nBackInputImage,* **int** *nFrontOutputImage,* **int** *nBackOutputImage,* **String** *options,* **String** *outputMICR,* **int** *outputCount* **);**
VB	**ReadCamera (** *nFrontInputImage* **as Integer,** *nBackInputImage* **as Integer,** *nFrontOutputImage* **as Integer,** *nBackOutputImage* **as Integer,** *options* **as String,** *pszMICR* **as String,** *nCount* **as Integer** **)** **as Integer;**

Parameters

nFrontInputImage
The image buffer 0-9 to use as the source image for the front of the check. A value of -1 will cause the contents of the main image property to be used. The source image must be grayscale.

nBackInputImage
The image buffer 0-9 to use as the source image for the back of the check. A value of -1 tells the method there is no back image to use.

nFrontOutputImage
The image buffer 0-9 to use as the destination image for the dewarped, resized, and thresholded front check image. A value of -1 tells the method that no output front image is needed.

nBackOutputImage
The image buffer 0-9 to use as the destination image for the dewarped, resized, and thresholded back check image. A value of -1 tells the method that no output back image is needed.

options
A string of single character commands that select cropping (C), trapezoidal warp detection and correction (T), 180 degree rotation detection and correction (R), blank suppression disable (B), and confidence processing (M=nn where nn defaults to 85). The default values for CTRB are off. Use the “C” option if the input image has not been previously cropped and corrected. Use the “T” option for camera images or if the source is unknown. The “T” option is not required for page scanner images

pszMICR
The string of MICR characters recognized from the selected front check.

nCount
The number of MICR characters recognized.

ampResult
The error code returned. Error codes are defined in Appendix A.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

The selected input image(s) must be loaded prior to reading MICR characters. Once **ReadCamera** has been called, the number of characters found can be accessed through the **nCount** value. The MICR characters for the image can be accessed through the pszMICR value. An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating variables within this method.

See Also

Appendix A

4.3. IampMICR::ReadCameraEx

The **IampMICR::ReadCameraEx** method uses special grayscale image processing to read the MICR line in check images that come from cameras and produces bilevel and grayscale 200 dpi output images.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT ReadCamera (** **LONG** *nFrontInputImage,* **LONG** *nBackInputImage,* **LONG** *nFrontOutputImage,* **LONG** *nBackOutputImage,* **BSTR** *options,* **BSTR*** *pszMICR,* **LONG*** *nCount,* **LONG** *nFrontOutputGrayImage,* **LONG** *nBackOutputGrayImage,* **LONG*** *ampResult* **);**
C#	**int ReadCamera (** **int** *nFrontInputImage,* **int** *nBackInputImage,* **int** *nFrontOutputImage,* **int** *nBackOutputImage* **string** *options,* **out string** *pszMICR,* **out int** *nCount* **int** *nFrontOutputGrayImage,* **int** *nBackOutputGrayImage* **);**
C	**HRESULT IampMICR_ReadCamera (** **LONG** *nFrontInputImage,* **LONG** *nBackInputImage,* **LONG** *nFrontOutputImage,* **LONG** *nBackOutputImage,* **BSTR** *options,* **BSTR*** *pszMICR,* **LONG*** *nCount,* **LONG** *nFrontOutputGrayImage,* **LONG** *nBackOutputGrayImage,* **LONG*** *ampResult* **);**

JAVA	**int ReadCamera (** **int** *nFrontInputImage,* **int** *nBackInputImage,* **int** *nFrontOutputImage,* **int** *nBackOutputImage,* **String** *options,* **String** *outputMICR,* **int** *outputCount* **int** *nFrontOutputGrayImage,* **int** *nBackOutputGrayImage,* **);**
VB	**ReadCamera (** *nFrontInputImage* **as Integer,** *nBackInputImage* **as Integer,** *nFrontOutputImage* **as Integer,** *nBackOutputImage* **as Integer,** *options* **as String,** *pszMICR* **as String,** *nCount* **as Integer** *nFrontOutputGrayImage* **as Integer,** *nBackOutputGrayImage* **as Integer,** **)** **as Integer;**

Parameters

nFrontInputImage
The image buffer 0-9 to use as the source image for the front of the check. A value of -1 will cause the contents of the main image property to be used. The source image must be grayscale.

nBackInputImage
The image buffer 0-9 to use as the source image for the back of the check. A value of -1 tells the method there is no back image to use.

nFrontOutputImage
The image buffer 0-9 to use as the destination image for the dewarped, resized, and thresholded front check image. A value of -1 tells the method that no output front image is needed.

nBackOutputImage
The image buffer 0-9 to use as the destination image for the dewarped, resized, and thresholded back check image. A value of -1 tells the method that no output back image is needed.

options
A string of single character commands that select cropping (C), trapezoidal warp detection and correction (T), 180 degree rotation detection and correction (R), blank suppression disable (B), and confidence processing (M=nn where nn defaults to 85). The default values for CTRB are off. Use the “C” option if the input image has not been previously cropped and corrected. Use the “T” option for camera images or if the source is unknown. The “T” option is not required for page scanner images

pszMICR
The string of MICR characters recognized from the selected front check.

nCount
The number of MICR characters recognized.

nFrontOutputGrayImage
The image buffer 0-9 to use as the destination image for the dewarped, and resized grayscale front check image. A value of -1 tells the method that no output grayscale front image is needed.

nBackOutputGrayImage
The image buffer 0-9 to use as the destination image for the dewarped, and resized grayscale back check image. A value of -1 tells the method that no output grayscale back image is needed.

ampResult
The error code returned. Error codes are defined in Appendix A.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

The selected input image(s) must be loaded prior to reading MICR characters. Once **ReadCameraEx** has been called, the number of characters found can be accessed through the **nCount** value. The MICR characters for the image can be accessed through the pszMICR value. An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating variables within this method.

See Also

Appendix A

4.4. IampMICR::ReadMICR

The **IampMICR::ReadMICR** method scans the loaded image for a line of MICR, OCR-A, or OCR-B characters.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT ReadMICR (** **BSTR*** *pszMICR,* **LONG*** *nCount,* **LONG*** *ampResult* **);**
C#	**int ReadMICR (** **out string** *pszMICR,* **out int** *nCount* **);**
C	**HRESULT IampMICR_ReadMICR (** **BSTR*** *pszMICR,* **LONG*** *nCount,* **LONG*** *ampResult* **);**
JAVA	**int ReadMICR (** **String** *outputMICR,* **int** *outputCount* **);**
VB	**ReadMICR (** *pszMICR* **as String,** *nCount* **as Integer** **)** **as Integer;**

Parameters

pszMICR
The string of MICR characters recognized from the selected front check.

nCount
The number of MICR characters recognized.

ampResult
The error code returned. Error codes are defined in Appendix A.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

The image must be loaded and the **symbologyMask** property must be set prior to reading MICR characters from an image. An exception will be thrown (e_SymbologyMaskNotSet) if this method is called prior to setting the mask. All other properties have default values as described in the respective sections. Once **ReadMICR** has been called, the number of characters found can be accessed through the **count** value. The MICR characters for the image can be accessed through the pszMICR value. An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating variables within this method.

Check images that have been acquired from a camera are frequently warped in a keystone fashion and have poor grayscale content. If the CameraMode property is enabled, ReadMICR will use special image processing to dewarp and threshold the image prior to reading the MICR line.

See Also

Appendix A

4.5. IampMICR::ReadMICRPage

The **IampMICR::ReadMICRPage** method scans the loaded image for a MICR routing field and then builds a region of interest that contains the entire line of MICR characters. The character shapes within the region of interest are then read in a fashion similar to ReadMICR.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT ReadMICRPage (** **BSTR*** *pszMICR,* **LONG*** *nCount,* **LONG*** *ampResult* **);**
C#	**int ReadMICRPage (** **out string** *pszMICR,* **out int** *nCount* **);**
C	**HRESULT IampMICR_ReadMICRPage (** **BSTR*** *pszMICR,* **LONG*** *nCount,* **LONG*** *ampResult* **);**
JAVA	**int ReadMICRPage (** **String** *outputMICR,* **int** *outputCount*
VB	**ReadMICRPage (** *pszMICR* **as String,** *nCount* **as Integer** **)** **as Integer;**

Parameters

pszMICR
The string of MICR characters recognized from the selected front check.

nCount
The number of MICR characters recognized.

ampResult
The error code returned. Error codes are defined in Appendix A.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

The image must be loaded and the **symbologyMask** property must be set prior to reading MICR characters from an image. An exception will be thrown (e_SymbologyMaskNotSet) if this method is called prior to setting the mask. All other properties have default values as described in the respective sections. Once **ReadMICRPage** has been called, the number of characters found can be accessed through the **count** value. The MICR characters for the image can be accessed through the pszMICR value. The extent of the MICR region of interest can be obtained through the getMICRRegion method. An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating variables within this method.

See Also

IampMICR::getMICRRegion, Appendix A

4.6. IampMICR::ReadScanner

The **IampMICR::ReadScanner** method uses special full page image processing to read the MICR line in check images that come from scanners.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT ReadScanner (** **LONG** *nFrontInputImage,* **LONG** *nBackInputImage,* **LONG** *nFrontOutputImage,* **LONG** *nBackOutputImage,* **BSTR** *options,* **BSTR*** *pszMICR,* **LONG*** *nCount,* **LONG*** *ampResult* **);**
C#	**int ReadScanner (** **int** *nFrontInputImage,* **int** *nBackInputImage,* **int** *nFrontOutputImage,* **int** *nBackOutputImage* **string** *options,* **out string** *pszMICR,* **out int** *nCount* **);**

C	**HRESULT IampMICR_ReadScanner (** **LONG** *nFrontInputImage,* **LONG** *nBackInputImage,* **LONG** *nFrontOutputImage,* **LONG** *nBackOutputImage,* **BSTR** *options,* **BSTR*** *pszMICR,* **LONG*** *nCount,* **LONG*** *ampResult* **);**
JAVA	**int ReadScanner (** **int** *nFrontInputImage,* **int** *nBackInputImage,* **int** *nFrontOutputImage,* **int** *nBackOutputImage,* **String** *options,* **String** *outputMICR,* **int** *outputCount* **);**
VB	**ReadScanner (** *nFrontInputImage* **as Integer,** *nBackInputImage* **as Integer,** *nFrontOutputImage* **as Integer,** *nBackOutputImage* **as Integer,** *options* **as String,** *pszMICR* **as String,** *nCount* **as Integer** **)** **as Integer;**

Parameters

nFrontInputImage
The image buffer 0-9 to use as the source image for the front of the check. A value of -1 will cause the contents of the main image property to be used.

nBackInputImage
The image buffer 0-9 to use as the source image for the back of the check. A value of -1 tells the method there is no back image to use.

nFrontOutputImage
The image buffer 0-9 to use as the destination image for the dewarped, resized, and thresholded front check image. A value of -1 tells the method that no output front image is needed.

nBackOutputImage
The image buffer 0-9 to use as the destination image for the dewarped, resized, and thresholded back check image. A value of -1 tells the method that no output back image is needed.

options
A string of single character commands that select cropping (C), trapezoidal warp detection and correction (T), 180 degree rotation detection and correction (R), blank suppression disable (B), and confidence processing (M=nn where nn defaults to 85). The default values for CTRB are off. Use the “C” option if the input image has not been previously cropped and corrected. Use the “T” option for camera images or if the source is unknown. The “T” option is not required for page scanner images

pszMICR
The string of MICR characters recognized from the selected front check.

nCount
The number of MICR characters recognized.

ampResult
The error code returned. Error codes are defined in Appendix A.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

ReadScanner is useful for isolating check images that have been scanned from a full page flatbed scanner where the check may be at the top, middle or bottom of the bitmap source image. The selected input image(s) must be loaded prior to reading MICR characters. Once **ReadScanner** has been called, the number of characters found can be accessed through the **nCount** value. The MICR characters for the image can be accessed through the pszMICR value. An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating variables within this method.

See Also

IampMICR::GetReadScannerResults

4.7. IampMICR::SetAppDirectory

The **IampMICR::SetAppDirectory** method sets the AMPLib install directory containing "storage.dat".

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT SetAppDirectory (** **BSTR** *pszAppDirectory* **);**
C#	**void SetAppDirectory (** **string** *pszAppDirectory* **);**
C	**HRESULT IampImage_SetAppDirectory (** **BSTR** *pszAppDirectory* **);**
JAVA	**void SetAppDirectory (** **String** *pszAppDirectory* **);**
VB	**SetAppDirectory (** *pszAppDirectory* **as String** **)**

Parameters

pszAppDirectory
Specifies the AMPLib install directory containing "storage.dat".

Remarks

If the default install directory on the C: drive is used, this method is unnecessary. The value will be preset to "C:\Program Files\AllMyPapers\AmpLib\bin\"

4.8. IampMICR::SetBottomCropDistance

The **IampMICR::SetBottomCropDistance** method sets a region on the bottom of the check image to be ignored during MICR reading.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT SetBottomCropDistance (** **double** *dblCropDistance* **);**
C#	**void SetBottomCropDistance (** **double** *dblCropDistance* **);**
C	**HRESULT IampImage_SetBottomCropDistance (** **double** *dblCropDistance* **);**
JAVA	**void SetBottomCropDistance (** **double** *dblCropDistance* **);**
VB	**SetBottomCropDistance (** *dblCropDistance* **as Double** **)**

Parameters

dblCropDistance
Specifies the crop distance from the bottom of the check, **in inches**, to ignore during MICR reading.

Remarks

This method sets a region at the bottom of the check to ignore during MICR reading.

4.9. IampMICR::SetComboMode

The **IampMICR::SetComboMode** method controls the settings for combo mode. Combo Mode uses a fast MICR reading algorithm to read a character sequence, then an enhanced MICR reading algorithm if the fast attempt did not successfully read enough characters.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT SetComboMode (** **int** *nEnableCombo*, **int** *nComboChars* **);**
C#	**void SetComboMode (** **int** *nEnableCombo*, **int** *nComboChars* **);**

C	**HRESULT IampImage_SetComboMode (** **int** *nEnableCombo*, **int** *nComboChars* **);**
JAVA	**void SetComboMode (** **int** *nEnableCombo*, **int** *nComboChars* **);**
VB	**SetComboMode (** *nEnableCombo* **as Integer**, *nComboChars* **as Integer** **)**

Parameters

nEnableCombo
Activates Combo Mode.

nComboChars
Sets the minimum successful character reads needed to skip the enhanced reading attempt.

Remarks

Combo mode can be used to apply enhanced scan techniques only where necessary, reducing processing time for groups of check images of varying quality.

4.10. IampMICR::SetDo180

The **IampMICR::SetDo180** method turns Do180 mode on or off. With Do180 mode on, images will be rotated 180 degrees and rescanned if a MICR line cannot be detected in the original.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT SetDo180 (** **int** *nDo180* **);**
C#	**void SetDo180 (** **int** *nDo180* **);**
C	**HRESULT IampImage_SetDo180(** **int** *nDo180* **);**
JAVA	**void SetDo180 (** **int** *nDo180* **);**
VB	**SetDo180 (** *nDo180* **as Integer** **)**

Parameters

pszAppDirectory
1 turns Do180 mode on, 0 turns it off.

Remarks

An unsuccessful MICR read attempt is determined by a result containing fewer than 4 correct characters.

4.11. IampMICR::SetDoImagePrep

The **IampMICR::SetDoImagePrep** method configures image deskewing and border removal.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT SetDoImagePrep (** **int** *nImagePrep*, **int** *nBlackEdge* **);**
C#	**void SetDoImagePrep (** **int** *nImagePrep*, **int** *nBlackEdge* **);**
C	**HRESULT IampImage_SetDoImagePrep(** **int** *nImagePrep*, **int** *nBlackEdge* **);**
JAVA	**void SetDoImagePrep (** **int** *nImagePrep*, **int** *nBlackEdge* **);**
VB	**SetDoImagePrep (** *nImagePrep* **as Integer,** *nBlackEdge* **as Integer** **)**

Parameters

nImagePrep
1 turns deskewing and border removal on, 0 turns it off.

nBlackEdge
1 turns black border removal mode on, 0 turns white border removal on.

Remarks

See the AMPPrepMICR entry in the AMPLib Manual for more information.

See Also

IampMICR::SetDoImageRepair

4.12. IampMICR::SetDoImageRepair

The **IampMICR::SetDoImageRepair** method allows the image object to be processed with special image enhancements.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT SetDoImageRepair (** **int** *nDoImageRepair* **);**
C#	**void SetDoImageRepair (** **int** *nDoImageRepair* **);**
C	**HRESULT IampImage_SetDoImageRepair(** **int** *nDoImageRepair* **);**
JAVA	**void SetDoImageRepair (** **int** *nDoImageRepair* **);**
VB	**SetDoImageRepair (** *nDoImageRepair* **as Integer** **)**

Parameters

nDoImageRepair
1 turns DoImageRepair mode on, 0 turns it off.

Remarks

DoImageRepair mode allows the image object to be updated with special image processing.

This is usually set to 0.

See Also

IampMICR::SetDoRepair

4.13. IampMICR::SetDoImageUpdate

The **IampMICR::SetDoImageUpdate** method allows the image object to be updated with user-selected image enhancements.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT SetDoImageUpdate (** **int** *nDoImageUpdate* **);**

C#	**void SetDoImageUpdate (** **int** *nDoImageUpdate* **);**
C	**HRESULT IampImage_SetDoImageUpdate (** **int** *nDoImageUpdate* **);**
JAVA	**void SetDoImageUpdate (** **int** *nDoImageUpdate* **);**
VB	**SetDoImageUpdate (** *nDoImageUpdate* **as Integer** **)**

Parameters

nDoImageUpdate
1 turns image update mode on, 0 turns it off.

Remarks

Image update mode allows the image object to be updated with user-selected image enhancements.

If Combo Mode is off, any user-selected deskewing, border removal, and image filtering operations will be applied to the image object.

If Combo Mode is on, user-selected enhancements will be applied only if their application was required to read the MICR data successfully.

See Also

IampMICR::doneImageUpdate

4.14. IampMICR::SetDoPrescale

The **IampMICR::SetDoPrescale** method turns image prescaling on or off. Prescaling effectively doubles the resolution of images with 150 dpi or less.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT SetDoPrescale (** **int** *nPrescale* **);**
C#	**void SetDoPrescale (** **int** *nPrescale* **);**
C	**HRESULT IampImage_SetDoPrescale(** **int** *nPrescale* **);**
JAVA	**void SetDoPrescale (** **int** *nPrescale* **);**

VB	**SetDoPrescale (** *nPrescale* **as Integer** **)**

Parameters

nPrescale
1 turns image prescaling mode on, 0 turns it off.

Remarks

MICR character reading requires a sufficiently high resolution to work well. Image prescaling can accommodate this: if the resolution of the image is 150 dpi or less, image prescaling will double the image resolution.

4.15. IampMICR::SetDoRepair

The **IampMICR::SetDoRepair** method turns image repairing on or off.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT SetDoRepair (** **int** *nDoRepair* **);**
C#	**void SetDoRepair (** **int** *nDoRepair* **);**
C	**HRESULT IampImage_SetDoRepair(** **int** *nDoRepair* **);**
JAVA	**void SetDoRepair (** **int** *nDoRepair* **);**
VB	**SetDoRepair (** *nDoRepair* **as Integer** **)**

Parameters

nDoRepair
1 turns nDoRepair mode on, 0 turns it off.

Remarks

When nDoRepair mode is on:

If any read errors are detected in the result data, then a temporary copy of the input image is repaired based on the results of the first read. The repaired image is then used for a second MICR read. The two MICR read results are then voted upon, and the result of the vote is reported.

When doing image repair and reprocess, the execution times will be about twice as long for images which have read errors in the first pass.

Image repairing is distinct to AMPLib's deskewing, border removal, and image filtration functions.

See Also

IampMICR::SetDoImageRepair

4.16. IampMICR::SetFieldVerifyMinConf

The **IampMICR::SetFieldVerifyMinConf** method sets the minimum acceptable confidence level for each section of a MICR line. If the confidence value for any field does not meet the specified minimum during a MICR Verify operation, the operation will be unsuccessful.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT SetFieldVerifyMinConf (** **int** *nCheck*, **int** *nRoute*, **int** *nAccount*, **int** *nAmount*, **int** *nFieldMICRMinConf* **);**
C#	**void SetFieldVerifyMinConf (** **int** *nCheck*, **int** *nRoute*, **int** *nAccount*, **int** *nAmount*, **int** *nFieldMICRMinConf* **);**
C	**HRESULT IampImage_SetFieldVerifyMinConf (** **int** *nCheck*, **int** *nRoute*, **int** *nAccount*, **int** *nAmount*, **int** *nFieldMICRMinConf* **);**
JAVA	**void SetFieldVerifyMinConf (** **int** *nCheck*, **int** *nRoute*, **int** *nAccount*, **int** *nAmount*, **int** *nFieldMICRMinConf* **);**
VB	**SetFieldVerifyMinConf (** **int** *nCheck* **as Integer,** **int** *nRoute* **as Integer,** **int** *nAccount* **as Integer,** **int** *nAmount* **as Integer,** **int** *nFieldMICRMinConf* **as Integer** **)**

Parameters

nCheck
A 2-digit integer value that sets the minimum acceptable confidence value for the Check/AuxOnUs Field during a MICR Verify operation.

nRoute
A 2-digit integer value that sets the minimum acceptable confidence value for the Route Field during a MICR Verify operation.

nAccount
A 2-digit integer value that sets the minimum acceptable confidence value for the Account/OnUs Field during a MICR Verify operation.

nAmount
A 2-digit integer value that sets the minimum acceptable confidence value for the Amount Field during a MICR Verify operation.

nFieldMICRMinConf
A 2-digit integer value that sets the minimum acceptable confidence value for the entire MICR line during a MICR Verify operation.

4.17. IampMICR::SetImageFilter

The **IampMICR::SetImageFilter** method configures background removal image filters.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT SetImageFilter (** **int** *nEnable*, **int** *nType*, **int** *nThreshold* **);**
C#	**void SetImageFilter (** **int** *nEnable*, **int** *nType*, **int** *nThreshold* **);**
C	**HRESULT IampImage_SetImageFilter (** **int** *nEnable*, **int** *nType*, **int** *nThreshold* **);**
JAVA	**void SetImageFilter (** **int** *nEnable*, **int** *nType*, **int** *nThreshold* **);**
VB	**SetImageFilter (** *nEnable* **as Integer,** *nType* **as Integer,** *nThreshold* **as Integer** **)**

Parameters

nEnable
1 turns image filtering on, 0 turns it off.

nType
Sets the filter type. Valid filters are 200, 201, and 202.

nThreshold
Sets the background removal strength. Valid values are 1 (weakest) to 9 (strongest).

Remarks

nType can be set to one of three filters: 200, 201, and 202.

These filters will remove the background from a large class of images and leave the foreground (text) information. Each one will also accept a threshold value to increase the degree of removal.

200 uses the lower 5/8" of the image to determine background (MICR line on a check).

201 will perform the removal filter regardless as no test is performed.

202 uses the middle 1/3rd of the image to determine background (CAR/LAR region on a check).

200 and 202 also perform a test to determine if background noise is present before running the removal filter. If not present no filter operation is performed.

The strength or amount of background removal is determined by the threshold value from 1 to 9 with 9 being the most aggressive background removal and 1 the least. When used for preparing an image for OCR, the aggressive value of 8 is often used. When the same image is being prepared for printing, the value of 6 is often used. This will leave some noise on the image but will not impact the small text content of the image.

4.18. IampMICR::SetMICRCode

The **IampMICR::SetMICRCode** method sets the type of MICR code being read: either E13B, or CMC7.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT SetMICRCode (** **int** *nMICRCode* **);**
C#	**void SetMICRCode (** **int** *nMICRCode* **);**
C	**HRESULT IampImage_SetMICRCode(** **int** *nMICRCode* **);**
JAVA	**void SetMICRCode (** **int** *nMICRCode* **);**
VB	**SetMICRCode (** *nMICRCode* **as Integer** **)**

Parameters

nMICRCode
1 = E13B

2 = CMC7

See Also

IampMICR::SetMICRRules

4.19. IampMICR::SetMICRMinConf

The **IampMICR::SetMICRMinConf** method sets the minimum confidence value to accept a MICR character read attempt as successful.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT SetMICRMinConf (** **int** *nMinConf* **);**
C#	**void SetMICRMinConf (** **int** *nMinConf* **);**
C	**HRESULT IampImage_SetMICRMinConf(** **int** *nMinConf* **);**
JAVA	**void SetMICRMinConf (** **int** *nMinConf* **);**
VB	**SetMICRMinConf (** *nMinConf* **as Integer** **)**

Parameters

nMinConf
Sets the minimum confidence value for character acceptance.

Remarks

An input parameter describing the Minimum Confidence value that should be used to accept or reject a character. The value range is between 0 and 99 but the only reasonable values are between 80 and 90. Setting the value too high will reject characters that are read correctly. Setting the value too low will cause the acceptance of characters which are misreads or substitution errors.

The user must decide the best parameter value based on testing with their data set and with their set of needs. In general, good images do not cause substitution errors. It is corrupted images that cause problems. In all cases, a substitution error rate over a large data set is still expected to be a fraction of one percent.

The following information is based on testing a wide range of images with the toolkit.

A MinCon of 80 is recommended for doing verification. It will generate some substitution errors on corrupted images but since it is being compared to another result, this effect is minimized.

A MinCon of 85 is recommended for general usage. This will reduce the substitution rate and only nominally reduce the read rate.

A MinCon of 89 is recommend for the lowest substitution error rate without dramatically reducing the read rate.

4.20. IampMICR::SetMICRRules

The **IampMICR::SetMICRCode** method specifies the national banking association rules to apply.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT SetMICRRules (** **int** *nMICRRules* **);**
C#	**void SetMICRRules (** **int** *nMICRRules* **);**
C	**HRESULT IampImage_SetMICRRules(** **int** *nMICRRules* **);**
JAVA	**void SetMICRRules (** **int** *nMICRRules* **);**
VB	**SetMICRRules (** *nMICRRules* **as Integer** **)**

Parameters

nMICRRules
0 = ABA rules set

Remarks

This specifies the national banking association rules to apply. Currently, only the ABA rule set from the US is available.

See Also

IampMICR::SetMICRCode

4.21. IampMICR::SetNoBlanks

The **IampMICR::SetNoBlanks** method sets whether or not MICR blanks are reported.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT SetNoBlanks (** **int** *nNoBlanks* **);**

C#	**void SetNoBlanks (** **int** *nNoBlanks* **);**
C	**HRESULT IampImage_SetNoBlanks(** **int** *nNoBlanks* **);**
JAVA	**void SetNoBlanks (** **int** *nNoBlanks* **);**
VB	**SetNoBlanks (** *nNoBlanks* **as Integer** **)**

Parameters

nNoBlanks
1 removes MICR blanks from output, 0 leaves them in.

4.22. IampMICR::SetNoRules

The **IampMICR::SetNoRules** method allows internal banking rules to be ignored.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT SetNoRules (** **int** *nNoRules* **);**
C#	**void SetNoRules (** **int** *nNoRules* **);**
C	**HRESULT IampImage_SetNoRules(** **int** *nNoRules* **);**
JAVA	**void SetNoRules (** **int** *nNoRules* **);**
VB	**SetNoRules (** *nNoRules* **as Integer** **)**

Parameters

nNoRules
1 deactivates internal banking rules, 0 activates them.

Remarks

This is normally used when the input image contains only a portion of a MICR line.

See Also

IampMICR::SetNoBlanks

4.23. IampMICR::SetOCRAEnable

The **IampMICR::SetOCRAEnable** method activates OCR-A character reading.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT SetOCRAEnable (** **int** *nEnable*, **int** *nNumeric* **);**
C#	**void SetOCRAEnable (** **int** *nEnable*, **int** *nNumeric* **);**
C	**HRESULT IampImage_SetOCRAEnable(** **int** *nEnable*, **int** *nNumeric* **);**
JAVA	**void SetOCRAEnable (** **int** *nEnable*, **int** *nNumeric* **);**
VB	**SetOCRAEnable (** *nEnable* **as Integer,** *nNumeric* **as Integer** **)**

Parameters

nEnable
1 enables OCR-A character reading, and 0 disables it. Enabling this will disable OCR-B and MICR reading.

NNumeric
1 enables OCR-A character reading optimized for numeric characters only. 0 enables full character set reading.

Remarks

OCR-A reading must be disabled to read OCR-B or MICR data.

See Also

IampMICR::SetOCRBEnable

4.24. IampMICR::SetOCRBEnable

The **IampMICR::SetOCRBEnable** method activates OCR-B character reading.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT SetOCRBEnable (** **int** *nEnable*, **int** *nNumeric* **);**
C#	**void SetOCRBEnable (** **int** *nEnable*, **int** *nNumeric* **);**
C	**HRESULT IampImage_SetOCRBEnable(** **int** *nEnable*, **int** *nNumeric* **);**
JAVA	**void SetOCRBEnable (** **int** *nEnable*, **int** *nNumeric* **);**
VB	**SetOCRBEnable (** *nEnable* **as Integer,** *nNumeric* **Integer** **)**

Parameters

nEnable
1 enables OCR-B character reading, and 0 disables it. Enabling this will disable OCR-A and MICR reading.

NNumeric
1 enables OCR-B character reading optimized for numeric characters only. 0 enables full character set reading.

Remarks

OCR-B reading must be disabled to read OCR-A or MICR data.

See Also

IampMICR::SetOCRAEnable

4.25. IampMICR::SetResolution

The **IampMICR::SetResolution** method sets the way that a check image's resolution is determined.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT SetResolution (** **int** *nForceResolution*, **int** *nResolution* **);**
C#	**void SetResolution (** **int** *nForceResolution*, **int** *nResolution* **);**
C	**HRESULT IampImage_SetResolution(** **int** *nForceResolution*, **int** *nResolution* **);**
JAVA	**void SetResolution (** **int** *nForceResolution*, **int** *nResolution* **);**
VB	**SetResolution (** *nForceResolution* **as Integer,** *nResolution* **as Integer** **)**

Parameters

nForceResolution
If nResolution is greater than 0, this value (in DPI) will be the check image's resolution.

nResolution
less than 0: use resolution data in the check image file
0: estimate resolution based on image resolution and common check sizes
greater than 0: use nForceResolution for resolution

Remarks

A good approximation to the actual resolution of the check image is needed to determine where the bottom approximately 5/8" of the check is located. Forcing resolution with this parameter is normally not necessary since the image file will generally contain this information.

If resolution data is not included in the file, a non-zero value for the Resolution parameter will be used.

The input image will have the resolution set if the input value is >= 0.

4.26. IampMICR::SetRotate

The **IampMICR::SetRotate** method sets check image rotation to 90 degrees right, 90 degrees left, or off.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT SetRotate (** **int** *nRotate* **);**
C#	**void SetRotate (** **int** *nRotate* **);**

C	**HRESULT lampImage_SetRotate(** **int** *nRotate* **);**
JAVA	**void SetRotate (** **int** *nRotate* **);**
VB	**SetRotate (** *nRotate* **as Integer** **)**

Parameters

nRotate
0 = no rotation
1 = rotate 90 degrees right
-1 = rotate 90 degrees left

Remarks

This setting is used to accommodate check images that are scanned 90 degrees off of normal screen orientation.

4.27. IampMICR::SetTimeout

The **IampMICR::SetTimeout** method sets the maximum allowed processing time for a MICR read attempt, in seconds.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT SetTimeout (** **double** *dblTimeout* **);**
C#	**void SetTimeout (** **double** *dblTimeout* **);**
C	**HRESULT lampImage_SetTimeout(** **double** *dblTimeout* **);**
JAVA	**void SetTimeout (** **double** *dblTimeout* **);**
VB	**SetTimeout (** *dblTimeout* **as Double** **)**

Parameters

dblTimeout

Sets the maximum allowed number of seconds for a MICR read attempt.

Remarks

2.00 seconds is the recommended default setting.

4.28. IampMICR::VerifyMICR

The **IampImage::VerifyMICR** method analyzes a check image and verifies a previous MICR read result on a character by character basis.

Quick Info

See IampMICR: IDispatch.

C++	**HRESULT VerifyMICR (** **int** *nMode*, **int** *nCount*, **BSTR** *pInputMICR*, **BSTR*** *pReadMICR*, **int*** *nReadCount*, **BSTR*** *pVerifyMICR*, **int*** *nVerifyCount*, **int*** *ampResult* **);**
C#	**int VerifyMICR (** **int** *nMode*, **int** *nCount*, **string** *pInputMICR*, **out string** *pReadMICR*, **out int** *nReadCount*, **out string** *pVerifyMICR*, **out int** *nVerifyCount* **);**
C	**HRESULT IampImage_VerifyMICR (** **int** *nMode*, **int** *nCount*, **BSTR** *pInputMICR*, **BSTR*** *pReadMICR*, **int*** *nReadCount*, **BSTR*** *pVerifyMICR*, **int*** *nVerifyCount*, **int*** *ampResult* **);**
JAVA	**int VerifyMICR (** **Integer** *nMode*, **Integer** *nCount*, **String** *pInputMICR*, **String** *pReadMICR*, **Integer** *nReadCount*, **String** *pVerifyMICR*, **Integer** *nVerifyCount* **);**

```
VB    VerifyMICR (
          nMode as Integer,
          nCount as Integer,
          pInputMICR as String,
          pReadMICR as String,
          nReadCount as Integer,
          pVerifyMICR as String,
          nVerifyCount as Integer
      )
          as Integer;
```

Parameters

nMode
An integer value that sets the mode of the Verify operation.
0: Verify MICR
1: Verify IRD

nCount
An integer containing the length of the pInputMICR string.

pInputMICR
A string containing the MICR input line to be verified.

pReadMICR
A string that will contain the result of the last MICR Read operation run by the VerifyMICR operation. This will be either the first successful read result found, or the last read result if no successful read occurs.

nReadCount
An integer that will be set to the length of the pReadMICR string.

pVerifyMICR
A string that will contain the output of the verify operation which is an integration of the pInputMICR and pReadMICR values.

nVerifyCount
An integer that will be set to the length of the pVerifyMICR string.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code. An exception will be thrown (E_OUTOFMEMORY) if the system runs out of memory while allocating variables within this method. An exception will be thrown (E_INVALIDARG) if the options, fileType or imageIndex parameters incorrect. An exception will be thrown (E_FAIL) if there is a problem while performing the verify operation.

Remarks

This API will read a MICR line from the source image and compare it with the data provided in pInputMICR which typically comes from a hardware reader or simply hardware. The vote or correctly the results of the vote are placed in pVerifyMICR. If the SetDoImageUpdate property is enabled then the source image will be modified based on the filtering needed to improve the accuracy of the read.

If nMode is set to 1 (IRD mode), VerifyMICR does a different style of voting on the EPC and Amount fields in the MICR line than if nMode is set to 0 (MICR mode). MICR mode requires an exact match in both the EPC and Amount fields in order for the voted upon results to indicate an overall match. IRD mode is not as strict and allows a match to occur even if the EPC field in the check image is completely missing. Similarly, a match may be declared if the Amount field is missing from the check image but is present in the hardware data. In addition, IRD mode does not try to exactly match dash characters, but will skip over dashes looking for correspondence on the more significant characters in a given field.

The data returned using getVerifyMICRData contains information on how the vote results were obtained for each character as opposed to a confidence percentage. The data represents the following:

0--indeterminate

1--hardware and software were the same, hardware selected

2--hardware and software best are same, hardware selected

3--hardware and software did NOT match, software selected

4--hardware and software best do NOT match, hardware selected

5--hardware misreads and software reads, software selected

6--hardware misread and no software selection, misread selected

The IRD mode voting process changes all percent 2 values to 1 values. Consequently, if there is a match, getVerifyMICRData will have all 1s.

See Also

IampMICR::getVerifyMICRData,

4.29. IampMICR::VerifyMICRField

The **IampImage::VerifyMICRField** method analyzes a check image and verifies a previous MICR read result on a field by field basis.

Quick Info

See IampMICR: IDispatch.

C++	**HRESULT VerifyMICRField (** **int** *nMode,* **int** *nCount,* **BSTR** *pInputMICR,* **BSTR*** *pReadMICR,* **int*** *nReadCount,* **BSTR*** *pVerifyMICR,* **int*** *nVerifyCount,* **int*** *ampResult* **);**
C#	**int VerifyMICRField (** **int** *nMode,* **int** *nCount,* **string** *pInputMICR,* **out string** *pReadMICR,* **out int** *nReadCount,* **out string** *pVerifyMICR,* **out int** *nVerifyCount* **);**
C	**HRESULT IampImage_VerifyMICRField (** **int** *nMode,* **int** *nCount,* **BSTR** *pInputMICR,* **BSTR*** *pReadMICR,* **int*** *nReadCount,* **BSTR*** *pVerifyMICR,* **int*** *nVerifyCount,* **int*** *ampResult* **);**
JAVA	**int VerifyMICRField (** **Integer** *nMode,* **Integer** *nCount,* **String** *pInputMICR,* **String** *pReadMICR,* **Integer** *nReadCount,* **String** *pVerifyMICR,* **Integer** *nVerifyCount* **);**

VB	**VerifyMICRField (** *nMode* **as Integer**, *nCount* **as Integer**, *pInputMICR* **as String**, *pReadMICR* **as String**, *nReadCount* **as Integer**, *pVerifyMICR* **as String**, *nVerifyCount* **as Integer** **)** **as Integer;**

Parameters

nMode
An integer value that sets the mode of the Verify operation.
0: Verify MICR Field – the input image is a normal check image.
1: Verify IRD Field – the input image is a reduced size check image as found in an IRD.

nCount
An integer containing the length of the pInputMICR string.

pInputMICR
A string containing the MICR input line to be verified.

pReadMICR
A string that will contain the result of the last MICR Read operation run by the VerifyMICRField operation. This will be either the first successful read result found, or the last read result if no successful read occurs.

nReadCount
An integer that will be set to the length of the pReadMICR string.

pVerifyMICR
A string that will contain the output of the verify operation which is an integration of the pInputMICR and pReadMICR values.

nVerifyCount
An integer that will be set to the length of the pVerifyMICR string.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code. An exception will be thrown (E_OUTOFMEMORY) if the system runs out of memory while allocating variables within this method. An exception will be thrown (E_INVALIDARG) if the options, fileType or imageIndex parameters incorrect. An exception will be thrown (E_FAIL) if there is a problem while performing the verify operation.

Remarks

The VerifyMICRField function will verify that the pInputMICR field data matches the MICR codeline on the image. It ensures the usability of the MICR codeline on the image and will detect items whose captured MICR data and image data came from different source documents. Currently only black and white source images are supported.

This function accepts as input the items image and the captured MICR field elements (Routing number, account number, check serial number, and check amount). The field data is passed in to the function as a complete MICR line and then parsed to separate out the field data. Returned will be the field data as read by the OCR process. A confidence factor for each field is also returned indicating the degree of match between the input and returned field data. Minimum confidence threshold values are also input for each field. The function uses these confidence thresholds to invoke “Try harder” processing to be able to return a result with the required confidence level or higher.

The pInputMICR line is parsed into its separate “check”, “route”, “account”, and “amount” fields and these fields are used for the verify operation. All dashes and special symbols are internally removed from each field so that only the numbers remain. The account, check number and amount field strings can contain leading zeros or have them removed. Leading zeros will be ignored in the verification process except for routing numbers.

The SetFieldVerifyMinConf method can be used to set the input confidence of each field to be verified. A typical input value is 94. If the field is to be excluded during verify, then the input confidence should be set to 0.

pVerifyMICR returns the MICR field contents as read from the check image. The entire numeric field content is returned including leading zeros. No special symbols or dashes are returned with the exception of the routing number where a dash will be returned for a “4-4” format. pReadMICR will return the entire codeline for the OCR read of the last OCR engine process used. This does not combine the results of the different OCR reads and should not be relied on as the best read.

Confidence Values

The following table shows the field confidence values returned with the getFieldVerifyResultMinConf method:

Confidence Score	Description
99	All characters match at above the min confidence specified in SetFieldVerifyMinConf.
98 to 94	All characters match with best choice. (99 minus number of best choice matches)
93-86	All characters match but some unreadable by OCR. (93 minus # unreadable characters)
85-81	characters match except for some missing on OCR Result. (85 minus number of missing characters)
80-71	Field Mismatch. OCR and reference do not agree. (81minus number of characters that mismatch in reference field,)
50	Field does not exist on codeline image. Indicates missing codeline data (E.g. Missing check number or account number)
40	Account reference field matches, but additional characters returned in account number result. (Indicates presence of transit field data, or short account number mapping)

SetFieldVerifyMinConf allows users to specify their required minimum threshold per field. Valid settings for these fields are 0 or 81 to 99 as per the confidence table above. Generally the lower the confidence the faster the throughput as the OCR verification process will terminate early once the minimum confidence threshold is achieved.

The OCR process can uses several OCR engine processes to verify the field with the desired confidence level. GetFieldVerifyResultData returns the engineLevel which ranges from 1-8 indicating the number of OCR engine processes it used to achieve the result.

General information about using FieldVerify:

Often checks do not have “amounts” encoded on the code line. This function will return an amount if found regardless if you provided the reference value or not.

The function uses pattern matching to interpret the field data. For best results you should provide the check serial number. If you do not need to validate the check number, you can use SetFieldVerifyMinConf->nCheck to 0.

See Also

IampMICR::VerifyMICR, IampMICR::GetFieldVerifyResultData, IampMICR::GetFieldVerifyResultMinConf, IampMICR::getVerifyMICRData, IampMICR::SetFieldVerifyMinConf

4.30. IampMICR::AssembleMICR

The **IampMICR::AssembleMICR** method assembles a single MICR line from the fields found on a check image. This method has the reverse effect of the IampMICR::getMICRFields method.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT AssembleMICR (** **BSTR** *pszAuxOnUs*, **BSTR** *pszEPC*, **BSTR** *pszRoute*, **BSTR** *pszOnUs*, **BSTR** *pszAmount*, **BSTR** *pszTranslation*, **BSTR*** *pszOutputMICR*, **int*** *returnValue* **);**
C#	**int AssembleMICR (** **string** *pszAuxOnUs*, **string** *pszEPC*, **string** *pszRoute*, **string** *pszOnUs*, **string** *pszAmount*, **string** *pszTranslation*, **string** * *pszOutputMICR* **);**
C	**HRESULT IampImage_ AssembleMICR (** **BSTR** *pszAuxOnUs*, **BSTR** *pszEPC*, **BSTR** *pszRoute*, **BSTR** *pszOnUs*, **BSTR** *pszAmount*, **BSTR** *pszTranslation*, **BSTR*** *pszOutputMICR*, **int*** *returnValue* **);**
JAVA	**int AssembleMICR (** **String** *pszAuxOnUs*, **String** *pszEPC*, **String** *pszRoute*, **String** *pszOnUs*, **String** *pszAmount*, **String** *pszTranslation*, **String** * *pszOutputMICR* **);**
VB	**AssembleMICR (** *pszAuxOnUs* **as String**, *pszEPC* **as String**, *pszRoute* **as String**, *pszOnUs* **as String**, *pszAmount* **as String**, *pszTranslation* **as String**, *pszOutputMICR* **as String**, **)** **as String**

Parameters

pszAuxOnUs
A pointer to the string representing the input Aux On Us field.

pszEPC
A pointer to the string representing the input EPC field.

pszRoute
A pointer to the string representing the input Route field.

pszOnUs
A pointer to the string representing the input On Us field.

pszAmount
A pointer to the string representing the input Amount field.

pszTranslation
A pointer to an ASCII character translation string. If present, all MICR output will be translated through this table.

pszOutputMICR
A pointer to the output string, in to which the assembled MICR string will be written.

Remarks

The MICR output will be in the following left-to-right format:

Aux On Us (max length 17 characters)

Space (1 character if Aux On Us data present)

EPC (max length 1 character)

Routing (fixed length 11 characters)

On Us (max length 20 characters)

Space (1 character if input Amount data present)

Amount field (fixed length 12 characters if present)

Translation tables are strings containing an ordered set of characters matching the available MICR characters in the current code (i.e., E13B vs CMC7). An example translation table is shown below:

Example Translation Table

Location	Meaning	MICR	Default
0	'0'	'0'	'0'
1	'1'	'1'	'1'
2	'2'	'2'	'2'
3	'3'	'3'	'3'
4	'4'	'4'	'4'
5	'5'	'5'	'5'
6	'6'	'6'	'6'
7	'7'	'7'	'7'
8	'8'	'8'	'8'
9	'9'	'9'	'9'
10	Routing	A	'A'
11	Amount	B	'B'
12	On Us	X	'C'
13	Dash	Δ	'D'
14	Blank	' '	' '
15	Misread	NA	'*'
16	Best is same as Selected	NA	'N'

See Also

IampMICR::getMICRFields

4.31. IampMICR::DetectCoupon

The **IampMICR::DetectCoupon** method provides for rapid detection and reading of Remittance Coupons, especially as found in Retail Lockbox processing. When processing a large number of intermixed coupons and checks, rapid detection of a coupon can greatly improve the throughput of the process. A coupon with multiple barcodes can be detected in less time than is needed to process the associated check.

Quick Info

See IampMICR : IDispatch.

C++

```
HRESULT AssembleMICR (
    int nFrontInput,
    int nBackInput,
    int nForceResolution,
    int nFieldsCount,
    int* pnDetected,
    int* pnRotated180,
    int* pnSide,
    int nFieldType1,
    int dwSymbologyMask1,
    int bChecksum1,
    int nMinLength1,
    int nMaxLength1,
    int nFieldType2,
    int dwSymbologyMask2,
    int bCheckSum2,
    int nMinLength2,
    int nMaxLength2,
    int* pnLengthField1,
    int* pnLengthField2,
    byte* pbyFieldResults1,
    byte* pbyFieldResults2,
    int* pnDx1,
    int* pnDy1,
    int* pnX1,
    int* pnY1,
    int* pnDx2,
    int* pnDy2,
    int* pnX2,
    int* pnY2,
    int* returnValue
);
```

C#

```
int AssembleMICR (
    int nFrontInput,
    int nBackInput,
    int nForceResolution,
    int nFieldsCount,
    ref int pnDetected,
    ref int pnRotated180,
    ref int pnSide,
    int nFieldType1,
    int dwSymbologyMask1,
    int bChecksum1,
    int nMinLength1,
    int nMaxLength1,
    int nFieldType2,
    int dwSymbologyMask2,
    int bCheckSum2,
    int nMinLength2,
    int nMaxLength2,
    ref int pnLengthField1,
    ref int pnLengthField2,
    byte[] pbyFieldResults1,
    byte[] pbyFieldResults2,
    ref int pnDx1,
    ref int pnDy1,
    ref int pnX1,
    ref int pnY1,
    ref int pnDx2,
    ref int pnDy2,
    ref int pnX2,
    ref int pnY2
);
```

C

```
HRESULT IampImage_ AssembleMICR (
    int nFrontInput,
    int nBackInput,
    int nForceResolution,
    int nFieldsCount,
    int* pnDetected,
    int* pnRotated180,
    int* pnSide,
    int nFieldType1,
    int dwSymbologyMask1,
    int bChecksum1,
    int nMinLength1,
    int nMaxLength1,
    int nFieldType2,
    int dwSymbologyMask2,
    int bCheckSum2,
    int nMinLength2,
    int nMaxLength2,
    int* pnLengthField1,
    int* pnLengthField2,
    byte* pbyFieldResults1,
    byte* pbyFieldResults2,
    int* pnDx1,
    int* pnDy1,
    int* pnX1,
    int* pnY1,
    int* pnDx2,
    int* pnDy2,
    int* pnX2,
    int* pnY2,
    int* returnValue
);
```

JAVA	**int AssembleMICR (** **int** *nFrontInput*, **int** *nBackInput*, **int** *nForceResolution*, **int** *nFieldsCount*, **int** *pnDetected*, **int** *pnRotated180*, **int** *pnSide*, **int** *nFieldType1*, **int** *dwSymbologyMask1*, **int** *bChecksum1*, **int** *nMinLength1*, **int** *nMaxLength1*, **int** *nFieldType2*, **int** *dwSymbologyMask2*, **int** *bCheckSum2*, **int** *nMinLength2*, **int** *nMaxLength2*, **int** *pnLengthField1*, **int** *pnLengthField2*, **byte** *pbyFieldResults1*, **byte** *pbyFieldResults2*, **int** *pnDx1*, **int** *pnDy1*, **int** *pnX1*, **int** *pnY1*, **int** *pnDx2*, **int** *pnDy2*, **int** *pnX2*, **int** *pnY2* **);**
VB	**DetectCoupon (** *nFrontInput* **as Integer,** *nBackInput* **as Integer,** *nForceResolution* **as Integer,** *nFieldsCount* **as Integer,** *pnDetected* **as Integer,** *pnRotated180* **as Integer,** *pnSide* **as Integer,** *nFieldType1* **as Integer,** *dwSymbologyMask1* **as Integer,** *bChecksum1* **as Integer,** *nMinLength1* **as Integer,** *nMaxLength1* **as Integer,** *nFieldType2* **as Integer,** *dwSymbologyMask2* **as Integer,** *bCheckSum2* **as Integer,** *nMinLength2* **as Integer,** *nMaxLength2* **as Integer,** *pnLengthField1* **as Integer,** *pnLengthField2* **as Integer,** *pbyFieldResults1()* **as Byte,** *pbyFieldResults2()* **as Byte,** *pnDx1* **as Integer,** *pnDy1* **as Integer,** *pnX1* **as Integer,** *pnY1* **as Integer,** *pnDx2* **as Integer,** *pnDy2* **as Integer,** *pnX2* **as Integer,** *pnY2* **as Integer** **)** **as Integer**

Parameters

nFrontInput
The index of the image object containing the front side of the document.

nBackInput
The index of the image object containing the back side of the document.

nForceResolution
This forces the resolution of input images to the specified DPI value, or uses the image default if set to -1.

nFieldsCount
The number of fields (barcode or OCR results) used to detect a coupon. This value can be 1 or 2.

pnDetected
DetectCoupon will set this to nonzero if *nFieldsCount* fields were successfully detected, or zero if not.

pnRotated180
DetectCoupon will set this to nonzero if the image was upside-down, or zero if it was right side up.

pnSide
DetectCoupon will set this to nonzero if a field was found on the second image, or zero if not.

nFieldType1
The first type of field to search for. 1 = barcode, 2 = OCR.

dwSymbologyMask1
The mask of the barcode types or OCR types to search for when searching for the first field. These can be made from the bitwise-or of standard AmpLib barcode or OCR mask values.

bChecksum1
If nonzero, DetectCoupon will look for checksums on the first field.

nMinLength1
The minimum character count allowed for success in the first field read operation.

nMaxLength1
The maximum character count allowed for success in the first field read operation.

nFieldType2
The second type of field to search for (if nFieldsCount = 2). 1 = barcode, 2 = OCR.

dwSymbologyMask2
The mask of the barcode types or OCR types to search for when searching for the second field. These can be made from the bitwise-or of standard AmpLib barcode or OCR mask values.

bCheckSum2
If nonzero, DetectCoupon will look for checksums on the second field.

nMinLength2
The minimum character count allowed for success in the second field read operation.

nMaxLength2
The maximum character count allowed for success in the second field read operation.

pnLengthField1
DetectCoupon will set this to the character count of the first field result, if found.

pnLengthField2
DetectCoupon will set this to the character count of the second field result, if found.

pbyFieldResults1
DetectCoupon will write the first field result into this array, if found.

pbyFieldResults2
DetectCoupon will write the second field result into this array, if found.

pnX1, pnY1, pnDx1, pnDy1
DetectCoupon will write the first field x location, y location, width, and height, respectively, into these values, if found.

pnX2, pnY2, pnDx2, pnDy2
DetectCoupon will write the first field x location, y location, width, and height, respectively, into these values, if found.

Remarks

The DetectCoupon function will examine the Front image and alternatively the Back image for the conditions that define a Coupon. It will then report what, in anything, was found. In many cases, the detection function is sufficient to provide all the needed information about the coupon and no additional read is necessary.

The DetectCoupon function will examine one or two fields for the conditions that define a Coupon. Those conditions are the presence of the specific symbology (barcode or OCR), the specific length of the result and additionally in future releases the specific location and content.

The resolution of the image is usually contained in the image structure but if this is wrong for any reason, the nForceResolution value can be used to input the current information.

A typical condition for defining a coupon is the presence of the USPS One Code. Most (but not all) addresses on retail coupons will have a USPS One Code. By the same token, most checks will not have such a code. Hence a coupon detector can have the following settings:

```
nFieldType1 =  1 ;   // barcode
SymbologyMask1 =  BC_4STATEUSPS ;   // usps one code only
MinLength1 = 40 ;
MaxLength1 = 80 ;
```

The ampDetectCoupon can be used to detect and read coupons when the fields are general purpose barcodes. For example, assume there are two barcodes, one has the amount due and is 6 digits and the other is the account and it is 8 digits. The following settings will detect a two barcode image as a coupon:

```
nFieldType1 =  1 ;   // barcode
SymbologyMask1 =  BC_3of9;    // Code 39
MinLength1 = 6 ;
MaxLength1 = 6 ;

nFieldType1 = 1 ;    // barcode
SymbologyMask1 =  BC_3of9;    // Code 39
MinLength1 = 8 ;
MaxLength1 = 8 ;
```

In this case the result fields for length, value and location (if output parameters present) will be set. A single call will provide the detection and the values needed for processing this coupon.

Another common detect and read case is for OCR characters in the bottom or top clear band on the coupon. The amount of data in a field tend to be very long in this case. Again, the length of the result will determine if it is the Coupon. An example of this is:

```
nFieldType1 = 2 ;    // OCR
SymbologyMask1 = ampOCRANUM    ;  //OCR A numeric only
MinLength1 = 50 ;
MaxLength1 = 80 ;
```

See Also

IampMICR::ReadMICR

4.32. IampMICR::enableReadRemit

The **IampMICR::enableReadRemit** property enables MICR reading over the entire workimage property.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT put_enableReadRemit (** **BOOL** *newVal* **);** **HRESULT get_enableReadRemit (** **BOOL *** *pVal* **);**
C#	**int enableReadRemit (** **boolean** *newVal* **};** **boolean enableReadRemit**
C	**HRESULT IampMICR_put_enableReadRemit (** **BOOL** *newVal* **);** **HRESULT IampMICR_get_enableReadRemit (** **BOOL *** *pVal* **);**
JAVA	**void set_enableReadRemit (** **boolean** *newVal* **);** **void get_enableReadRemit (** **boolean*** *pVal* **);**
VB	**enableReadRemit (** *NewVal* **As Boolean** **) as Integer;** **enableReadRemit () as Boolean;**

Parameters

newVal, pVal
A flag that turns on or off enableReadRemit.

Remarks

This property causes the ReadMICR method to read the entire workimage property looking for MICR data. Normally, just the bottom .5 inch of the image is examined. The cropDistance property is still used when enableReadRemit is true.

See IampMICR::ReadMICR

4.33. IampMICR::OCRCode

The **IampMICR::OCRCode** property controls the OCR algorithm used in the ReadMICR operation.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT put_OCRCode (** **int** *newVal* **);** **HRESULT get_OCRCode (** **int *** *pVal* **);**

C#	**int OCRCode (** **int** *newVal* **};** **int OCRCode**
C	**HRESULT IampMICR_put_OCRCode (** **int** *newVal* **);** **HRESULT IampMICR_get_OCRCode (** **int *** *pVal* **);**
JAVA	**void set_OCRCode (** **int** *newVal* **);** **void get_OCRCode (** **int*** *pVal* **);**
VB	**OCRCode (** *NewVal* **As Integer** **) as Integer;** **OCRCode () as Integer;**

Parameters

newVal, pVal
An integer that gets or sets OCRCode.

Remarks

The value of the OCRCode property controls the algorithm used in ReadMICR according to the following table:

OCRCode	Recognition Algorithm
0	Standard two engine MICR processing
1	E13B
1	MICR
2	CMC7
3	CMC7 Line Seek
4	OCRA Numeric
8	OCRA Numeric Special
12	OCRA Alphanumeric
16	OCRA Alphanumeric Special
20	OCRA Euro
32	OCRB Numeric
64	OCRB Numeric Special
96	OCRB Alphanumeric
128	OCRB Alphanumeric Special
160	OCRB Euro
256	Omni Numeric
272	Omni Numeric Multi Line Seek
512	MICR Line Seek
528	MICR Multi Line Seek
1024	Courtesy Amount Recognition Numeric
2048	Four State Barcode
2056	Four State Barcode - Denmark
2064	Four State Barcode Multi
4096	Page Remit
8192	Square

See IampMICR::ReadMICR

4.34. IampMICR::FormatMICRFields

The **IampMICR::FormatMICRFields** method uses AMPLib to generate formatted MICR substrings from a source MICR string. This method has the ability to translate unusual characters with a translation table.

Quick Info

See IampMICR : IDispatch.

C++

```
HRESULT FormatMICRFields (
    BSTR pszSource,
    int nFilter,
    BSTR pszTranslate,
    BSTR *pszOutput,
    BSTR *pszAuxOnUs,
    BSTR *pszEPC,
    BSTR *pszRoute,
    BSTR *pszOnUs,
    BSTR *pszAmount,
    int *returnValue
);
```

C#

```
int FormatMICRFields (
    string pszSource,
    int nFilter,
    string pszTranslate,
    string *pszOutput,
    string *pszAuxOnUs,
    string *pszEPC,
    string *pszRoute,
    string *pszOnUs,
    string *pszAmount
);
```

C

```
HRESULT IampImage_ FormatMICRFields (
    BSTR pszSource,
    int nFilter,
    BSTR pszTranslate,
    BSTR *pszOutput,
    BSTR *pszAuxOnUs,
    BSTR *pszEPC,
    BSTR *pszRoute,
    BSTR *pszOnUs,
    BSTR *pszAmount,
    int *returnValue
);
```

JAVA

```
int FormatMICRFields (
    String pszSource,
    int nFilter,
    String pszTranslate,
    String *pszOutput,
    String *pszAuxOnUs,
    String *pszEPC,
    String *pszRoute,
    String *pszOnUs,
    String *pszAmount
);
```

VB	**FormatMICRFields (** *pszSource* **as String,** *nFilter* **as Integer,** *pszTranslate* **as String,** *pszOutput* **as String,** *pszAuxOnUs* **as String,** *pszEPC* **as String,** *pszRoute* **as String,** *pszOnUs* **as String,** *pszAmount* **as String** **) as Integer**

Parameters

pszSource
A pointer to the source string containing the MICR line to be formatted.

nFilter
If non-zero, any characters preceding the Aux On Us field will be removed.

pszTranslation
A pointer to an ASCII character translation string. If present, all MICR output will be translated through this table.

pszOutput
A pointer to the output string, to receive the complete formatted string.

pszAuxOnUs
A pointer to the string representing the output Aux On Us field.

pszEPC
A pointer to the string representing the output EPC field.

pszRoute
A pointer to the string representing the output Route field.

pszOnUs
A pointer to the string representing the output On Us field.

pszAmount
A pointer to the string representing the output Amount field.

Remarks

If non-null, szOutput points to the formatted 62 byte MICR data. The traditional field byte boundary assignments after formatting are:

0-16 AuxOnUs field - length: 17 bytes
18 EPC Code - length: 1 byte
19-29 Route field - length 11 bytes
30-49 OnUs field - length 20 bytes
50-61 Amount field - length 12 bytes

Field data that is not present in the original input will be filled with space characters.

Translation tables are strings containing an ordered set of characters matching the available MICR characters in the current code (i.e., E13B vs CMC7). An example translation table is shown below:

Example Translation Table

Location	Meaning	MICR	Default
0	'0'	'0'	'0'
1	'1'	'1'	'1'
2	'2'	'2'	'2'

3	'3'	'3'	'3'
4	'4'	'4'	'4'
5	'5'	'5'	'5'
6	'6'	'6'	'6'
7	'7'	'7'	'7'
8	'8'	'8'	'8'
9	'9'	'9'	'9'
10	Routing	A	'A'
11	Amount	B	'B'
12	On Us	X	'C'
13	Dash	Δ	'D'
14	Blank	' '	' '
15	Misread	NA	'*'
16	Best is same as Selected	NA	'N'

See Also

IampMICR::getMICRFields

4.35. IampMICR::GetMoreReadCameraResults

The **IampMICR::GetMoreReadCameraResults** method returns additional status results following the ReadCamera operation.

Quick Info

See IampMICR : IDispatch.

C++

```
HRESULT GetMoreReadCameraResults (
    int * pnOrientation,
    int * pnOriginX,
    int * pnOrginY,
    int * pnCheckWidth,
    int * pnCheckHeight,
    int * pnResX,
    int * pnResY,
    int * pnSpare4,
    int * pnSpare5,
    int * pnSpare6,
    int * pnSpare7,
    int * pnSpare8,
    int *returnValue
);
```

C#	**int GetMoreReadCameraResults (** **out int** *pnOrientation*, **out int** *pnOriginX*, **out int** *pnOrginY*, **out int** *pnCheckWidth*, **out int** *pnCheckHeight*, **out int** *pnResX*, **out int** *pnResY*, **out int** *pnSpare4*, **out int** *pnSpare5*, **out int** *pnSpare6*, **out int** *pnSpare7*, **out int** *pnSpare8* **);**
C	**HRESULT IampImage_GetMoreReadCameraResults (** **int *** *pnOrientation*, **int *** *pnOriginX*, **int *** *pnOrginY*, **int *** *pnCheckWidth*, **int *** *pnCheckHeight*, **int *** *pnResX*, **int *** *pnResY*, **int *** *pnSpare4*, **int *** *pnSpare5*, **int *** *pnSpare6*, **int *** *pnSpare7*, **int *** *pnSpare8*, **int *** *returnValue* **);**
JAVA	**int GetMoreReadCameraResults (** **Integer** *pnOrientation*, **Integer** *pnOriginX*, **Integer** *pnOrginY*, **Integer** *pnCheckWidth*, **Integer** *pnCheckHeight*, **Integer** *pnResX*, **Integer** *pnResY*, **Integer** *pnSpare4*, **Integer** *pnSpare5*, **Integer** *pnSpare6*, **Integer** *pnSpare7*, **Integer** *pnSpare8* **);**
VB	**GetMoreReadCameraResults (** *pnOrientation* **as Integer,** *pnOriginX* **as Integer,** *pnOrginY* **as Integer,** *pnCheckWidth* **as Integer,** *pnCheckHeight* **as Integer,** *pnResX* **as Integer,** *pnResY* **as Integer,** *pnSpare4* **as Integer** *pnSpare5* **as Integer,** *pnSpare6* **as Integer,** *pnSpare7* **as Integer,** *pnSpare8* **as Integer** **) as Integer**

Parameters

pnOrientation

Indicates the source image was rotated a number of degrees: 0 – no rotation, 90 – the source image was rotated clockwise, 180 – the image was upside down, 270 – the source image was rotated counter clockwise. The R parameter must be used when calling ReadCamera in order to handle rotated source images.

pnOriginX
The X Coordinate of the check front image.

pnOriginY
The Y coordinate of the check front image.

pnCheckWidth
The pixel width of the check image.

pnCheckHeight
The pixel height of the check image.

pnResX
The horizontal resolution of the check image.

pnResY
The vertical resolution of the check image.

pnSpare4 – pnSpare8
These return parameters are reserved for future use.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

This method returns additional values set by the last use of ReadCamera on the input and output buffer images.

See Also

See IampMICR::ReadCamera, IampMICR::GetReadCameraResults

4.36. IampMICR::GetReadCameraResults

The **IampMICR::GetReadCameraResults** method returns status results following the ReadCamera operation.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT GetReadCameraResults (** **BSTR** **pszAuxOnUs*, **BSTR** **pszRoute*, **BSTR** **pszOnUs*, **int *** *pnMicrOK,* **int *** *pnMicrConf,* **int *** *pnImagesOK,* **int *** *pnImagesTooDark,* **int *** *pnImagesTooLight,* **int *** *pnImgNotSize,* **int *** *pnRearMissing,* **int *** *pnFrontMissing,* **int *** *returnValue* **);**

C#	**int GetReadCameraResults (** **string ***pszAuxOnUs*, **string** *pszRoute*, **string ***pszOnUs*, **out int** *pnMicrOK,* **out int** *pnMicrConf,* **out int** *pnImagesOK,* **out int** *pnImagesTooDark,* **out int** *pnImagesTooLight,* **out int** *pnImgNotSize,* **out int** *pnRearMissing,* **out int** *pnFrontMissing* **);**
C	**HRESULT IampImage_ GetReadCameraResults (** **BSTR** *pszAuxOnUs*, **BSTR** *pszRoute*, **BSTR** *pszOnUs*, **int *** *pnMicrOK,* **int *** *pnMicrConf,* **int *** *pnImagesOK,* **int *** *pnImagesTooDark,* **int *** *pnImagesTooLight,* **int *** *pnImgNotSize,* **int *** *pnRearMissing,* **int *** *pnFrontMissing,* **int ***returnValue* **);**
JAVA	**int GetReadCameraResults (** **String** *pszAuxOnUs*, **String** *pszRoute*, **String** *pszOnUs*, **Integer** *pnMicrOK,* **Integer** *pnMicrConf,* **Integer** *pnImagesOK,* **Integer** *pnImagesTooDark,* **Integer** *pnImagesTooLight,* **Integer** *pnImgNotSize,* **Integer** *pnRearMissing,* **Integer** *pnFrontMissing,* **);**
VB	**GetReadCameraResults (** *pszAuxOnUs* **as String,** *pszRoute* **as String,** *pszOnUs* **as String,** *pnMicrOK* **as Integer,** *pnMicrConf* **as Integer,** *pnImagesOK* **as Integer,** *pnImagesTooDark* **as Integer** *pnImagesTooLight* **as Integer,** *pnImgNotSize* **as Integer,** *pnRearMissing* **as Integer,** *pnFrontMissing* **as Integer** **) as Integer**

Parameters

pszAuxOnUs
This string will contain the AuxOnUs field of the recognized MICR line.

pszRoute
This string will contain the Route field of the recognized MICR line.

pszOnUs
This string will contain the OnUs field of the recognized MICR line.

pnMicrOK
A nonzero value indicates that the MICR line was read successfully.

pnMicrConf
0-100 level indicating confidence of reading MICR line. Over 50 indicates a successful read of the MICR line routing number.

pnImagesOK
A nonzero value indicates success in generating compliant TIFF images.

pnImagesTooDark
Indicates one or both of the images are too dark.

pnImagesTooLight
Indicates one or both of the images are too light.

pnImgNotSize
Indicates one or both of the images are the wrong size.

pnRearMissing
No output image is available. Indicates a rear image was not received or failed to process.

pnFrontMissing
No output image available. Indicates a front image failed to process.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

This method returns values set by the last use of ReadCamera on the input and output buffer images.

See Also

See IampMICR::ReadCamera

4.37. IampMICR::GetReadScannerResults

The **IampMICR::GetReadScannerResults** method returns status results following the ReadScanner operation.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT GetReadScannerResults (** **BSTR** **pszAuxOnUs*, **BSTR** **pszRoute*, **BSTR** **pszOnUs*, **int *** *pnMicrOK*, **int *** *pnMicrConf*, **int *** *pnImagesOK*, **int *** *pnImagesTooDark*, **int *** *pnImagesTooLight*, **int *** *pnImgNotSize*, **int *** *pnRearMissing*, **int *** *pnFrontMissing*, **int *** *returnValue* **);**

C#	**int GetReadScannerResults (** **string ***pszAuxOnUs*, **string** *pszRoute*, **string ***pszOnUs*, **out int** *pnMicrOK,* **out int** *pnMicrConf,* **out int** *pnImagesOK,* **out int** *pnImagesTooDark,* **out int** *pnImagesTooLight,* **out int** *pnImgNotSize,* **out int** *pnRearMissing,* **out int** *pnFrontMissing* **);**
C	**HRESULT IampImage_ GetReadScannerResults (** **BSTR** *pszAuxOnUs*, **BSTR** *pszRoute*, **BSTR** *pszOnUs*, **int *** *pnMicrOK,* **int *** *pnMicrConf,* **int *** *pnImagesOK,* **int *** *pnImagesTooDark,* **int *** *pnImagesTooLight,* **int *** *pnImgNotSize,* **int *** *pnRearMissing,* **int *** *pnFrontMissing,* **int ***returnValue* **);**
JAVA	**int GetReadScannerResults (** **String** *pszAuxOnUs*, **String** *pszRoute*, **String** *pszOnUs*, **Integer** *pnMicrOK,* **Integer** *pnMicrConf,* **Integer** *pnImagesOK,* **Integer** *pnImagesTooDark,* **Integer** *pnImagesTooLight,* **Integer** *pnImgNotSize,* **Integer** *pnRearMissing,* **Integer** *pnFrontMissing,* **);**
VB	**GetReadScannerResults (** *pszAuxOnUs* **as String,** *pszRoute* **as String,** *pszOnUs* **as String,** *pnMicrOK* **as Integer,** *pnMicrConf* **as Integer,** *pnImagesOK* **as Integer,** *pnImagesTooDark* **as Integer** *pnImagesTooLight* **as Integer,** *pnImgNotSize* **as Integer,** *pnRearMissing* **as Integer,** *pnFrontMissing* **as Integer** **) as Integer**

Parameters

pszAuxOnUs
This string will contain the AuxOnUs field of the recognized MICR line.

pszRoute
This string will contain the Route field of the recognized MICR line.

pszOnUs
This string will contain the OnUs field of the recognized MICR line.

pnMicrOK
A nonzero value indicates that the MICR line was read successfully.

pnMicrConf
0-100 level indicating confidence of reading MICR line. Over 50 indicates a successful read of the MICR line routing number.

pnImagesOK
A nonzero value indicates success in generating compliant TIFF images.

pnImagesTooDark
Indicates one or both of the images are too dark.

pnImagesTooLight
Indicates one or both of the images are too light.

pnImgNotSize
Indicates one or both of the images are the wrong size.

pnRearMissing
No output image is available. Indicates a rear image was not received or failed to process.

pnFrontMissing
No output image available. Indicates a front image failed to process.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

This method returns values set by the last use of ReadScanner on the input and output buffer images.

See Also

See IampMICR::ReadScanner

4.38. IampMICR::GetFieldVerifyResultData

The **IampMICR::GetFieldVerifyResultData** method gets OCR engine information about the most recent Verify operation.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT GetFieldVerifyResultData (** **int *** *nEngineLevel*, **int *** *nMinTotalConf* **);**
C#	**void GetFieldVerifyResultData (** **out int** *nEngineLevel*, **out int** *nMinTotalConf* **);**
C	**HRESULT IampImage_GetFieldVerifyResultData (** **int *** *nEngineLevel*, **int *** *nMinTotalConf* **);**
JAVA	**void GetFieldVerifyResultData (** **int** *nEngineLevel*, **int** *nMinTotalConf* **);**

VB	**GetFieldVerifyResultData (** *nEngineLevel* **as Integer,** *nMinTotalConf* **as Integer** **)**

Parameters

nEngineLevel
An integer value indicating the number of internal OCR processes required for the most recent Verify operation.

nMinTotalConf
An integer value indicating the lowest character confidence value found in the most recent Verify operation.

See Also

IampMICR::VerifyMICR, IampMICR::VerifyMICRField

4.39. IampMICR::GetFieldVerifyResultMinConf

The **IampMICR::GetFieldVerifyResultMinConf** method gets the lowest individual character confidence value from each of the MICR fields from the most recent Verify operation.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT GetFieldVerifyResultMinConf (** **int *** *nCheck*, **int *** *nRoute*, **int *** *nAccount*, **int *** *nAmount* **);**
C#	**void GetFieldVerifyResultMinConf (** **out int** *nCheck*, **out int** *nRoute*, **out int** *nAccount*, **out int** *nAmount* **);**
C	**HRESULT IampImage_GetFieldVerifyResultMinConf (** **int *** *nCheck*, **int *** *nRoute*, **int *** *nAccount*, **int *** *nAmount* **);**
JAVA	**void GetFieldVerifyResultMinConf (** **Integer** *nCheck*, **Integer** *nRoute*, **Integer** *nAccount*, **Integer** *nAmount* **);**
VB	**GetFieldVerifyResultMinConf (** *nCheck* **as Integer,** *nRoute* **as Integer,** *nAccount* **as Integer,** *nAmount* **as Integer** **)**

Parameters

nCheck
An integer value indicating the lowest character confidence value found in the Check/AuxOnUs field during the most recent Verify operation.

nRoute
An integer value indicating the lowest character confidence value found in the Route field during the most recent Verify operation.

nAccount
An integer value indicating the lowest character confidence value found in the Account/OnUs field during the most recent Verify operation.

nAmount
An integer value indicating the lowest character confidence value found in the Amount field during the most recent Verify operation.

See Also

IampMICR::VerifyMICR, IampMICR::VerifyMICRField

4.40. IampMICR::getMICRData

The **IampMICR::getMICRData** method returns information about a specific character in a ReadMICR, ReadCamera, or ReadScanner operation result.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT getMICRData (** **int** *charPos*, **BYTE *** *pbyResult*, **BYTE *** *pbyBestAlternate*, **int *** *nConfidence*, **int *** *nXPos* **);**
C#	**void getMICRData (** **int** *charPos*, **out Byte** *pbyResult*, **out Byte** *pbyBestAlternate*, **out int** *nConfidence*, **out int** *nXPos* **);**
C	**HRESULT IampImage_getMICRData (** **int** *charPos*, **BYTE *** *pbyResult*, **BYTE *** *pbyBestAlternate*, **int *** *nConfidence*, **int *** *nXPos* **);**
JAVA	**void getMICRData (** **Integer** *charPos*, **Byte** *pbyResult*, **Byte** *pbyBestAlternate*, **Integer** *nConfidence*, **Integer** *nXPos* **);**

VB	**getMICRData (** *charPos* **as Integer,** *pbyResult* **as Byte,** *pbyBestAlternative* **as Byte,** *nConfidence* **as Integer,** *nXPos* **as Integer** **)**

Parameters

charPos
An input integer representing the (zero-based) character position index in the most recent MICR read operation result for which information will be returned.

pbyResult
The MICR character identified at the specified position in the string.

pbyBestAlternate
The best alternate MICR character identified at the specified position in the string.

nConfidence
A 2-digit integer value ranging from 40 to 99, indicating MICR character identification confidence.

nXPos
This is an internal diagnostic variable and may be ignored.

See Also

IampMICR::ReadMICR, IampMICR::ReadCamera, IampMICR::ReadScanner

4.41. IampMICR::getMICRFields

The **IampMICR::getMICRFields** method returns fields from the last successful IampMICR::ReadMICR result, using AMPLib's ampParseMicr method.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT getMICRFields (** **BSTR** **pszAll*, **BSTR** **pszAccount*, **BSTR** **pszEPC*, **BSTR** **pszRoute*, **BSTR** **pszCheck*, **BSTR** **pszAmount*, **int** ***returnValue* **);**
C#	**int getMICRFields (** **string** ***pszAll*, **string** ***pszAccount*, **string** **pszEPC*, **string** **pszRoute*, **string** **pszCheck*, **string** **pszAmount*, **);**

C	**HRESULT IampImage_ getMICRFields (** **BSTR** *pszAll, **BSTR** *pszAccount, **BSTR** *pszEPC, **BSTR** *pszRoute, **BSTR** *pszCheck, **BSTR** *pszAmount, **int** **returnValue* **);**
JAVA	**int getMICRFields (** **String** *pszAll, **String** *pszAccount, **String** *pszEPC, **String** *pszRoute, **String** *pszCheck, **String** *pszAmount, **);**
VB	**getMICRFields (** *pszAll* **as String,** *pszAccount* **as String,** *pszEPC* **as String,** *pszRoute* **as String,** *pszCheck* **as String,** *pszAmount* **as String** **)** **as Integer**

Parameters

pszAll
A pointer to the output string to contain the complete MICR line.

pszAccount
A pointer to the string representing the output Account field.

pszEPC
A pointer to the string representing the output EPC field.

pszRoute
A pointer to the string representing the output Route field.

pszCheck
A pointer to the string representing the output Check field.

pszAmount
A pointer to the string representing the output Amount field.

Remarks

This function provides a simple interface for retrieving data from the most recent successful ReadMICR operation. For more functionality, including a character translation feature, see IampMICR::FormatMICRFields.

See Also

IampMICR::FormatMICRFields, IampMICR::AssembleMICR

4.42. IampMICR::getMICRRegion

The **IampMICR::getMICRRegion** method returns the region of interest that contains the MICR line last read with ReadMICRPage.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT getMICRRegion (** **LONG ***nTop*, **LONG ***nLeft*, **LONG ***nRight*, **LONG ***nBottom*, **int ***returnValue* **);**
C#	**int getMICRRegion (** **out int** *nTop*, **out int** *nLeft*, **out int** *nRight*, **out int** *nBottom*, **);**
C	**HRESULT IampImage_ getMICRRegion (** **LONG ***nTop*, **LONG ***nLeft*, **LONG ***nRight*, **LONG ***nBottom*, **int ***returnValue* **);**
JAVA	**int getMICRRegion (** **int** *outputTop*, **int** *outputLeft*, **int** *outputRight*, **int** *outputBottom*, **);**
VB	**getMICRRegion (** *nTop* **as Integer,** *nLeft* **as Integer,** *nRight* **as Integer,** *nBottom* **as Integer** **)** **as Integer**

Parameters

nTop
The top pixel margin of the MICR line.

nLeft
The left pixel margin of the MICR line.

nRight
The right pixel margin of the MICR line.

nBottom
The bottom pixel margin of the MICR line.

Remarks

This function provides a simple interface for retrieving the MICR region boundaries from the most recent successful ReadMICRPage operation.

See Also

IampMICR::ReadMICRPage

4.43. IampMICR::getRemitAlternateAmount

The **IampMICR::getRemitAlternateAmount** property returns a low-confidence alternative amount value from a remittance document.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT get_RemitAlternateAmount (** **int*** *nRemitAlternateAmount* **);**
C#	**int** *nRemitAlternateAmount*
C	**HRESULT IampBarcode_get_RemitAlternateAmount (** **int*** *nRemitAlternateAmount* **);**
JAVA	**void get_RemitAlternateAmount (** **int*** *nRemitAlternateAmount* **);**
VB	**RemitAlternateAmount () as Integer;**

Remarks

This property will provide an alternate low-confidence amount value from the last ReadMICRRemit operation. The value may be an amount found in the document that was less frequent than the amount returned by ReadMICRRemit. If no alternative amount value was found, the property will contain a null string.

See Also

IampMICR::ReadMICRRemit

4.44. IampMICR::getRemitAlternateCheckNumber

The **IampMICR::getRemitAlternateCheckNumber** property returns a low-confidence alternative check number value from a remittance document.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT get_RemitAlternateCheckNumber (** **int*** *nRemitAlternateCheckNumber* **);**
C#	**int** *nRemitAlternateCheckNumber*
C	**HRESULT IampBarcode_get_RemitAlternateCheckNumber (** **int*** *nRemitAlternateCheckNumber* **);**
JAVA	**void get_RemitAlternateCheckNumber (** **int*** *nRemitAlternateCheckNumber* **);**
VB	**RemitAlternateCheckNumber () as Integer;**

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

This property will provide an alternate low-confidence check number value from the last ReadMICRRemit operation. The value may be a check number found in the document that was less frequent than the check number returned by ReadMICRRemit. If no alternative check number value was found, the property will contain a null string.

See Also

IampMICR::ReadMICRRemit

4.45. IampMICR::getRemitAlternateDate

The **IampMICR::getRemitAlternateDate** property returns a low-confidence alternative date value from a remittance document.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT get_RemitAlternateDate (** **int*** *nRemitAlternateDate* **);**
C#	**int** *nRemitAlternateDate*
C	**HRESULT IampBarcode_get_RemitAlternateDate (** **int*** *nRemitAlternateDate* **);**
JAVA	**void get_RemitAlternateDate (** **int*** *nRemitAlternateDate* **);**
VB	**RemitAlternateDate () as Integer;**

Remarks

This parameter will provide an alternate low-confidence date value from the last ReadMICRRemit operation. The value may be a date found in the document that was less frequent than the date returned by ReadMICRRemit. If no alternative date value was found, the property will contain a null string.

See Also

IampMICR::ReadMICRRemit

4.46. IampMICR::getRemitAmountConfidence

The **IampMICR::getRemitAmountConfidence** property returns the confidence value of the check amount last read with ReadMICRRemit.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT get_RemitAmountConfidence (** **int*** *nRemitAmountConfidence* **);**
C#	**int** *nRemitAmountConfidence*
C	**HRESULT IampBarcode_get_RemitAmountConfidence (** **int*** *nRemitAmountConfidence* **);**

JAVA	**void get_RemitAmountConfidence (** **int*** *nRemitAmountConfidence* **);**
VB	**RemitAmountConfidence () as Integer;**

Remarks

This property provides a confidence value for the check amount found in the most recent successful ReadMICRRemit operation. A confidence value in the 90s indicates multiple occurrences of an amount in the document. Confidence lower than 90 indicates a lack of corroborative findings for the result.

See Also

IampMICR::ReadMICRRemit

4.47. IampMICR::getRemitAmountRegion

The **IampMICR::getRemitAmountRegion** method returns the region of interest that contains the amount last read with ReadMICRRemit.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT getRemitAmountRegion (** **LONG ***nTop*, **LONG ***nLeft*, **LONG ***nRight*, **LONG ***nBottom*, **int ***returnValue* **);**
C#	**int getRemitAmountRegion (** **out int** *nTop*, **out int** *nLeft*, **out int** *nRight*, **out int** *nBottom*, **);**
C	**HRESULT IampImage_ getRemitAmountRegion (** **LONG ***nTop*, **LONG ***nLeft*, **LONG ***nRight*, **LONG ***nBottom*, **int ***returnValue* **);**
JAVA	**int getRemitAmountRegion (** **int** *outputTop*, **int** *outputLeft*, **int** *outputRight*, **int** *outputBottom*, **);**
VB	**getRemitAmountRegion (** *nTop* **as Integer,** *nLeft* **as Integer,** *nRight* **as Integer,** *nBottom* **as Integer** **)** **as Integer**

Parameters

nTop
The top pixel margin of the amount line.

nLeft
The left pixel margin of the amount line.

nRight
The right pixel margin of the amount line.

nBottom
The bottom pixel margin of the amount line.

Remarks

This function provides a simple interface for retrieving the amount region boundaries from the most recent successful ReadMICRRemit operation.

See Also

IampMICR::ReadMICRRemit

4.48. IampMICR::getRemitDateConfidence

The **IampMICR::getRemitDateConfidence** property returns the confidence value of the date last read with ReadMICRRemit.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT get_RemitDateConfidence (** **int*** *nRemitDateConfidence* **);**
C#	**int** *nRemitDateConfidence*
C	**HRESULT IampBarcode_get_RemitDateConfidence (** **int*** *nRemitDateConfidence* **);**
JAVA	**void get_RemitDateConfidence (** **int*** *nRemitDateConfidence* **);**
VB	**RemitDateConfidence () as Integer;**

Remarks

This property provides a confidence value for the check date found in the most recent successful ReadMICRRemit operation. A confidence value in the 90s indicates multiple occurrences of a date in the document. Confidence lower than 90 indicates a lack of corroborative findings for the result.

See Also

IampMICR::ReadMICRRemit

4.49. IampMICR::getRemitDateRegion

The **IampMICR::getRemitDateRegion** method returns the region of interest that contains the date last read with ReadMICRRemit.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT getRemitDateRegion (** **LONG ***nTop*, **LONG ***nLeft*, **LONG ***nRight*, **LONG ***nBottom*, **int ***returnValue* **);**
C#	**int getRemitDateRegion (** **out int** *nTop*, **out int** *nLeft*, **out int** *nRight*, **out int** *nBottom*, **);**
C	**HRESULT IampImage_ getRemitDateRegion (** **LONG ***nTop*, **LONG ***nLeft*, **LONG ***nRight*, **LONG ***nBottom*, **int ***returnValue* **);**
JAVA	**int getRemitDateRegion (** **int** *outputTop*, **int** *outputLeft*, **int** *outputRight*, **int** *outputBottom*, **);**
VB	**getRemitDateRegion (** *nTop* **as Integer,** *nLeft* **as Integer,** *nRight* **as Integer,** *nBottom* **as Integer** **)** **as Integer**

Parameters

nTop
The top pixel margin of the date line.

nLeft
The left pixel margin of the date line.

nRight
The right pixel margin of the date line.

nBottom
The bottom pixel margin of the date line.

Remarks

This function provides a simple interface for retrieving the date region boundaries from the most recent successful ReadMICRRemit operation.

See Also

IampMICR::ReadMICRRemit

4.50. IampMICR::getVerifyMICRData

The **IampMICR::getVerifyMICRData** method returns information about a specific character in a Verify operation result.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT getVerifyMICRData (** **int** *charPos*, **BYTE *** *pbyResult*, **BYTE *** *pbyBestAlternate*, **int *** *nConfidence*, **int *** *nXPos* **);**
C#	**void getVerifyMICRData (** **int** *charPos*, **out Byte** *pbyResult*, **out Byte** *pbyBestAlternate*, **out int** *nConfidence*, **out int** *nXPos* **);**
C	**HRESULT IampImage_getVerifyMICRData (** **int** *charPos*, **BYTE *** *pbyResult*, **BYTE *** *pbyBestAlternate*, **int *** *nConfidence*, **int *** *nXPos* **);**
JAVA	**void getVerifyMICRData (** **Integer** *charPos*, **Byte** *pbyResult*, **Byte** *pbyBestAlternate*, **Integer** *nConfidence*, **Integer** *nXPos* **);**
VB	**getVerifyMICRData (** *charPos* **as Integer,** *pbyResult* **as Byte,** *pbyBestAlternative* **as Byte,** *nConfidence* **as Integer,** *nXPos* **as Integer** **)**

Parameters

charPos
An input integer representing the (zero-based) character position index in the most recent Verify operation result for which information will be returned.

pbyResult
The MICR character identified at the specified position in the string.

pbyBestAlternate
The best alternate MICR character identified at the specified position in the string.

nConfidence
A 2-digit integer value ranging from 40 to 99, indicating MICR character identification confidence.

nXPos
This is an internal diagnostic variable and may be ignored.

See Also

IampMICR::VerifyMICR, IampMICR::VerifyMICRField

4.51. IampMICR::getRemitCheckNumberConfidence

The **IampMICR::getRemitCheckNumberConfidence** property returns the confidence value of the check number last read with ReadMICRRemit.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT get_RemitCheckNumberConfidence (** **int*** *nRemitCheckNumberConfidence* **);**
C#	**int** *nRemitDateConfidence*
C	**HRESULT IampBarcode_get_RemitCheckNumberConfidence (** **int*** *nRemitCheckNumberConfidence* **);**
JAVA	**void get_RemitCheckNumberConfidence (** **int*** *nRemitCheckNumberConfidence* **);**
VB	**RemitCheckNumberConfidence () as Integer;**

Remarks

This property provides a confidence value for the check number found in the most recent successful ReadMICRRemit operation. A confidence value in the 90s indicates multiple occurrences of a check number in the document. Confidence lower than 90 indicates a lack of corroborative findings for the result.

See Also

IampMICR::ReadMICRRemit

4.52. IampMICR::getRemitCheckNumberRegion

The **IampMICR::getRemitCheckNumberRegion** method returns the region of interest that contains the check number last read with ReadMICRRemit.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT getRemitCheckNumberRegion (** **LONG ***nTop*, **LONG ***nLeft*, **LONG ***nRight*, **LONG ***nBottom*, **int ***returnValue* **);**
C#	**int getRemitCheckNumberRegion (** **out int** *nTop*, **out int** *nLeft*, **out int** *nRight*, **out int** *nBottom*, **);**

C	**HRESULT IampImage_ getRemitCheckNumberRegion (** **LONG *** *nTop*, **LONG *** *nLeft*, **LONG *** *nRight*, **LONG *** *nBottom*, **int *** *returnValue* **);**
JAVA	**int getRemitCheckNumberRegion (** **int** *outputTop*, **int** *outputLeft*, **int** *outputRight*, **int** *outputBottom*, **);**
VB	**getRemitCheckNumberRegion (** *nTop* **as Integer,** *nLeft* **as Integer,** *nRight* **as Integer,** *nBottom* **as Integer** **)** **as Integer**

Parameters

nTop
The top pixel margin of the check number line.

nLeft
The left pixel margin of the check number line.

nRight
The right pixel margin of the check number line.

nBottom
The bottom pixel margin of the check number line.

Remarks

This function provides a simple interface for retrieving the check number region boundaries from the most recent successful ReadMICRRemit operation.

See Also

IampMICR::ReadMICRRemit

4.53. IampMICR::getRemitCheckRegion

The **IampMICR::getRemitCheckRegion** method returns the pixel boundaries of the region that contains the check last read with ReadMICRRemit.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT getRemitCheckRegion (** **LONG *** *nTop*, **LONG *** *nLeft*, **LONG *** *nRight*, **LONG *** *nBottom*, **int *** *returnValue* **);**

C#	**int getRemitCheckRegion (** **out int** *nTop*, **out int** *nLeft*, **out int** *nRight*, **out int** *nBottom*, **);**
C	**HRESULT IampImage_ getRemitCheckRegion (** **LONG *** *nTop,* **LONG *** *nLeft,* **LONG *** *nRight,* **LONG *** *nBottom,* **int *** *returnValue* **);**
JAVA	**int getRemitCheckRegion (** **int** *outputTop,* **int** *outputLeft,* **int** *outputRight,* **int** *outputBottom,* **);**
VB	**getRemitCheckRegion (** *nTop* **as Integer,** *nLeft* **as Integer,** *nRight* **as Integer,** *nBottom* **as Integer** **)** **as Integer**

Parameters

nTop
The top pixel margin of the check region.

nLeft
The left pixel margin of the check region.

nRight
The right pixel margin of the check region.

nBottom
The bottom pixel margin of the check region.

Remarks

This function provides a simple interface for retrieving the check region boundaries from the most recent successful ReadMICRRemit operation.

See Also

IampMICR::ReadMICRRemit

4.54. IampMICR::PrepareCouponImage

The **IampMICR::PrepareCouponImage** method uses special grayscale image processing and character shape (glyph) detection to find the corners in the grayscale source image and then produce a destination grayscale image that is cropped and deskewed.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT PrepareCouponImage (** **LONG** *nSourceImage*, **LONG** *nDestinationImage*, **PAMPPREPINFO** *pInfo*, **LONG** *nImageType*, **PAMPQUAD** *pVirtualCoord*, **LONG** *nMinDX*, **LONG** *nMinDY*, **LONG** *nMinGlyphs*, **LONG** *nTypeGlyphs*, **LONG*** *pnFoundGlyphs*, **LONG*** *pnGlyphX*, **LONG*** *pnGlyphY*, **LONG*** *pnGlyphDX*, **LONG*** *pnGlyphDY*, **LONG*** *pnStrengthLeftOut*, **LONG*** *pnStrengthRightOut*, **LONG*** *pnStrengthTopOut*, **LONG*** *pnStrengthBottomOut* **LONG*** *ampResult* **);**
C#	**int PrepareCouponImage (** **int** *nSourceImage*, **int** *nDestinationImage*, **int** *nBlackEdges*, **int** *nRotate*, **int** *nResolution*, **out double** *pdblSkewDetected* **int** *nSourceImage*, **int** *nDestinationImage*, **PAMPPREPINFO** *pInfo*, **int** *nImageType*, **PAMPQUAD** *pVirtualCoord*, **int** *nMinDX*, **int** *nMinDY*, **int** *nMinGlyphs*, **int** *nTypeGlyphs*, **out int *** *pnFoundGlyphs*, **out int *** *pnGlyphX*, **out int *** *pnGlyphY*, **out int *** *pnGlyphDX*, **out int *** *pnGlyphDY*, **out int *** *pnStrengthLeftOut*, **out int *** *pnStrengthRightOut*, **out int *** *pnStrengthTopOut*, **out int *** *pnStrengthBottomOut* **out int *** *ampResult* **);**

C	**HRESULT IampMICR_PrepareCouponImage (** **LONG** *nSourceImage*, **LONG** *nDestinationImage*, **PAMPPREPINFO** *pInfo*, **LONG** *nImageType*, **PAMPQUAD** *pVirtualCoord*, **LONG** *nMinDX*, **LONG** *nMinDY*, **LONG** *nMinGlyphs*, **LONG** *nTypeGlyphs*, **LONG*** *pnFoundGlyphs*, **LONG*** *pnGlyphX*, **LONG*** *pnGlyphY*, **LONG*** *pnGlyphDX*, **LONG*** *pnGlyphDY*, **LONG*** *pnStrengthLeftOut*, **LONG*** *pnStrengthRightOut*, **LONG*** *pnStrengthTopOut*, **LONG*** *pnStrengthBottomOut* **LONG*** *ampResult* **);**
JAVA	**int PrepareCouponImage (** **int** *nSourceImage*, **int** *nDestinationImage*, **PAMPPREPINFO** *pInfo*, **int** *nImageType*, **PAMPQUAD** *pVirtualCoord*, **int** *nMinDX*, **int** *nMinDY*, **int** *nMinGlyphs*, **int** *nTypeGlyphs*, **int** *pnFoundGlyphs*, **int** *pnGlyphX*, **int** *pnGlyphY*, **int** *pnGlyphDX*, **int** *pnGlyphDY*, **int** *pnStrengthLeftOut*, **int** *pnStrengthRightOut*, **int** *pnStrengthTopOut*, **int** *pnStrengthBottomOut* **);**
VB	**PrepareCouponImage (** **int** *nSourceImage* **as Integer,** **int** *nDestinationImage* **as Integer,** *pInfo* **as PAMPPREPINFO,** *nImageType* **as Integer,** *pVirtualCoord* **as PAMPQUAD,** *nMinDX* **as Integer,** *nMinDY* **as Integer,** *nMinGlyphs* **as Integer,** *nTypeGlyphs* **as Integer,** *pnFoundGlyphs* **as Integer,** *pnGlyphX* **as Integer,** *pnGlyphY* **as Integer,** *pnGlyphDX* **as Integer,** *pnGlyphDY* **as Integer,** *pnStrengthLeftOut* **as Integer,** *pnStrengthRightOut* **as Integer,** *pnStrengthTopOut* **as Integer,** *pnStrengthBottomOut* **as Integer** **)** **as Integer;**

Parameters

nSourceImage

The image buffer 0-9 is used as the source check image. A value of -1 will cause the contents of the main image property to be used. The source image must be grayscale.

nDestinationImage

The image buffer 0-9 is used as the destination image for the cropped and deskewed check image. A value of -1 tells the method to use the main image property as the destination. The destination image must not be the same as the source image.

pInfo – BlackEdges (input only)

A value of 0 tells the algorithm to look for white edges around the input check image. A value of 1 indicates black edge processing is enabled.

pInfo – rotation (input)

This input forces rotation of the input image or detect rotation.
Value: 0 – no rotation, 90 – rotate 90 degrees clockwise, -90 – rotate 90 degrees counter-clockwise, 1 or 91 – if rotation detected rotate 90 degrees clockwise, -1 or -91 – if rotation detected rotate 90 degrees counter-clockwise.

pInfo – rotation (output)

Indicates whether rotation occurred.
Value: 0 – no rotation, 1 – rotation of the amount and direction requested did happen.a

pInfo – resolution (input)

If this value is 0 then the source image resolution will be determined algorithmically. A nonzero value will force that resolution to be used during crop and deskew.

pInfo – resolution (output)

The detected horizontal resolution found during crop and deskew.

nImageType

This input value specifies the type of digitized document contained within the limits of the image.
Value: 0 - page, 1 – check, 5 – coupon, 6 - coupon A (no glyph detection), 7 - back side(no glyph detection)

pVirtualCoord

These output values are the XY coordinates of the document corners in the original input image.

nMinDX

This input used when nImageType = 7 indicates the minimum document width found on the front side. The size of the back side of a check or coupon should be the same as the front side. These parameters provide that information. In the future this could also be used to input expected size of a front size in a normalized fashion. For example it could describe the size expected if at 200 dpi. This would then be adjusted based on the actual dpi found.

nMinDY

This input used when nImageType = 7 indicates the minimum document height found on the front side. The size of the back side of a check or coupon should be the same as the front side. These parameters provide that information. In the future this could also be used to input expected size of a front size in a normalized fashion. For example it could describe the size expected if at 200 dpi. This would then be adjusted based on the actual dpi found.

nMinGlyphs

This input is used only when resolution detection is turned on.
Value: 0 – uses a default value depending on nImageType. Positive values are used as the minimum number of glyphs to find in the document subimage. For example, the default length of glyphs for a check is 10. There are numerous cases where this value should be modified. If a MICR font is used on a coupon but it only has a routing number, then the minGlyphs should be one less than the number of full size MICR fonts. Special characters should not be considered in the minGlyphs in this case. Hence the minGlyphs for a coupon with only a MICR routing number should be 8.

In the case of coupons, the typical glyph being found is an OCRA or OCRB font. In this case a minGlyph count should be at least half of the length of gyphs expected. By having a larger value for the min, random strings of characters on the image are less likely to have an effect or be detected as the glyph of interest.

nTypeGlyphs

This input is reserved – Value = 0. When implemented, this will allow the caller to identify the glyph and hence the GPI. For example:

Glyph	GPI
MICR	8
OCRA/B	10(typical)
Postal Barcode	22
General Barcode	random

pnFoundGlyphs

The number of glyphs found on the image.

pnGlyphX, pnGlyphY, pnGlyphDX, pnGlyphDY

The location and dimensions of the located glyph region. These values can be used to confirm that the resolution value is valid based on the location of the glyph on the image. For example, a glyph on a check at the top probably means it is upside down.

The DY value (height of the glyph region) will be taller because of any unadjusted skew, which should be minimal. A DY much taller than the expected glyph height usually means that multi lines of gyphs were found and the resulting resolution will be decreased.

pnStrengthLeftOut, pnStrengthRightOut, pnStrengthTopOut, pnStrengthBottomOut

These outputs are the confidence or strength of the position of the edges of the coupon region on the source image. Strength values can range from 0 thru 4 - number of character shapes recognized:

0 no edge detected. the best estimate of end of data used. End of data is simple threshold
1 edge detected that meets min length and intensity values.
2,3 corner found for this edge. E.G. left edge plus top edge meet hence a 2. If top and bottom meet left it is a 3.
4 To be determined.

When most of the edges are a zero, it means that the object was detected by its simple extent and not by edges. This means that skew correction probably did not occur.

ampResult

The error code returned. Error codes are defined in Appendix A.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

The selected input grayscale image must be loaded into the specified input buffer prior to cropping and deskew. A value of -1 for the input buffer will use the current contents of the main image property. The output cropped and deskewed grayscale image will be transferred to the main image. The algorithm used for cropping and deskewing is influence by the value in nImageType. An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating variables within this method.

The resolution of the destination image will be set to the found resolution (in x and y) if resolution detection was requested and was found. In the case of a coupon and when resolution was detected, the destination image will be scaled so that there is only a single resolution value for both x and y. The location of the result in the original image is reported in the virtualCoord values. The location of the gyphs will be reported relative to the result image.

See Also

Appendix G – PrepareCouponImage Software Example

4.55. IampMICR::PrepareMICRImage

The **IampMICR::PrepareMICRImage** method uses special bilevel and grayscale image processing to find the corners in the source check image and then produce a destination image that is cropped and deskewed.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT PrepareMICRImage (** **LONG** *nSourceImage,* **LONG** *nDestinationImage,* **LONG** *nBlackEdges,* **LONG** *nRotate,* **LONG** *nResolution,* **DOUBLE*** *pdblSkewDetected,* **LONG*** *ampResult* **);**
C#	**int PrepareMICRImage (** **int** *nSourceImage,* **int** *nDestinationImage,* **int** *nBlackEdges,* **int** *nRotate,* **int** *nResolution,* **out double** *pdblSkewDetected* **);**
C	**HRESULT lampMICR_PrepareMICRImage (** **LONG** *nSourceImage,* **LONG** *nDestinationImage,* **LONG** *nBlackEdges,* **LONG** *nRotate,* **LONG** *nResolution,* **DOUBLE*** *pdblSkewDetected,* **LONG*** *ampResult* **);**
JAVA	**int PrepareMICRImage (** **int** *nSourceImage,* **int** *nDestinationImage,* **int** *nBlackEdges,* **int** *nRotate,* **int** *nResolution,* **double** *pdblSkewDetected* **);**
VB	**PrepareMICRImage (** *nSourceImage* **as Integer,** *nDestinationImage* **as Integer,** *nBlackEdges* **as Integer,** *nRotate* **as Integer,** *nResolution* **as Integer,** *pdblSkewDetected* **as Double** **)** **as Integer;**

Parameters

nSourceImage
: The image buffer 0-9 is used as the source check image. A value of -1 will cause the contents of the main image property to be used. The source image must be bilevel or grayscale.

nDestinationImage
: The image buffer 0-9 is used as the destination image for the cropped and deskewed check image. A value of -1 tells the method to use the main image property as the destination. The destination image must not be the same as the source image.

nBlackEdges
: A value of 0 tells the algorithm to look for white edges around the input check image. A value of 1 indicates black edge processing is enabled.

nRotate
: This input controls whether the check image is rotated before being cropped and deskewed.

Value: 0 – no rotation, -1 – auto detect rotate left, 1, auto detect rotate right, -90 rotate left always, 90 rotate right always.

nResolution
If this value is 0 then the source image resolution will be used for the destination. A nonzero value will be used as the destination image resolution (e.g. 200).

pdblSkewDetected
The number of MICR characters recognized.

ampResult
The error code returned. Error codes are defined in Appendix A.

Return Values

If the function succeeds, the return value is zero. Otherwise, the return code is an error code.

Remarks

The selected input image must be loaded prior to cropping and deskew. If the source image is bilevel, the cropped and desked destination image will also be bilevel. If the source image is grayscale, the output image will be grayscale. The skew value of the source image is returned through the pdblSkewDetected variable. An exception will be thrown (e_NoMemory) if the system runs out of memory while allocating variables within this method.

See Also

Appendix A

4.56. IampMICR::ReadMICRRemit

The **IampMICR::ReadMICRRemit** method reads a remittance document and returns information fields.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT ReadMICRRemit (** **BSTR** **pszMICR*, **LONG** ***pnMICRCount,* **BSTR** **pszAmount*, **LONG** ***pnAmountCount,* **BSTR** **pszDate*, **LONG** ***pnDateCount,* **BSTR** **pszCheckNumber*, **LONG** ***pnCheckNumberCount,* **LONG** ***returnValue* **);**
C#	**int ReadMICRRemit (** **string** **pszMICR*, **int** ***pnMICRCount,* **string** **pszAmount*, **int** ***pnAmountCount,* **string** **pszDate*, **int** ***pnDateCount,* **string** **pszCheckNumber*, **int** ***pnCheckNumberCount* **);**

C	**HRESULT lampImage_ ReadMICRRemit (** **BSTR** **pszMICR*, **LONG *** *pnMICRCount,* **BSTR** **pszAmount*, **LONG *** *pnAmountCount,* **BSTR** **pszDate*, **LONG *** *pnDateCount,* **BSTR** **pszCheckNumber*, **LONG *** *pnCheckNumberCount,* **LONG *** *returnValue* **);**
JAVA	**int ReadMICRRemit (** **String** **pszMICR*, **int *** *pnMICRCount,* **String** **pszAmount*, **int *** *pnAmountCount,* **String** **pszDate*, **int *** *pnDateCount,* **String** **pszCheckNumber*, **int *** *pnCheckNumberCount* **);**
VB	**ReadMICRRemit (** *pszMICR* **as String**, *pnMICRCount* **as Integer**, *pszAmount* **as String**, *pnAmountCount* **as Integer**, *pszDate* **as String**, *pnDateCount* **as Integer**, *pszCheckNumber* **as String**, *pnCheckNumberCount* **as Integer** **)** **as Integer**

Parameters

pszMICR
The output string to contain the MICR line.

pnMICRCount
The number of characters in pszMICR.

pszAmount
The output string to contain the check amount.

pnAmountCount
The number of characters in pszAmount.

pszDate
The output string to contain the check date.

pnDateCount
The number of characters in pszDate.

pszCheckNumber
The output string to contain the check number.

pnCheckNumberCount
The number of characters in pszCheckNumber.

Remarks

This function reads a remittance document and provides output strings containing the MICR line, the amount, the date, and the check number.

See Also

4.57. IampMICR::enableCameraMode

The **IampImage::enableCameraMode** property determines whether ReadMICR will use special image processing logic before reading the MICR line.

Quick Info

See IampMICR: IDispatch.

C++	**HRESULT put_enableCameraMode (** **int** *enableCameraMode* **);** **HRESULT get_enableCameraMode (** **int*** *enableCameraMode* **);**
C#	**int** *enableCameraMode*
C	**HRESULT IampImage_put_enableCameraMode (** **int*** *enableCameraMode* **);** **HRESULT IampImage_get_enableCameraMode (** **int*** *enableCameraMode* **);**
JAVA	**void put_enableCameraMode (** **int*** *enableCameraMode* **);** **void get_enableCameraMode (** **int*** *enableCameraMode* **);**
VB	**enableCameraMode as Integer;**

Parameters

enableCameraMode
An integer value. If set to 1, camera-oriented image processing will be performed MICR line reading. If set to 0, this feature is disabled.

Remarks

Check images that have been acquired from a camera are frequently warped in a keystone fashion and have poor grayscale content. If the CameraMode property is enabled, ReadMICR will use special image processing to dewarp and threshold the image prior to reading the MICR line. The image object will not be updated with the dewarped and thresholded image. To do that, use the ReadCamera method. Internal to the COM object, ReadMICR uses ampReadMicrCamera when CameraMode is enabled.

See Also

IampMICR::ReadMICR

4.58. IampMICR::enableFullPage

The **IampImage::enableFullPage** property determines whether ReadMICR will search through the entire document looking for the MICR line.

Quick Info

See IampMICR: IDispatch.

C++	**HRESULT put_enableFullPage (** **int** *enableFullPage* **);** **HRESULT get_enableFullPage (** **int*** *enableFullPage* **);**
C#	**int** *enableFullPage*
C	**HRESULT lampImage_put_enableFullPage (** **int*** *enableFullPage* **);** **HRESULT lampImage_get_enableFullPage (** **int*** *enableFullPage* **);**
JAVA	**void put_enableFullPage (** **int*** *enableFullPage* **);** **void get_enableFullPage (** **int*** *enableFullPage* **);**
VB	**enableFullPage as Integer;**

Parameters

enableFullPage
An integer value. If set to 1, the full page remittance feature will be enabled. If set to 0, this feature will be disabled.

Remarks

Usually the MICR line is located near the bottom of a check image. This feature enables ReadMICR to search the entire image before reading the MICR line.

See Also

IampMICR::ReadMICR

4.59. IampMICR::enableScannerMode

The **IampImage::enableScannerMode** property determines whether ReadMICR will use full page image processing logic before reading the MICR line.

Quick Info

See IampMICR: IDispatch.

C++	**HRESULT put_enableScannerMode (** **int** *enableScannerMode* **);** **HRESULT get_enableScannerMode (** **int*** *enableScannerMode* **);**
C#	**int** *enableCameraMode*
C	**HRESULT lampImage_put_enableScannerMode (** **int*** *enableScannerMode* **);** **HRESULT lampImage_get_enableScannerMode (** **int*** *enableScannerMode* **);**

JAVA	**void put_enableScannerMode (** **int*** *enableScannerMode* **);** **void get_enableScannerMode (** **int*** *enableScannerMode* **);**
VB	**enableScannerMode as Integer;**

Parameters

enableScannerMode
An integer value. If set to 1, scanner-oriented image processing will be performed before and during MICR line reading. If set to 0, this feature is disabled.

Remarks

Check images that have been acquired from a flatbed scanner may be located anywhere in an 8.5x11 inch bitmap. If the ScannerMode property is enabled, ReadMICR will use special image processing to isolate the image while reading the MICR line. The image object will not be updated with the isolated. To do that, use the ReadScanner method. Internal to the COM object, ReadMICR uses ampReadMicrScanner when ScannerMode is enabled.

See Also

IampMICR::ReadMICR

4.60. IampMICR::done180

The **IampMICR::done180** property indicates whether or not a 180 degree rotation was necessary to read the most recent image.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT get_done180 (** **int*** *done180* **);**
C#	**int** *done180*
C	**HRESULT IampBarcode_get_done180 (** **int*** *done180* **);**
JAVA	**void get_done180 (** **int*** *done180* **);**
VB	**done180 () as Integer;**

Parameters

done180
A pointer to an integer variable that will receive the value.

4.61. IampMICR::doneCombo

The **IampMICR::doneCombo** property indicates whether or not combo mode was used to read the most recent image.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT get_doneCombo (** **int*** *doneCombo* **);**
C#	**int** *doneCombo*
C	**HRESULT IampBarcode_get_doneCombo (** **int*** *doneCombo* **);**
JAVA	**void get_doneCombo(** **int*** *doneCombo* **);**
VB	**DoneCombo () as Integer;**

Parameters

doneCombo
A pointer to an integer variable that will receive the value.

4.62. IampMICR::doneImageUpdate

The **IampMICR::doneImageUpdate** property indicates whether or not image enhancements were necessary to read the most recent image.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT get_doneImageUpdate (** **int*** *doneImageUpdate* **);**
C#	**int** *doneImageUpdate*
C	**HRESULT IampBarcode_get_doneImageUpdate (** **int*** *doneImageUpdate* **);**
JAVA	**void get_doneImageUpdate (** **int*** *doneImageUpdate* **);**
VB	**doneImageUpdate () as Integer;**

Parameters

doneImageUpdate
A pointer to an integer variable that will receive the value.

4.63. IampMICR::doneRepair

The **IampMICR::doneRepair** property indicates whether or not special processing was needed to read the most recent image.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT get_doneRepair (** **int*** *doneRepair* **);**
C#	**int** *doneRepair*
C	**HRESULT IampBarcode_get_doneRepair (** **int*** *doneRepair* **);**
JAVA	**void get_doneRepair (** **int*** *doneRepair* **);**
VB	**doneRepair () as Integer;**

Parameters

doneRepair
A pointer to an integer variable that will receive the value.

Remarks

This property is usually set to zero.

4.64. IampMICR::skew

The **IampMICR::skew** property returns the detected skew ratio of the MICR baseline after a read operation.

Quick Info

See IampMICR : IDispatch.

C++	**HRESULT get_skew (** **double*** *dblSkew* **);**
C#	**double** *dblSkew*
C	**HRESULT IampBarcode_get_dblSkew (** **double*** *dblSkew* **);**
JAVA	**void get_dblSkew (** **double*** *dblSkew* **);**
VB	**dblSkew () as Double;**

Parameters

dblSkew
A pointer to a double variable that will receive the value.

Remarks

The skew ratio is calculated as (dy/dx). The origin is in the top left corner, so the sign will be inverted.

5. Appendix A

AMPLIB Error Codes

1	Could not allocate PC memory space. A local or global allocation failed that was needed to complete the requested operation.
3	Specified work image does not exist. No image by the given name can be located.
4	Name already in use.
6	Not a primary image. An alias image may not be used in this instance.
10	AMPLIB cannot support any more tasks. The maximum number of callers has already been reached.
11	Internal error. A software error has been detected in the AMPLIB system. Please report this to AllMyPapers Technical Support.
12	Image bounds exceeded. The requested DX, DY, X, Y values exceed the values allowed for this image, as given by MaxHeight and MaxWidth, or the requested sub-image lies outside of the current image dimensions.
13	Image metrics error. The requested sub-image lies outside of the current image dimensions.
14	Internal error calling the Windows API.
15	Bad handle passed to function. The given handle is incorrect or inappropriate for the function in question.
16	User interrupt. A function terminated because of an improper call.
19	AMP function call error. There is an error in the arguments passed to the function in question.
20	No size information. The image has not yet been loaded with any image data and thus has no dimensions.
21	No cross-board operations are allowed. You may not perform an operation where the source and destination image operands reside on different co-processors.
22	Incompatible image sizes. When a destination image is fixed size, the result image must be less than or equal to the size of the destination.
23	Bad file name. The file path name given is incorrect or cannot be opened.
24	I/O error. The I/O system reported an error during execution of this function.
25	Cannot open trace file.
26	An invalid compression type was given.
27	An internal TIFF operation failed. In the processing of the IFD list or header, some critical operation failed.
28	Required TIFF tag missing. The TIFF 6.0 Specification defines those tags which at a minimum must be present in all baseline TIFF files. One of those tags is missing.
29	Image organization not supported. Only 1 bit per pixel bi-level images are supported.
30	This system is unable to run AMPLIB. Call AllMyPapers Technical Support.
31	Unable to open the requested TIFF file. It may not be a TIFF file, or has an invalid header.
32	The requested image within a multi-image TIFF file is not in the image file directory of that TIFF file.
33	An error occurred while reading the TIFF IFD.
34	The KDY value given for Group 3 2d compression is invalid.
35	Assertion logic error. Some internal software data or pointer consistency has occurred.

36 No region has been selected to support the requested operation.

37 The number passed to the function is out of range.

40 The resolution value given is not valid.

41 The page size value given is not valid.

42 The operation type is not valid.

43 The mode given is not valid.

46 The scale ratio given is not valid for this operation.

47 One of the arguments passed to the function is invalid.

48 AMPLIB is unable to create the requested file. This is most likely due to an invalid path or some I/O permission error.

49 The margins are not legal for the page size.

51 No file specification was given and is required for this operation.

52 No index string was found in the file path name string.

53 Huge objects not supported yet.

54 The clipboard is empty.

55 General error. No detail available.

56 Download failure.

60 General printer failure.

62 A bad tag was found in a TIFF file.

64 An invalid TIFF header was detected.

65 Scaling while printing requires buffered print mode.

66 Source and destination images must be different.

67 The function in question timed out.

68 A callback function returned an error.

69 Application lockout.

70 This version of AMPLIB is not correct for this application.

71 An invalid file type was specified.

72 The image IX value must be a multiple of 32 for this operation.

73 The margins specified are not legal for this operation.

74 The requested TIFF tag already exists in the IFD list.

75 An invalid Optika header was detected.

76 The requested file format is unsupported in this mode.

83 No ensigns defined or allowed.

84 Bad MODCA RECID parameter

85 IBM MMR format not supported

86 Unsupported compression type

87 Decompression error

88 Unsupported MODCA or IOCA file format

89 Compression error

90 Thread already attached to DLL

91 Disk is full

92 File Access error

93 Too many files open

94 File exists

95 Bad file handle

96 No such file or directory

98	Thread not attached
101	No object data in image block
102	Can't find a needed DLL
103	Can't find entry point in DLL
104	License file fails security check
105	License check detected date rollback
106	License expired
107	License required for this feature
108	Image degenerated to dx=0 or dy=0
113	Software implementation only
114	Not an AMPLib PDF file
115	Error parsing PDF file
116	Missing files/files not loaded
117	License computer id error
118	Problem opening license file
119	Problem opening TWAIN device
120	Problem reading paper sensor on TWAIN device
121	Scanner Timeout
122	Not supported scanner
123	No image acquired while scanning
124	Failure during image warp
125	Failure during Data Matrix read
126	Invalid JBIG header
127	JBIG decompression problem
128	Failure while rotating image
129	CLICK count exhausted
130	No image content(all black/white or nearly so)
150	Exception happended in the wrapper accessing AmpLib.dll
161	Failure during Quick Response barcode read
201	No pointer to the TLS available
202	Input image error – low resolution or not enough levels of gray

5. Appendix B – AmpLibNet and COM Object Methods to AMPLIB API Guide

Shown below is a list of AmpLibNet/COM object functions and properties along with each corresponding AMPLIB API function.

AnnotateImage (ampAnnotateImage)
AssembleMICR(ampAssembleMICR)
FilterImage (ampFilterImage)
FormatMICRFields(ampFormatMICRFields)
GetAmplibVersion (ampGetFileVersion)
GetBarCodes (ampGetBarCodeData)
GetLicenseInfo (ampGetLicenseInfo)
GetMessage (ampGetMessageText)
GetRunsInfo (ampGetRunsInfo)
getMICRFields (ampParseMicr)
GetScaledImageAddress (ampScaleImage, ampGrayQuickScaleImage, ampGetImageAddress)
GetWindow (ampGetImageMetrics)
InterpolateGrayImage (ampGrayInterpolate)
LoadBlankImage (ampCreateWorkImage, ampCreateGrayWorkImage)
LoadClipboardImage (ampLoadClipboard)
LoadImage (ampLoadImage)
LoadImageBuffer (ampLoadImageBuffer)
LoadMICRImage (ampLoadImage, ampDynamicThreshold, ampScaleImage, ampReadMicr)
PasteImageFromBuffer (ampCopyImage)
pixelBitDepth (ampGetGrayImageMetrics)
PrepareCouponImage (ampSobelEdgePrepEx)
PrepareMICRImage (ampPrepMicr)
ProcessGrayImage (ampGrayProcesses)
PromoteBilevelImage (ampConvertImage)
ReadCamera, GetReadCameraResults (ampReadCamera)
ReadMICR (ampReadMicr, ampReadMicrCamera, ampReadMicrPage, ampPrepMicr, ampScaleImage, ampFilterImage, ampDeBack, ampDeSpec, ampReadOCR)
ReadMICRPage (ampReadMicrPage, ampPrepMicr, ampScaleImage, ampFilterImage)
ReadMICRRemit, GetRemitAmountRegion, GetRemitDateRegion,
GetRemitCheckNumberRegion, GetRemitCheckRegion, getMICRData (ampReadMicrRemit)
ReadScanner, GetReadScannerResults (ampReadScannerForChecks)
RotateImageToBuffer (ampRotateImage)
SaveImage (ampSaveImage)
SaveImageToClipboard (ampSaveClipboard)
ScaleImageToBuffer (ampScaleImage)
ScaleWidthResolution (ampScaleWidthResolution)
SaveImageToMemory (ampSaveImageMem)
SaveImageToMemoryTest (ampSaveImageMem)
ScanBarCodes (ampReadBar)
SecurityEnableAppsFile (ampSecurityEnableAppsFile)
SetWindow (ampSetImageMetrics)
ThresholdGrayImage (ampDynamicThreshold, ampGrayAdaptiveThreshold, ampGreyImageProcess)
traceEnable (ampTraceEnable)
traceFile (ampTraceEnable)
VerifyMICR (ampVoteMicrRetry, ampVoteIRDRetry)
VerifyMICRField (ampFieldVerifyEx)

6. Appendix C – AMPLIB API Calls not used by AMPLib COM/AmpLibNet

Shown below is a list of AMPLIB API calls not used by AMPLib COM or AmpLibNet.

MICR Functions
ampPrepPage
ampReadMicrRepair
ampReadMicrDouble
ampVoteIRDRepair
ampVoteMicrRepair

Image Manipulation Functions
ampBitBltImage
ampDeSkew
ampDitherImage
ampGrayMirrorImage
ampMirrorImage

Image Filtering Functions
ampDeBorder
ampDeLine
ampDeShade
ampDeStreak

Miscellaneous Functions
ampPutImageBlock

7. Appendix D – AMPLib COM/AmpLibNet functionality not present in AMPLIB DLL

Shown below is a list of AMPLib COM functions not present in AMPLIB DLL.

AutoPrep - The AMPLib COM Object's AutoPrep feature uses AMPLIB DLL functionality to optimize MICR Read results on an image. If AutoPrep is enabled when an image is loaded (LoadImage) into the image object, the COM object will perform several combinations of image processing, thresholding, and MICR Read operations to find the best settings for MICR reading. Once the optimal settings are found, the image will be scaled to 200 dpi. AutoPrep is primarily useful for grayscale images of unknown resolution.

ReadMICR - In the AMPLib COM Object, the MICR Read operation incorporates elements of the All My Papers MICRBatch program (Combo Mode, Filter, ImagePrep, OCR A, and OCR B) to read MICR lines robustly.

WorkImage Encapsulation - The AMPLib COM object provides an array of 10 permanently allocated image buffers, which encapsulate the direct manual allocation calls available in AMPLIB DLL.

Image Display Styles - In the AmpCOMDemo programs, images can be displayed in a variety of different zoom and scaling styles.

MICR Result Image Overlay - In the AmpCOMDemo programs, MICR results are displayed over the check image after a successful MICR Read operation.

8. Appendix E – Module Redistribution

The typical AMPLib COM object installation places executables and DLL support files in the C:\Program Files\AllMyPapers\AmplibCOM folder. If for example you create a Visual Basic application that references the AMPLib COM object, you may choose to use a different subdirectory to hold that application and the AMPLib support files on the target PC. These files are:

amplib.dll
amplibif.dll and interop.amplibif.dll (or AmpLibNet.dll)
amplm.dll
LogosDM.dll
ampPX.dll
storage.dat
lvscmc7.fnt
lvsmicr.fnt
lvsocrb.fnt
lvsocra.fnt
lvsfsb.fnt
lvscar.fnt
lvsnum.fnt
lvssquare.fnt

For a 64-bit AmpLibNet install, all DLLs should have a 64 suffix as in: AmpLib64.dll, AmpLibNet64.dll, AmpLM64.dll, LogosDM64.DLL, and ampPX64.dll.

Next, the AX9Lib COM object must be registered on the PC so that it will be “visible” to your application. This can be done automatically if you are using a product like InstallShield. Otherwise, use the following call to the regsvr32 application using a batch file or exec call:

regsvr32 amplibif.dll

9. Appendix F – ReadCamera Software Example

The ReadCamera method offers a powerful means to process check front and back images captured from cellphone cameras. ReadCamera therefore needs up to two images for input and can provide one or two images as output. The CSharp code below is an example of how to allocate the image buffers in AmpLibNet as ReadCamera input and output images.

```
using System;
using System.IO;

using System.Collections.Generic;
using System.Text;
using System.Runtime.InteropServices;
using AmpLibNetClasses;

namespace ImageinMemoryTest
{
    class Program
    {
        static void Main(string[] args)
        {
            AmpLibNet micrLib = new AmpLibNet();
            // See the CSharp AmpLibNet demo Form1_Load routine for these next two lines of code
            // AMPLibCode ampC = new AMPLibCode(this);
            // ampC.CSInitAmplib(micrLib);

            int status;

            micrLib.grayImageEnable = 1;
            // Load the image file into the main AmpLibNet image property
            status = micrLib.LoadImage(@"D:\CustomerData\Amp Crop Failures\Amp Testing
Redact\Front\N00100133091345443403013 8.jpg", "", "", 1);
            if (status == 0)
            {
                // Transfer the image in the main image property to image buffer 1 of 1-10
                status = micrLib.CopyImageToBuffer(1);
            }
            if (status == 0)
            {
                // Load the image file into the main AmpLibNet image property
                status = micrLib.LoadImage(@"D:\CustomerData\Amp Crop Failures\Amp Testing
Redact\Front\N00100133091345443403013 8.jpg", "", "", 1);
            }
            if (status == 0)
            {
                // Transfer the image in the main image property to image buffer 2 of 1-10
                status = micrLib.CopyImageToBuffer(2);
            }
            if (status > 0)
            {
                String msg = String.Format("LoadImage failed with error = {0} - ", status);
                return;
            }

            String CameraOptions = "CARQ=0M=85";

            String MICROutput = "";
            int MICRLength = 0;

            string strAuxOnUs = "";
            string strRoute = "";
            string strOnUs = "";

            int nMicrOK = 0;
            int nMicrConf = 0;
            int nImagesOK = 0;
            int nImagesTooDark = 0;
            int nImageTooLight = 0;
            int nImgNotSize = 0;
            int nRearMissing = 0;
            int nFrontMissing = 0;

            try
            {
                // Read the camera front and back images from image buffers 1 and 2 transferring the output
                // to AmpLibNet image buffers 3 and 4
                status = micrLib.ReadCamera(1, 2, 3, 4, CameraOptions, ref MICROutput, ref MICRLength);
```

```
                if (status == 0)
                {
                    status = micrLib.GetReadCameraResults(ref strAuxOnUs, ref strRoute, ref strOnUs, ref nMicrOK,
                        ref nMicrConf, ref nImagesOK, ref nImagesTooDark, ref nImageTooLight, ref nImgNotSize,
                        ref nRearMissing, ref nFrontMissing);
                    Console.WriteLine(MICROutput);
                }
                if (status == 0)
                {
                    status = micrLib.PasteImageFromBuffer(3);
                }
                if (status == 0)
                {
                    status = micrLib.SaveImage("D:\\frontout.tif", "B=1", "tif");
                }
                if (status == 0)
                {
                    status = micrLib.PasteImageFromBuffer(4);
                }

                if (status == 0)
                {
                    status = micrLib.SaveImage("D:\\rearout.tif", "B=1", "tif");
                }

            }
            catch (System.Exception ex)
            {
                Console.WriteLine(ex.Message);
            }

            if (status > 0)
            {
                String msg = String.Format("Recognition failled with  = ", status);

                return;
            }
        return;
        }
    }
}
```

10. Appendix G – PrepareCouponImage Software Example

The PrepareCouponImage method offers a powerful means to crop and deskew coupon, check, and other document image types captured from cellphone cameras. PrepareCouponImage uses the main AmpLibNet image property for input if the input value is -1 otherwise it uses one of the 10 image buffers. Output to any of the 10 image buffers is also routed back to the main image property for viewing in the application. The VB code below is from the button handler code in the demo application.

```
    Private Sub btnPrepareImage_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnPrepareCouponImage.Click

        Dim nStat As Integer = 0    'stores return codes for ampobj functions

        Dim strTemp As String = ""

        'front and back check image source and destination buffers
        Dim nSource As Integer
        Dim nDestination As Integer

        Dim PrepInfo As AmpLibAPI.ampPrepInfo
        Dim VirtualCoord As AmpLibAPI.ampQUAD

        Dim nBlackEdges As Integer = 0
        Dim nImageType As Integer = 0
        Dim nResolution As Integer = 0
        Dim nRotate As Integer = 0
        Dim nMinDX As Integer = 0
        Dim nMinDY As Integer = 0
        Dim nMinGlyphs As Integer = 0
        Dim nTypeGlyphs As Integer = 0
        Dim nFoundGlyphs As Integer = 0
        Dim nGlyphDX As Integer = 0
        Dim nGlyphDY As Integer = 0
        Dim nGlyphX As Integer = 0
        Dim nGlyphY As Integer = 0

        Dim nStrengthLeftOut As Integer = 0
        Dim nStrengthRightOut As Integer = 0
        Dim nStrengthTopOut As Integer = 0
        Dim nStrengthBottomOut As Integer = 0

        'temp message strings
        Dim msg As String = ""
        Dim msg2 As String = ""

        'read front and back check image source and destination buffers from the interface
        nSource = (Integer.Parse(piSourceBuffer.Text))
        nDestination = (Integer.Parse(piDestinationBuffer.Text))

        PrepInfo.ZeroMemory()

        VirtualCoord.ulcX = 0
        VirtualCoord.ulcY = 0
        VirtualCoord.urcX = 0
        VirtualCoord.urcY = 0
        VirtualCoord.llcX = 0
        VirtualCoord.llcY = 0
        VirtualCoord.lrcX = 0
        VirtualCoord.lrcY = 0

        Try
            ' Read the prepare image input from the user interface
            '  Sobel Input
            'nImageType       = 0  page
            '            = 1  check
            '            = 2  check sides (second phase of detrap, two pases)
            '            = 4  check dewrap (one pass)
            '            = 5  coupon
            '            = 6  coupon A ( no glyph)
            '            = 7  back side( no glyph)
            '
            ' minDx,minDy used for minimun  back side

            If piPage.Checked Then
                nImageType = AmpLibAPI.PREPPAGE
            ElseIf piCheck.Checked Then
```

```
                nImageType = AmpLibAPI.PREPCHECK
            ElseIf piBack.Checked Then
                nImageType = AmpLibAPI.PREPBACK
            ElseIf piCoupon.Checked Then
                nImageType = AmpLibAPI.PREPCOUPON
            ElseIf piCouponA.Checked Then
                nImageType = AmpLibAPI.PREPCOUPONA
            End If

            If piBlackEdges.Checked Then
                nBlackEdges = 1
            Else
                nBlackEdges = 0
            End If

            nResolution = Integer.Parse(piResolution.Text)
            nRotate = Integer.Parse(piRotation.Text)

            PrepInfo.BlackEdges = nBlackEdges
            PrepInfo.Resolution = nResolution
            PrepInfo.Rotate = nRotate

            nMinDX = Integer.Parse(piMinDX.Text)
            nMinDY = Integer.Parse(piMinDY.Text)
            nMinGlyphs = Integer.Parse(piMinGlyphs.Text)
            nTypeGlyphs = Integer.Parse(piTypeGlyphs.Text)

        Catch
        End Try

        If g_nActiveOperation <> AO_NONE Then
            DisplayBusyMessage("prep Coupon")
            Return
        End If

       g_nActiveOperation = AO_PREPMICR

        ' Turn on the hourglass cursor
        Me.Cursor = System.Windows.Forms.Cursors.WaitCursor

        Try
            'Prepare Coupon Image
            nStat = g_ampLibNetObj.PrepareCouponImage(nSource, nDestination, PrepInfo, nImageType, VirtualCoord, _
                            nMinDX, nMinDY, nMinGlyphs, nTypeGlyphs, nFoundGlyphs, _
                            nGlyphX, nGlyphY, nGlyphDX, nGlyphDY, _
                            nStrengthLeftOut, nStrengthRightOut, nStrengthTopOut, nStrengthBottomOut)

            g_nActiveOperation = AO_NONE

            ' Turn off the hourglass cursor
            Me.Cursor = System.Windows.Forms.Cursors.Default

        Catch ex As Exception
            g_nActiveOperation = AO_NONE

            ' Turn off the hourglass cursor
            Me.Cursor = System.Windows.Forms.Cursors.Default

            ' If there is no status error because of the exception, then create one
            If nStat = 0 Then
                nStat = 11      ' Internal error.  A software error has been detected in the AMPLIB system.  Please
report this to AllMyPapers Technical Support.
            End If

            MsgBox("There was an exception calling PrepareCouponImage: " & ex.Source & "-" & ex.Message)
            Return
        End Try

        If (nStat <> 0) Then
            msg = String.Format("PrepareCouponImage returned with error = {0} - ", nStat)
            g_ampLibNetObj.GetMessage(nStat, msg2)
            msg = msg + msg2
            MessageBox.Show(msg, "AMPLIB Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            Return
        End If

        'if the call was successful, display the results
        If (nStat = 0) Then
            piGlyphCount.Text = nFoundGlyphs.ToString
            piGlyphX.Text = nGlyphX.ToString
            piGlyphY.Text = nGlyphY.ToString
            piGlyphDX.Text = nGlyphDX.ToString
            piGlyphDY.Text = nGlyphDY.ToString
            piOutResolution.Text = PrepInfo.Resolution.ToString
            piOutRotation.Text = PrepInfo.Rotate.ToString
            piStrengthLeft.Text = nStrengthLeftOut.ToString
            piStrengthTop.Text = nStrengthTopOut.ToString
            piStrengthRight.Text = nStrengthRightOut.ToString
            piStrengthBottom.Text = nStrengthBottomOut.ToString
```

```
            piULCX.Text = VirtualCoord.ulcX.ToString
            piULCY.Text = VirtualCoord.ulcY.ToString
            piURCX.Text = VirtualCoord.urcX.ToString
            piURCY.Text = VirtualCoord.urcY.ToString
            piLLCX.Text = VirtualCoord.llcX.ToString
            piLLCY.Text = VirtualCoord.llcY.ToString
            piLRCX.Text = VirtualCoord.lrcX.ToString
            piLRCY.Text = VirtualCoord.lrcY.ToString

            g_bDrawOutputGlyphBoxOnImage = True
            dblOutputGlyphBoxTop = nGlyphY
            dblOutputGlyphBoxLeft = nGlyphX
            dblOutputGlyphBoxRight = nGlyphX + nGlyphDX
            dblOutputGlyphBoxBottom = nGlyphY + nGlyphDY
        End If

        'if a destination has been selected
        If nStat = 0 And nDestination <> -1 Then
            'paste the destination buffer contents into the main image object
            g_ampLibNetObj.PasteImageFromBuffer(nDestination)
            ScaleImage(1)
            UpdateDisplay()
        ElseIf nStat = 0 And nDestination = -1 Then
            'the destination was the main image object
            ScaleImage(1)
            UpdateDisplay()
        End If

        ''if the call was successful, display the results in a message box
        'If (nStat = 0) Then
        '    msg = "Detected Skew: " + dblSkewDetected.ToString + vbCrLf
        '    MessageBox.Show(msg, "PrepareMICRImage", MessageBoxButtons.OK)
        'End If

        Me.imgBox.Invalidate()
    End Sub
```