Ciansoft PDFBuilderASP User Manual

Introduction

Ciansoft PDFBuilderASP is a COM Object that enables PDF files to be created. It is primarily intended to be used on a web server running ASP or an alternative scripting language. This document contains comprehensive instructions on installing and using PDFBuilderASP.

For further information, visit our website: www.ciansoft.com.

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

Getting Started

An instance of PDFBuilderASP created in a script represents a single PDF document. In order to create and save a document, the following steps must be followed:

1. Configure document settings such as the default page size and units of measure.
2. Add pages to the document.
3. Add objects to the document that are to be written on the pages. This can include images, graphics or blocks of text. These objects are referred to as “Resources”.
4. Draw resources onto the pages.
5. Stream the file to the browser or save it to disk.

Resources are added to the document in different ways. For images, the *AddImageFile* function can be used to add an image from a file on disk. Graphics and blocks of text are created using the *CreateGraphic* or *CreateText* functions, with further functions then being used to draw the graphic or add the text block content.

All resources are allocated a unique identifying number. This is the return value of the function used to add or create the resource.

Resources are drawn on pages using the *ApplyResource* function. This function also returns a reference number, which can then be used to identify the object on the page. This reference number is used when calling functions to change the position or size of the object as it is displayed on the page.

Note that this reference number is not the same as the number that identifies resources in the document as a whole and should not be confused with it. To help clarify this issue, parameters used in functions will usually be called *Resource* where they refer to a resource identifier, *Page* where they refer to a page number and *Index* where they refer to an object on a page.

A resource can be used as many times as necessary, either on the same page, or on multiple pages. This is useful, for example, for setting up headers and footers on pages of a document, or for displaying an image full size on one page and as a thumbnail elsewhere in the document.

These instructions include the following sections:

- Installation.
- The Trial Version.
- Creating an Instance of the Object
- Managing Pages in the Document.
  Adding pages, setting page sizes, applying resources to pages and modifying the size and position of those resources as displayed on the page.
- Working with Images.
- Working with Graphics.
- Working with Text.
- Specifying Colours
- General Document Functions.
  Functions for setting the units of measure, saving PDF files, streaming to the browser, etc.

An example ASP script showing how a simple document involving images, graphics and text is created and displayed in the browser is available on our web site.

Installation

PDFBuilderASP is distributed as an archive file in ZIP format (PDFBuilderASP.zip). This archive contains the following files:

PDFBuilderASP.dll:	The component
PDFBuilderASP User Manual.pdf:	These instructions
Licence_xxx.txt:	A copy of the licence agreement

The DLL file, PDFBuilderASP.dll, must be registered on the server or other computer running the script or application. To do this, the command line utility Regsvr32.exe can be used. This is usually found in the Windows system folder and runs using the syntax:

```
regsvr32 dllname
```

where *dllname* is the path and name of the dll file to register.

The application using the component must have appropriate permissions. This means that for use in ASP, the Internet Guest User account on the server must have Read and Execute permissions on the DLL file. Appropriate permissions must also be set if the component is to be used to read or write files on the server.

As a COM object, PDFBuilderASP can be used in a range of other Windows based environments and languages, but for form based languages such as Visual Basic or Delphi, we recommend using its sister product, the ActiveX control PDFBuilderX instead.

The Trial Version

The trial version of PDFBuilderASP is supplied as a different DLL file, called PDFBuilderASPTrial.dll. The trial version has all the functionality of the full version of the control. The only limitation is that each page of PDF files created using the control will have a line of text written on it indicating that trial software was used. Visit www.ciansoft.com to purchase the full version.

Creating an Instance of the Object

Before PDFBuilderASP can be used in an ASP script an instance of the object must be created using the Server.CreateObject command. The syntax for this is:

```
Set objPDF = Server.CreateObject("PDFBuilderASP.PDFSvrDoc")
```

Where objPDF is a variable name. Any variable name can be used, but we usually use the variable name “objPDF” in code examples in these instructions and elsewhere on our web site.

For the trial version, the syntax is:

```
Set objPDF = Server.CreateObject("PDFBuilderASPTrial.PDFSvrDoc")
```

Managing Pages in the Document

The following functions are used to add pages to the document, to change the page sizes and to adjust the position and size of objects on a page.

AddPage *Page* - Adds a new page to the document. The position of the page in the document is defined by *Page*, which must be an Integer. If *Page* is 0, the page will be added at the end of the document. The page size is initially defined by the *DefaultPageSize* property, but can be modified.

PageSize *Page* - Integer. The size of the page referenced by *Page*. The possible values are listed below.

0	Page size is defined by the PageWidth and PageHeight properties
1	A4 Portrait (210 mm x 297 mm)
2	A4 Landscape (297 mm x 210 mm)
3	Letter Portrait (8.5” x 11”)
4	Letter Landscape (11” x 8.5”)
5	A3 Portrait (297 mm x 420 mm)
6	A3 Landscape (420 mm x 297 mm)
7	A5 Portrait (148.5 mm x 210 mm)
8	A5 Landscape (210 mm x 148.5 mm)
9	Tabloid Portrait (11” x 17”)
10	Tabloid Landscape (17” x 11”)
11	Legal Portrait (8.5” x 14”)
12	Legal Landscape (14” x 8.5”)
13	Statement Portrait (5.5” x 8.5”)
14	Statement Landscape (8.5” x 5.5”)
15	Executive Portrait (7.25” x 10.5”)
16	Executive Landscape (10.5” x 7.25”)

PageWidth *Page* - Real. The width of the page referenced by *Page* in the units of measure currently set by the property *Units*.

PageHeight *Page* - Real. The height of the page referenced by *Page* in the units of measure currently set by the property *Units*.

DefaultPageSize - Integer. The page size that will be used for new pages as they are added to the document. (Default = 1, A4 Portrait).

ApplyResource *Page*, *Resource* - Integer. Applies the resource referenced by *Resource* to the page referenced by *Page*, both of which are Integers. The resource is applied using default sizing and positioning which can then be modified using the *Locate* or *ScaleObject* functions. The return value of this function is an Integer which indicates the index number of this specific resource on this specific page.

Locate *Page*, *Index*, *X*, *Y* - The object referenced by *Index* on the page referenced by *Page* will be positioned so that its top-left corner will be at the co-ordinates *X*, *Y* measured from the bottom-left corner of the page. *Page* and *Index* are Integers; *X* and *Y* are Real.

ScaleObject *Page*, *Index*, *ScaleFactor* - The object referenced by *Index* on the page referenced by *Page* will be scaled. *ScaleFactor* is a percentage value, so an object will be displayed at normal size if this value is 100. For example, to display an image as a thumbnail, one eighth of its normal size, use a *ScaleFactor* of 12.5, i.e., 100/8. *Page* and *Index* are Integers; *ScaleFactor* is Real.

Working with Images

Images can be added to the document as resources either by reading image files in a supported format from disk or by copying a bitmap in memory as a bitmap handle.

AddImageFile *FileName* - Integer. Adds an image from a file on disk as a resource in the document. The return value of the function is the resource identifying number. Images in the following file formats can be used: .bmp, .tif, .jpg, .png, .gif, .pcx, .psd, .wbmp. *FileName* is a String and must be a complete path to the file including the file extension.

AddImageBMPHandle *Handle* - Integer. Adds an image referenced by a bitmap handle. This can be used to transfer an image to PDFBuilderASP directly from another component used for processing images, without the need to save the image to disk and read it back into memory. The return value of the function is the resource identifying number.

ReleaseBMPHandle - Boolean. This property is used to determine whether the handle is released to its original owner when the *AddImageBMPHandle* function is used. If True, then the original owner is responsible for clearing the image from memory when it is no longer needed. (Default = True).

ImageReadNumber - Integer. When adding an image resource from a TIFF file containing multiple images, *ImageReadNumber* indicates the number of the image in the file that is to be read.
(Default = 1).

ImageWidth *Resource* - Real, read-only property. The width of the image resource referenced by *Resource* in the units of measure currently set by the property *Units*. *Resource* is an Integer.

ImageHeight *Resource* - Real, read-only property. The height of the image resource referenced by *Resource* in the units of measure currently set by the property *Units*. *Resource* is an Integer.

For most purposes, it is not necessary for the user of PDFBuilderASP to be concerned about the compression algorithms used for storing images. The default behaviour of the control is appropriate for most situations.

Images stored in a PDF document will be either uncompressed or compressed using one of the following methods:

0	Uncompressed.
1	CCITT Group4 compression. Used only for black and white images.
2	ZIP (or Flate) compression.
3	JPEG compression.

Group4 and ZIP are lossless compression methods which means that the full quality of the image is retained whilst the space occupied by the image on disk is reduced. JPEG compression is a lossy method which sacrifices some image quality for a significant reduction in size and is commonly used for full colour or greyscale photographic images.

The compression method can be set for each possible image type independently using the following properties. Each is an Integer taking a value from 0 to 3 as defined in the above table.

CompressionBW - Integer. The compression method to be used for storing black and white images in the document. (Default = 1, CCITT Group4).

CompressionGray - Integer. The compression method to be used for storing greyscale images in the document. (Default = 2, ZIP).

CompressionIndexed - Integer. The compression method to be used for storing indexed (paletted) colour images in the document. (Default = 2, ZIP).

CompressionRGB - Integer. The compression method to be used for storing full colour images in the document. (Default = 2, ZIP).

UseSourceCompression - Boolean. If this is set to True, the image compression method used in the source file will be retained if possible. This is used to avoid unnecessary decoding of images to satisfy the settings of one of the other compression properties when the image is already compressed using an efficient method. (Default = True).

Working with Graphics

Graphics are drawings made up of combinations of shapes and lines. When drawing a graphic as a resource, the size of the page on which the graphic will eventually be displayed should be kept in mind. The graphic functions require that the co-ordinates of the shapes and lines on the page be specified during drawing and these co-ordinates cannot be changed using the *Locate* function when the graphic resource is later applied to a page. The graphic also cannot be resized using the *ScaleObject* function.

Typical uses for graphics in a document are the drawing of borders and tables.

CreateGraphic - Integer. Creates a new graphic resource. The return value of the function is the resource identifying number. The value of *CurrentGraphic* will automatically be set to this value.

CurrentGraphic - Integer. The reference number of the graphic resource that is currently in use. This must be set before any drawing is done on the graphic by using the *DrawLine*, *Rectangle* functions etc.

DrawLine *X1*, *Y1*, *X2*, *Y2* - Draws a line on the current graphic using the current line settings (*LineWidth*, *LineColor* etc.) from co-ordinates *X1*, *Y1* to *X2*, *Y2* which are all Real numbers.

Rectangle *X1*, *Y1*, *X2*, *Y2* - Draws a rectangle on the current graphic using the current line settings (*LineWidth*, *LineColor* etc.) with opposite corners at co-ordinates *X1*, *Y1* and *X2*, *Y2* which are all Real numbers.

LineColor - The colour to be used on the current graphic for drawing lines. See later section on ‘Specifying Colours’ for explanation of how to define colours in ASP. (Default = Black).

LineWidth - Integer. The thickness of lines drawn on the current graphic. (Default = 1).

FillColor - The colour to be used on the current graphic for filling shapes. See later section on ‘Specifying Colours’ for explanation of how to define colours in ASP. (Default = White).

Working with Text

Blocks of text can be added to the document as resources. Each block consists of any number of single lines of text written underneath each other. Within each block of text, a single font and font size must be used.

CreateText - Integer. Creates a new text resource. The return value of the function is the resource identifying number. The value of *CurrentText* will automatically be set to this value.

CurrentText - Integer. The reference number of the text resource that is currently in use. This must be set before writing any text on the graphic, or modifying settings such as *TextColor* or *TextFont*.

WriteText ***Text*** - Adds a single line of text to the current text resource. The text will be written immediately below the last line of text to be written. The first line of text added to the resource will be written at the position on the page defined by a call to the *Locate* function. *Text* is a String.

TextColor - The colour to be used on the current text block for text. See later section on 'Specifying Colours' for explanation of how to define colours in ASP. (Default = Black).

TextFont - Integer. Text can be written in any one of the 14 standard Type 1 fonts listed below. The TextFont property takes a value from 1 to 14 indicating the font to be used for the current text resource.

1	Courier
2	Courier-Bold
3	Courier-BoldOblique
4	Courier-Oblique
5 (Default)	Helvetica
6	Helvetica-Bold
7	Helvetica-BoldOblique
8	Helvetica-Oblique
9	Times-Roman
10	Times-Bold
11	Times-Italic
12	Times-BoldItalic
13	Symbol
14	ZapfDingbats

TextSize - Integer. The size of the text for the current text block, in points. If a block of text is scaled using the *ScaleObject* after it is applied to a page, the size of the text will also be scaled accordingly. (Default = 10).

Specifying Colours

The colour properties (*LineColor*, *FillColor*, *TextColor*) can be defined either in hexadecimal format or by using the predefined VB colour constants.

Hexadecimal colours are written in the format &HBBGGRR& where BB, GG and RR are the values for blue, green and red.

Examples:

Set FillColor to black (red, green and blue are all zero)

```
objPDF.FillColor = &H000000&
```

or

```
objPDF.FillColor = vbBlack
```

Set LineColor to yellow (mixture of green and red)

```
objPDF.LineColor = &H00FFFF&
```

or

```
objPDF.FillColor = vbYellow
```

General Document Functions

General functions operating at a document level.

Clear - Deletes all pages and all resources from the control, allowing a new document to be started.

DeleteAllPages - Deletes all pages from the control, but retains all resources for use in a new document.

SaveToFile *FileName* - Saves the document to disk in PDF format using *FileName* which is a String.

StreamData - Variant. Returns a copy of the PDF document in binary format for streaming to the browser. Note that in ASP, the *BinaryWrite* command provides a more convenient solution for streaming to the browser in a single command. It is only necessary to use *StreamData* if the HTTP header information needs to be customised in some way.

BinaryWrite - Streams the contents of the current PDF document to the browser by invoking a Response.BinaryWrite command in ASP. This command also sends appropriate header information and is equivalent to the following lines of ASP code:

```
Response.ContentType = "application/pdf"
Response.AddHeader "Content-Disposition", _
 "inline; filename=StreamFileName"
[  or, depending on value of the StreamInline property,
   Response.AddHeader "Content-Disposition", _
    "attachment; filename=StreamFileName" ]
Response.AddHeader "Content-Length", "Length of the StreamData"
Response.BinaryWrite objPDF.StreamData
```

StreamFileName - String. The file name that will be used in the “Content-Disposition” line of the HTTP header when the *BinaryWrite* command is called. (Default = empty string).

StreamInline - Boolean. If set to True, the “Content-Disposition” line of the HTTP header when the *BinaryWrite* command is called will specify “inline”, otherwise it will specify “attachment”.
(Default = True).

Units - Integer. The units of measure to be used for sizing and locating all pages and objects in the current document. It is recommended that this property be set to the preferred value before any pages or resources are added to the document, and not subsequently changed. The results can be confusing if resources created using one value for *Units* are then drawn onto pages using a different value.

Units can take any of the following values:

0 (Default)	A point is 1/72 of an inch.
1	Inches. 1 inch = 72 points.
2	Centimeters. 1 cm = 28.3465 points
3	Millimeters. 1 mm = 2.83465 points

All PDF documents generated by PDFBuilderASP use points as the internal unit of measure. Nevertheless, any of the above options can be used in your application as all conversions will be made automatically.

All co-ordinates used in the *Locate* function and graphic drawing functions are based on *Units* and are measured from the bottom-left corner of each page.

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