



# Ciansoft TwainControlX User Manual

## Introduction

Ciansoft TwainControlX is an ActiveX control that enables applications to acquire images from TWAIN compliant devices such as scanners and digital cameras. This document contains comprehensive instructions on installing and using TwainControlX.

For further information, visit our website: [www.ciansoft.com](http://www.ciansoft.com).

Alternatively, contact us by email, we will be pleased to answer your questions: [info@ciansoft.com](mailto:info@ciansoft.com).

## How to use these Instructions

These instructions are divided into several main sections. First, the installation of the control is described and there is some information about the trial version. The next four sections explain how to use the control in a programming environment to import images from a TWAIN device. There are generally four steps involved in acquiring an image and each section relates to one of these steps:

1. Selecting the TWAIN device to be used.
2. Configuring the settings of the device, e.g. resolution.
3. Acquiring the image
4. Exporting the image from TwainControlX, e.g. by displaying it in another control or saving to a file.

At its simplest, a programme to acquire an image can consist of just three lines of code as shown in the following Visual Basic example. As you will see, each line of code relates to one of the above mentioned steps. Note: step 2 is omitted in this example as it is not necessary if all default settings are accepted.

```
Twain1.SelectDevice           ' Presents a dialog listing the
                              ' available devices to select from
Twain1.Acquire                ' Acquires an image from the device
Twain1.SaveToFile "C:\Image.jpg" ' Saves the image to a file
```

A further section describes how to use TwainControlX to acquire multiple images, e.g., from a scanner equipped with an automatic document feeder (ADF).

Finally, there is a section on miscellaneous functions and some notes on deploying an application that uses TwainControlX.

For Visual Basic users, a simple example project is included with the installation. This can be accessed from the Windows Start Menu. Additional examples in Visual Basic and a range of other programming languages can be found on the Ciansoft website.

## Installation

The OCX file, TwainControlX.ocx, must be registered on the computer running the application. This should be done during the installation, but if it is not, the command line utility Regsvr32.exe can be used. This is usually found in the Windows system folder and runs using the syntax:

```
regsvr32 ocxname
```

where *ocxname* is the path and name of the ocx file to register. When deploying an application that uses TwainControlX, the ocx file will also need to be registered on the target machine. Note that the

licence file, TwainControlX.lic, is needed to use this control in a design environment and this file must not be distributed with an application, as is confirmed in your licence agreement.

To use this control in a Visual Basic project, select Project and Components from the pull down menu. This gives a list of available controls. Select TwainControlX Library and click Apply. This adds the control to the component palette. The class name is “Twain” and this will appear in the Object Browser where the properties and methods are listed. For other design environments, consult the documentation for importing ActiveX controls.

As an ActiveX control, TwainControlX can be used in a range of Windows based environments and languages. There will be slight differences in syntax especially with the use of parentheses/brackets surrounding method parameters.

TwainControlX makes use of the TWAIN Source Manager which is in the file “TWAIN\_32.DLL” and several associated files. These files are installed as standard with all versions of Microsoft Windows from 98 onwards. They are not included with Windows 95 but are likely to be present on the system if any TWAIN device has been installed previously. They are usually located in the Windows directory.

## The Trial Version

The trial version of TwainControlX is supplied as a different OCX file, called TwainControlXTrial.ocx, and has a separate installation programme. The trial version has all the functionality of the full version of the control, with the exception of the *AcquireToFile* method which is not available. The only other limitation is that each image acquired using the control will have a line of text written on the image indicating that trial software was used in its creation. Visit [www.ciansoft.com](http://www.ciansoft.com) to purchase the full version.

## Selecting the Device

The first step to acquiring an image is to select the hardware device to be used. There are three alternative ways to do this. The first is to use the *SelectDevice* method which will bring up a dialog box offering the user a choice from the list of devices installed on the system. Alternatively, the device number can be set directly using the *CurrentDevice* property or the *SelectDeviceByName* function can be called if the exact name is known.

Two other properties are provided to give access to information about the installed devices. These are *DeviceCount* and *DeviceName*. These could be used, for example, to create a custom dialog for device selection rather than using the default dialog in *SelectDevice*.

Note that all hardware devices that have been installed on the system, i.e. their drivers are installed, will be available at this stage, regardless of whether or not they are physically connected to the system.

### Selecting the Device – Methods and Properties

**SelectDevice** As Boolean - Read-only property. Displays a standard dialog box containing a list of the devices installed on the system. After selecting from the list and clicking “Select”, the value of *CurrentDevice* will be set automatically. The return value is True if a device was selected and False if the user clicked the Cancel button in the dialog box.

**CurrentDevice** As Long - The index number of the currently selected TWAIN device. The first installed device on the system is number 0, so for example, on a system with 3 installed devices the possible values of *CurrentDevice* would be 0, 1 or 2. A value of -1 indicates that no device is selected. (Default = -1).

**SelectDeviceByName** (*Name* As String) As Boolean - Selects a TWAIN device by its name, which must be an exact match with the name as it would be returned by the *DeviceName* property. The name is case-sensitive. The return value is True if the device is successfully found and selected.

**DeviceCount** As Long - Read-only. The number of TWAIN compliant devices currently installed on the system. Note that this property counts all installed devices, it does not matter whether they are physically connected at the present time.

**DeviceName** (*Index* As Long) As String - Read-only. The name of the device referenced by *Index*.

## Device Configuration

All TWAIN hardware devices have a built-in user interface that is normally displayed when the device is used. This allows the user to adjust settings such as resolution and image size before acquiring the image. By default, TwainControlX will display this interface, in which case it is not necessary to configure the device first.

By setting the *UseInterface* property to False, this behaviour is overridden. In this case the image is acquired without displaying the interface and it will be necessary to configure the device first.

Several properties are available for configuration including *Resolution*, *PixelFormat*, *Units* and properties to define the size of the image.

When values are set programmatically for these properties, TwainControlX will communicate directly with the currently selected device. At this stage it is necessary for the device to be connected and you can check this using the *Connected* property. Some values of properties cannot be set on some devices and this can be checked using other properties, e.g. *PixelFormatAllowed*.

### Device Configuration – Methods and Properties

**Connected** As Boolean - Read-only. If this property is True, TwainControlX is successfully communicating with the currently selected device. This is normally taken as confirmation that the device is physically connected to the system, however, the drivers of some hardware devices will respond on behalf of the device even when it is not present and this can give a misleading result. In such a case, your application may only get an indication of the problem when it attempts to acquire an image, and fails to do so.

**Units** As TxTwainUnits - The units of measure to be used for any measurement related to the image. This is an enumerated property which can take one of the following values:

unInches:	Inches.	0
unCentimeters:	Centimeters.	1
unPicas:	Picas.	2
unPoints:	Points.	3
unTwips:	Twips.	4
unPixels:	Pixels.	5
unUnknown:	No recognisable response from device.	-1

**UnitsAllowed** (*UnitType* As TxTwainUnits) As Boolean - Read-only. Returns True if the currently selected device will support the specified value for the *Units* property.

**PixelFormat** As TxPixelFormat - The type of colour format used to define each pixel in the image. This is an enumerated property which can take any of the following values:

ptBW:	Black and white.	0
ptGray:	Greyscale.	1
ptRGB:	Red/Green/Blue Colour.	2
ptUnknown:	No recognisable response from device.	-1

**PixelFormatAllowed** (*PixType* As *TxPixelFormat*) As Boolean - Read-only. Returns True if the currently selected device will support the specified value for the *PixelFormat* property.

**Resolution** As Double - The resolution of the image in pixels per unit of measure. For example, if *Units* is set to *unInches*, a value of 300 indicates a resolution of 300 dpi (dots per inch). Devices will only support particular values for *Resolution* as indicated by the properties *ResolutionMin*, *ResolutionMax* and *ResolutionStep*. If *Resolution* is set to an unsupported value, the nearest allowable value will be used instead.

**ResolutionMin** As Double - Read-only. The minimum resolution of the image in pixels per unit of measure that is supported by the currently selected device.

**ResolutionMax** As Double - Read-only. The maximum resolution of the image in pixels per unit of measure that is supported by the currently selected device.

**ResolutionStep** As Double - Read-only. The step size in supported resolutions of the currently selected device. For example, if a device has *ResolutionMin* = 100, *ResolutionMax* = 500 and *ResolutionStep* = 50, it will support *Resolution* values of 100, 150, 200, 250,....etc..., 500.

If *ResolutionStep* has the value -1, this indicates that the step size between the minimum and maximum values is not uniform.

**ImageLeft** As Double - The position of the left side of the image, measured in *Units* from the left side of the device. For example, if *Units* is set to *unInches* and *ImageLeft* = 1.5, an image acquired from a scanner will begin 1.5 inches from the left side of the scanner.

**ImageTop** As Double - The position of the top of the image, measured in *Units* from the top of the device.

**ImageRight** As Double - The position of the right side of the image, measured in *Units* from the left side of the device.

**ImageBottom** As Double - The position of the bottom of the image, measured in *Units* from the top of the device.

**SetImageLayout** (*Left* As Double, *Right* As Double, *Top* As Double, *Bottom* As Double) - This method allows the *ImageLeft*, *ImageRight*, etc. properties to be set simultaneously using a single command.

**MaxWidth** As Double - Read-only. The maximum physical width of the device in *Units*, e.g., for a flatbed scanner, this will return the size of the flatbed. For devices with no meaningful size, e.g., cameras, -1 is returned.

**MaxHeight** As Double - Read-only. The maximum physical height of the device in *Units*, e.g., for a flatbed scanner, this will return the size of the flatbed. For devices with no meaningful size, e.g., cameras, -1 is returned.

**Brightness** As Double - The brightness of the image to be acquired. The units are arbitrary as defined by the device. Devices will only support particular values for *Brightness* as indicated by the properties *BrightnessMin*, *BrightnessMax* and *BrightnessStep*. If *Brightness* is set to an unsupported value, the nearest allowable value will be used instead.

**BrightnessMin** As Double - Read-only. The minimum brightness of the image in the arbitrary units used by the device. This value is normally -1000, but some devices may use other values.

**BrightnessMax** As Double - Read-only. The maximum brightness of the image in the arbitrary units used by the device. This value is normally +1000, but some devices may use other values.

**BrightnessStep** As Double - Read-only. The step size in supported brightness values of the currently selected device.

If *BrightnessStep* has the value -1, this indicates that the step size between the minimum and maximum values is not uniform.

**Contrast** As Double - The contrast of the image to be acquired. The units are arbitrary as defined by the device. Devices will only support particular values for *Contrast* as indicated by the properties *ContrastMin*, *ContrastMax* and *ContrastStep*. If *Contrast* is set to an unsupported value, the nearest allowable value will be used instead.

**ContrastMin** As Double - Read-only. The minimum contrast of the image in the arbitrary units used by the device. This value is normally -1000, but some devices may use other values.

**ContrastMax** As Double - Read-only. The maximum contrast of the image in the arbitrary units used by the device. This value is normally +1000, but some devices may use other values.

**ContrastStep** As Double - Read-only. The step size in supported contrast values of the currently selected device.

If *ContrastStep* has the value -1, this indicates that the step size between the minimum and maximum values is not uniform.

**AutoBright** As Boolean - Setting this property to True will enable the Automatic Brightness function of the device, if available. Note that this property only has any effect with devices that support such a feature.

**AutoDeskew** As Boolean - Setting this property to True will enable the Automatic Deskew Correction feature of the device, if available. This will rotate the image received from the device (usually a scanner) to align the image correctly when the paper has not been aligned correctly. Note that this property only has any effect with devices that support such a feature.

**AutoBorder** As Boolean - Setting this property to True will enable the Automatic Border Detection feature of the device, if available. This will mean the border of the image or edge of the page is detected and the size of the acquired image set accordingly.

**Threshold** As Double - This property determines the dividing line between black pixels and white pixels for black and white scanning. It can take any value from 0 to 255, and for most devices, the default value is 128. A smaller value gives a whiter image and a larger value gives a blacker image.

With many scanners it is necessary to set the *PixelFormat* property to *ptBW* before this property can be accessed.

**FileFormat** As TxFileFormat - The file format to be used when acquiring by direct file transfer using the *AcquireToFile* command:

ffTIFF:	TIFF format.	0
ffPICT:	PICT format.	1
ffBMP:	BMP format.	2
ffXBM:	XBM format.	3
ffJPEG:	JPEG format.	4
ffFPX:	FPX format.	5
ffTIFFMult:	Multi-page TIFF format.	6
ffPNG:	PNG format.	7
ffSPIFF:	SPIFF format.	8
ffEXIF:	EXIF format.	9
ffUnknown:	No recognisable response from device.	-1

**FileFormatAllowed** (*FileType* As TxFileFormat) As Boolean - Read-only. Returns True if the currently selected device will support the specified value for the *FileFormat* property.

## Resolution in X and Y Directions

On many devices there is no distinction between resolution in the X direction (across the page) and resolution in the Y direction (down the page). The properties described above, i.e., *Resolution*, *ResolutionMin*, etc., are used to set the resolution to the same value in both directions.

If the current device supports the setting of different resolutions in the two directions, an alternative set of properties are available. These are:

XResolution / XResolutionMin / XResolutionMax / XResolutionStep, and

YResolution / YResolutionMin / YResolutionMax / YResolutionStep

These properties are used in exactly the same way as the properties already described, but allow for the resolution in the two directions to be set independently.

## Acquiring the Image

The *Acquire* method is used to read an image from the device. There are some options the user can set to define how the acquire process will be executed.

The first option is whether or not to use the default user interface provided by the device. As discussed in the preceding section, this will determine whether it is necessary to configure the device before calling the *Acquire* method. The *UseInterface* property is used to select this option. Note that some hardware devices do not allow the interface to be disabled. This behaviour can be determined by checking the *CanDisableInterface* property.

If the user interface is disabled, it is still possible to display a progress bar while the image is being acquired by setting the *ShowProgress* property.

The second option is to either wait for the *Acquire* method to complete, or to continue execution of the code in the calling application and have the completion of the image acquisition indicated by the *OnAcquire* event. This behaviour is controlled by the *WaitForAcquire* property.

## Acquiring the Image – Methods and Properties

**Acquire** ( ) - Starts the process of reading an image from the currently selected device. The exact behaviour of the *Acquire* method is determined by the following properties.

**WaitForAcquire** As Boolean - If this property is True, the calling application will stop executing when the *Acquire* method is called and the next line of code will only be executed after the acquisition of the image has been completed. If this property is False, the calling application will continue to execute and the *Acquire* method simply initiates the acquisition process. In this case, it is necessary to write an event handling procedure to finish the processing of the image (e.g., save it to file) after the *OnAcquire* event is fired. (Default = True).

**UseInterface** As Boolean - Determines whether the default user interface provided by the device will be displayed when the *Acquire* method is called. (Default = True).

**ShowProgress** As Boolean - If this property is True, and *UseInterface* is False, the device will display a progress bar during the acquisition of the image. If *UseInterface* is True, this property has no effect. (Default = False).

**AcquireToFile** (*FileName* As String) - Commands the currently selected device to acquire an image by direct file transfer. *FileName* must be a complete path to the file, including the file extension. The

file will be written to disk by the device driver and will not be available in memory. The file format is determined by the *FileFormat* property.

Important note: *AcquireToFile* is not available in the trial version.

**CanDisableInterface** As Boolean - Read-only. Some devices do not allow their user interface to be disabled, which means they will ignore the setting of *UseInterface* to False and always respond as if *UseInterface* is True. This property indicates whether or not the currently selected device behaves in this way.

**Busy** As Boolean - Read-only. Indicates that the currently selected device is busy acquiring an image, i.e. the *Acquire* method has been called to start importing the image but this process has not yet been completed.

**FileTransferSupported** As Boolean - Read-only. Indicates whether the currently selected device supports direct file transfer, i.e., use of the *AcquireToFile* function.

## Acquiring the Image – Events

In addition to the above methods and properties, events can be raised by TwainControlX during the image acquisition process.

If the *Acquire* method is used with *WaitForAcquire* set to False, it will be necessary to use these events to process the images received by the control and to determine when image acquisition is completed.

**OnAcquire** ( ) - This is raised when acquisition of an image is completed.

**OnCancel** ( ) - This is raised when the acquisition of an image is cancelled, for example by a user closing the device's user interface or clicking the Cancel button on the progress bar.

**OnFinish** ( ) - This is raised when the *Acquire* function completes execution. This can be used when multiple images are being scanned from an ADF, to signal that the last page has been read.

## Exporting the Image

When the acquisition of an image is complete, the image will be held in the memory of TwainControlX. Before this image can be used in any way it is necessary to copy it somewhere else. There are several possibilities:

- Copy it to another control to be displayed, e.g. a PictureBox control.
- Save it to a file on disk.
- Copy it to the clipboard for pasting into another application.

Methods and properties are provided to allow all these options.

## Exporting the Image – Methods and Properties

**SaveToFile** (*FileName* As String) - Saves the image as a file on disk. *FileName* must be a complete path to the file, including the file extension. The format of the saved file is determined by the file extension which must be one of the following:

.BMP	File is saved in Bitmap format.
.JPG or .JPEG	File is saved in Jpeg format.
.TIF or .TIFF	File is saved in TIFF format.
.PDF	File is saved in PDF format.

For files in Jpeg format, the compression quality will be defined by the *JPEGQuality* property. If the file already exists, it will be overwritten without warning.

**Picture** As IPictureDisp - Read-only. Provides the image data in a format that can be transferred to any object with a 'Picture' property, e.g. a PictureBox or ListImage in VB. For example, the following code copies an image from a TwainControlX object called 'Twain1' to a VB PictureBox object called 'Picture1':

```
Picture1.Picture = Twain1.Picture
```

**Copy** () - Places a copy of the acquired image onto the Windows clipboard. This can then be pasted into another application as a bitmap or as a device independent bitmap (DIB).

**BMPHandle** As Long - Read-only. Returns a Windows handle to a copy of the image that has been acquired by the control. This can be used, for example, to transfer a copy of the image to another object that uses bitmap handles. The following code example in Borland Delphi shows how the image can be assigned to an Image control for display:

```
Image1.Picture.Bitmap.Handle := Twain1.BMPHandle;
```

**JPEGQuality** As Integer - When an image is saved in Jpeg format (file extensions .jpg or .jpeg), the quality can be varied between a high quality image, which gives a large file, or a lower quality image, which gives a smaller file. This property can take a value from 1 to 100. 100 is the highest quality, 1 is very low quality. (Default = 90).

**InsertTIF** (*Source* As String, *Dest* As String, *Page* As Long) - Inserts the image as an additional page into an existing TIF file. *Source* must be a complete path to the existing file, *Dest* is a complete path to the new file to be created, and *Page* is the position in the TIF file where the image will be inserted with the first image in the file being *Page* = 1.

*Source* and *Dest* can be identical, in which case, the existing file is replaced. If *Dest* is an empty string, the existing file will be replaced. If *Source* cannot be found, *Dest* will be created new with only the single image. If *Page* is set to zero, the image will be inserted at the end of the file.

**InsertPDF** (*Source* As String, *Dest* As String, *Page* As Long) - Inserts the image as an additional page into an existing PDF file. *Source* must be a complete path to the existing file, *Dest* is a complete path to the new file to be created, and *Page* is the position in the PDF file where the image will be inserted with the first image in the file being *Page* = 1.

*Source* and *Dest* can be identical, in which case, the existing file is replaced. If *Dest* is an empty string, the existing file will be replaced. If *Source* cannot be found, *Dest* will be created new with only the single image. If *Page* is set to zero, the image will be inserted at the end of the file.

## Multiple Images

In its default mode of operation, as described in the previous sections, TwainControlX only acquires one image each time the *Acquire* method is called. By setting the *MultImage* property to True, this behaviour is changed, and the acquisition of multiple images is possible, e.g., by using a scanner with an automatic document feeder (ADF).

### Multiple Images – Methods and Properties

**MultImage** As Boolean - This is the property that must be set to True to activate multi-image mode in TwainControlX. (Default = False).

**KeepImages** As Boolean - When *KeepImages* is set to False, each image acquired replaces the previous image in memory. Each image must be saved, if required, by placing appropriate code (e.g., a *SaveToFile* command) in the event handler of the *OnAcquire* event.

Under most conditions, the time taken to scan each image is much longer than the time taken to save the image, so this approach is acceptable to capture all the images.

Alternatively, *KeepImages* can be set to True, in which case, all the images acquired are retained in memory until the *Acquire* command is completed. This makes certain that no images can be lost if the scanning is very fast, or file saving is very slow, and also gives the possibility of saving multiple image TIF files. (Default = False).

**ClearBeforeAcquire** As Boolean - By default, all images previously held in memory by the control are cleared when a new call is made to the *Acquire* function. By setting *ClearBeforeAcquire* to False, these images are retained in memory allowing the results of several *Acquire* calls to be available simultaneously. (Default = True).

**KeepInterfaceOpen** As Boolean - By default, the user interface of the device is always closed after an image has been acquired. Unless an ADF is being used, this terminates the *Acquire* command. By setting *KeepInterfaceOpen* to True, the interface remains open and can be used to acquire multiple images until closed by the user. The properties *MultImage* and *UseInterface* must both be set to True for *KeepInterfaceOpen* = True to be used. (Default = False).

**ImagesToRead** As Long - Sets the maximum number of images to acquire in multi-image mode. The *Acquire* command will stop when this number of images have been read. It may stop earlier if no more images are available, e.g., the document feeder is empty. Setting a value of zero for this property means an unlimited number of images will be read, i.e., the *Acquire* command continues until no more images are available. (Default = 0).

**ImageCount** As Long - Read-only. The number of images currently held in memory after acquiring in multi-image mode.

**PageCount** As Long - Read-only. The number of images that were acquired the last time the *Acquire* command was executed.

If *KeepImages* is True, *ImageCount* and *PageCount* should return the same value.

If *KeepImages* is False, *ImageCount* should normally be 1, whereas *PageCount* shows the number of pages that were read.

**SelectedImage** As Long - This property is used to identify the image that will be exported by the *SaveToFile*, *Picture*, *Copy* or *BMPHandle* commands when more than one image is held in memory. As an example, the following VB code could be used to save each of the images recently acquired to a series of numbered files (Image1.bmp, Image2.bmp...etc.):

```
For i = 1 To Twain1.ImageCount
    Twain1.SelectedImage = i
    Twain1.SaveToFile "C:\Image" & Str(i) & ".bmp"
Next i
```

(Default = 1).

**HasADF** As Boolean - Read-only. Indicates whether the currently selected device has an automatic document feeder (ADF).

**UseADF** As Boolean - Determines whether an ADF (if available) will be used.

**ADFLoaded** As Boolean - Read-only. Indicates whether the ADF is loaded with pages or not.

**DuplexSupported** As Long - Read-only property. Indicates whether the currently selected device supports duplex scanning. If so, it further indicates whether one-path or two-path duplex is supported. The value can be 0 (duplex unsupported), 1 (one-path duplex supported) or 2 (two-path duplex supported).

**DuplexEnabled** As Boolean - Indicates whether or not duplex scanning is enabled.

**SaveMultiPageTIF** (*FileName* As String) - Saves all the images currently held in memory to a multi-page TIF file.

**SaveMultiPagePDF** (*FileName* As String) - Saves all the images currently held in memory to a multi-page PDF file, one image per page.

It is also possible to build up a multi-page TIFF or PDF file from a number of images acquired in single image mode. This is done using the following functions:

**AddToTIF** (*Page* As Long) - Stores the image temporarily in memory as a page which is ready to be written to a TIFF file. *Page* is the number of the page in the TIFF file where the image will be saved. If *Page* is set to zero, the image will be positioned at the end of the file.

**WriteTIF** (*FileName* As String) - Saves a TIFF file to disk. The file will include all the images that have previously been stored using the *AddToTIF* function. *FileName* must be a complete path to the file.

**ClearTIF** ( ) - Clears all the images from memory that have been stored using *AddToTIF*.

**AddToPDF** (*Page* As Long) - Stores the image temporarily in memory as a page which is ready to be written to a PDF file. *Page* is the number of the page in the PDF file where the image will be saved. If *Page* is set to zero, the image will be positioned at the end of the file.

**WritePDF** (*FileName* As String) - Saves a PDF file to disk. The file will include all the images that have previously been stored using the *AddToPDF* function. *FileName* must be a complete path to the file.

**ClearPDF ( )** - Clears all the images from memory that have been stored using *AddToPDF*.

The following VB example shows how to use the above functions. Here, five images are scanned separately and then written to a PDF file with five pages:

```
Twain1.ClearPDF
Twain1.SelectDevice
For i = 1 To 5
    Twain1.Acquire
    Twain1.AddToPDF i
Next i
Twain1.WritePDF "NewFile.pdf"
```

*PDF (Portable Document Format)* is copyright of Adobe Systems Incorporated.

## Miscellaneous Functions

**AppName** As String - Some devices will display, or use in some other way, the name of the application which is calling them. For example, some scanners show a message such as "Scanning to Application XYZ..", while scanning is in progress. *AppName* allows the name of your application to be set for such use by the device. By default, this is set to 'Ciansoft TwainControlX', but you can change this to the name of your application.

**Unload ( )** - Unloads the TWAIN\_32.DLL library from memory. Under normal circumstances there is never any reason to call this function, but it can be used to reset the control if an error occurs. After calling this function, the library will be reloaded automatically as soon as any function that requires it is called.

**Clear ( )** - Clears all previously scanned images from memory.

**IsBlank** As Boolean - Indicates whether or not the current image is a blank white page. This property is typically used to identify separator pages when scanning multiple pages in a document feeder, the blank pages having been added to the batch to indicate where one multi-page file should end and the next begin. The sensitivity of this property can be adjusted using the *BlankTol* and *BlankBorder* properties.

An example VB project showing how to use this technique is available on the Ciansoft web site at <http://www.ciansoft.com/samples/tcxvbmult3.htm>.

**BlankTol** As Long - When the *IsBlank* property is used, some allowance must be made for imperfections. For example, there may be some marks on the blank pages, or specks of dust on the scanner glass that could cause some black pixels to appear in the scanned image. This property allows the tolerance to be adjusted and represents the number of non-white pixels that are permitted per 1,000,000 total pixels in the image before a page is judged not to be blank. (Default = 100).

**BlankBorder** As Double - The *IsBlank* property will ignore some pixels around the edge of the page when checking for a blank page. This allows for cases where the paper is slightly misaligned in the scanner and a black edge appears on the scanned image. This property allows the size of this border to be modified. The value is the percentage of the image height and width that will be ignored as border. For example, setting *BlankBorder* to 10% when scanning 8.5" x 11" paper will mean that a top and bottom margin each of 1.1" and a left and right margin each of 0.85" will be ignored by the *IsBlank* property. (Default = 5.0).

**MICREnabled** As Boolean - Enables Magnetic Ink Character Recognition (MICR) on scanners which support this feature. Must be set to True before calling the *Acquire* method. (Default = False).

**MICRHandle** As Long - Returns a Windows handle to the MICR data for the most recently scanned image. This must be called after *MICREnabled* has been set to True and *Acquire* is complete.

The following properties allow the property tags of PDF files to be set. They are all strings and their use is self-explanatory. All are empty strings by default.

**PDFTitle** As String - Read/Write Property.

**PDFSubject** As String - Read/Write Property.

**PDFAuthor** As String - Read/Write Property.

**PDFKeywords** As String - Read/Write Property.

## Use as a Client Side Control in a Browser

TwainControlX can be used in a browser as a client side control using Javascript (or VBScript). For users of the licensed version of the control (not the trial version), a signed CAB file is provided for this purpose. The CAB file is copied to the chosen installation directory when the installer for the licensed version is run. Licence package (.lpk) files are provided for both the trial and full versions and are also copied to the installation directory.

For more information about configuring the control to work in a browser, visit the following page on our web site:

<http://www.ciansoft.com/samples/tcxbrowser.htm>

## Deploying an Application

In order to deploy an application that uses TwainControlX you will need to distribute the OCX file, TwainControlX.ocx, together with the files that make up your application. This file will need to be registered on the machine running your application and you may wish to use a proprietary installer to do this. When TwainControlX was installed on your system our installer will have copied the OCX file to the directory "Program Files\Ciansoft\TwainControlX \" , assuming you used the installer and accepted the defaults.

The number of copies of the OCX file that may be distributed is not limited by the licence. In order to use the control in a design environment the licence file, TwainControlX.lic, is also required. A separate licensed copy of TwainControlX is required for each computer it is used on in the design environment.

## Revision History

The current version of TwainControlX is 2.4. New features (since version 2.1) include:

### New in Version 2.1

InsertTIF method.  
Brightness and associated properties.  
Contrast and associated properties.  
AutoBright property.  
AutoDeskew property.

### New in Version 2.2

Save images to PDF file format, either single or multiple pages.

### New in Version 2.3

Duplex scanning supported.  
OnFinish event.  
ClearBeforeAcquire property.  
Improved multi-page TIFF support (AddToTIF, WriteTIF, ClearTIF functions).  
Threshold property.  
MaxWidth, MaxHeight properties.  
ADFLoaded property.  
Unload function.  
Blank page detection (IsBlank property).

### New in Version 2.4

InsertPDF method.  
AcquireToFile and associated properties (FileFormat, FileFormatAllowed, FileTransferSupported).  
KeepInterfaceOpen property.