




aspEasyPDF

Last modified : 30/01/2006 (Version 3.30)

What's aspEasyPDF ?

aspEasyPDF is a library for developers that integrates on projects the ability to **create PDF documents in real-time**. It has been optimized for use in a Web development environment using standard languages like ASP, ASP.NET, Visual Basic, Delphi and all other languages that accept calls to external libraries or using COM+ objects.

What is revolutionary of this library is that integrates a visual document design form that creates the code for you, this will help you to start using the library on your projects without reading manuals and understand which functions to use. [VisualEasyPDF](#)  will type the code for you.



The library supports the most important PDF features, text in any place and format, drawings, graphics, fonts, aligns, bookmarks, compression, security (40 & 128 bits), forms, javascript, import PDFs and more. It's very fast, compared to others that use virtual printer captures. Full document and with examples in any programming environment.

The library is **Shareware** which has one limitation; it displays a message on each page of the PDF document, the **Crippleware** technique is used to guaranty piracy protection. **The price? Starts at just 89USD\$ per machine!**

If you need more features there is a **Professional** version which enables you to read HTML and render directly to PDF, security encryption, JavaScript, form fields, barcodes, URL links, Jumps and True Type fonts. [See this table to see the difference between standard a Pro version.](#)

[See the prices and conditions of the aspEasyPDF family.](#)

Undecided to buy this component!, then [see the following page](#)  to see how many clients are just working with our components.

If you want to use it on your site but you don't find the time to do it your self, please contact a [developer](#)  that will be happy to do it for a small amount of money. [See a list of resellers and web developers](#) 



email: support@mitdata.com

http : www.mitdata.com

Phone: +34.93.7547075

Fax: +34.93.7548065

Feature Matrix

aspEasyPDF

Last modified : 30/01/2006

Feature matrix between different versions

Booths uses the same technologies and are very easy to use.



When you evaluate the component you get all the features to test it, but when ordering you must decide which version suits best to you.

The **Standard version** can do practically all things you want to use on normal works. Draw text, use 14 fonts, alignments, insert pictures, control pages and text outputs and many more features, read the docs.

The **Professional version** is done for the developers / enterprises that want to go beyond the standard features, like doing bar codes inside the document, developing in a faster time when using the html features (you reduce the ASP code from the standard version in a 80% off), use more fonts than the 14 standards from PDF , and more features that will came in next version.

Also an important price feature is that you get all **major releases for free**, the standard version only gets minor releases, for majors one you must pays low upgrade price.

The **Enterprise version** adds Professional version and also adds the easyReportPDF library inside the aspEasyPDF, so you will only use one library to create all PDF documents and reports.

Feature	Standard	 Professional	 Enterprise
Fast technologies and easy to use	Yes	Yes	Yes
Page properties and size	Yes	Yes	Yes
Text properties (alignments)	Yes	Yes	Yes
Font properties (size, bold, etc)	Yes	Yes	Yes
Write text like adding paragraphs	Yes	Yes	Yes
Write text at any position inside the page	Yes	Yes	Yes
Write text using html syntax	No	Yes	Yes
Loads HTML sites	No	Yes	Yes
Loads HTML sites from secured site (Basic, NTLM, SSL)	No	No	Yes
Draw graphics at any position inside the page (lines, boxes, ellipses, shadows)	Yes	Yes	Yes
Support for Graphics in BMP, PNG, TIFF, GIF and JPEG	Yes	Yes	Yes
Chart capabilities for one series	Yes	Yes	Yes
Chart capabilities for multi-series	No	Yes	Yes
Import PDF	No	Yes	Yes
Draw Code Bars inside the document (EAN, Code39, Code128, etc)	No	Yes	Yes
Multidimensional PDF417 barcodes	No	Yes	Yes
Set URLs links inside the document	No	Yes	Yes
CJK Adobe standard fonts with Unicode	Yes	Yes	Yes
Use external fonts like True Type	No	Yes	Yes
True Type Fonts with Unicode support	No	Yes	Yes
Compression page	No	Yes	Yes
Move dynamically into any created pages	No	Yes	Yes
Load VisualEasyPDF form files, draw it and modify objects	Yes	Yes	Yes
Password PDF protection	No	Yes	Yes

Security definition 40 bits and 128 bits encryption (print, modify, save...)	No	Yes	Yes
Forms (Buttons, Text Labels, List Box, Combo Box...)	No	Yes	Yes
JavaScript for event objects	No	Yes	Yes
Bookmarks / Outlines	No	Yes	Yes
EasyReportPDF - Reports render (ERP)	No	No	Yes
ERP - Connection to ADO databases	No	No	Yes
ERP - Connection to native MySQL Servers	No	No	Yes
ERP - Connection to native msSQL Servers	No	No	Yes
ERP - Connection to native Oracle Servers	No	No	Yes
ERP - Connection to other native drivers	No	No	Yes
ERP - Basic Scripts in reports	No	No	Yes



" Professional features "

Those are the functions that can not be used on the standard version and that are only for Professional version:

- [AddBarCode](#)
- [AddFont](#) (only TTF fonts)
- [AddFormObj](#)
- [AddFDFValue](#)
- [AddEventObj](#)
- [AddHTML](#) [AddHTMLPos](#)
- [AddLink](#) [AddLinkPos](#)
- [AddNote](#)
- [AddOutline](#)
- [AddPattern](#)
- [AddPDF](#)
- [AddProtection](#)
- [MoveToPage](#)
- [Advanced functions](#)



" Enterprise version features "

Those are the functions that can not be used on the standard or Professional version and that are only for the Enterprise version:

(note, in Enterprise version you can use all Professional features)

- [AddDBBand](#)
- [AddDBBarCode](#)
- [AddDBConnection](#)
- [AddDBText](#)
- [AddDBGraphic](#)
- [AddParameter](#)
- [AddScript](#)
- [AddTextField](#)
- [RenderReport](#)
- [SetDBBandActive](#)
- [SetDBConnection](#)
- [SetDBSQL](#)
- [SetReportPage](#)
- [SetParameter](#)

Those are the functions that can not be used on the standard version and that are only for Professional version:

HTML Commands

aspEasyPDF Pro & Ent

Last modified : 30/01/2006

HTML Commands that supports the aspEasyPDF Professional version.

The HTML support on the aspEasyPDF Pro is for an easy, common and faster way to develop PDF documents, you can reduce the ASP code in a 60% to 80% and make it faster as there are single sentences that you use on the ASP and all the work is transferred to the component. Some parts of the component as been done in pure assembler to do it faster than using scripting technologies.

As the intention is to help to reduce the coded size and for faster developing, don't expect to get the same output as you see on the Internet Explorer or Netscape. Is so difficult to make it look on the same way, think about Opera and Internet Explorer, the rendering are quite the same but you see a lot of differences when loading pages on it, different standards on the Html, supports different html versions, and some companies adds a non html standard syntax.

Our intention is to develop a product that will increase on functionality and make a closer look as you see on the Internet Explorer or Netscape.

Those are the html tags that are fully supported:

BODY - Sets the background color if specified

TITLE - Sets the title of the document

H1 to H6- Headers

HR- Lines (Size and Color options)

BR - **BR/** - Branch Line

U - Underline tag

I - **EM**- Italic font

B - Bold Tag, also the **STRONG** is supported

FONT - To specify a different font to be used, it accepts the **FACE**, **SIZE** and color parameters . For fixed font sizes you can use negative values or using **CSS**.

`` and `` are the same size font... to get a font size lower than 8 then use negative values. (*from version 2.22*)

P - Paragraph, accepts the **ALIGN** parameter (Left, Right and Center)

SMALL - Small font

BLOCKQUOTE- Add consecutive quotes

SUB - **SUP**

UL - **OL**- **LI** - Bullets (You can use **START** and **TYPE** parameters)

CENTER - To center text

A - Adds a Link. Supports outside link, file link () and inside link #.

IMG - Adds an image. Supports outside link http and file link ().

TABLE - Supports tables. It does not support the rowspan. All other properties are supported.

(Note: Tables structure must be completely written in a full variable to the AddHTML, you can not call several times the AddHTML to draw one table)

SCRIPT - All information from the script will be ignored.

Basic CSS support (font-family, font-weight, font-size, color, page-break-after) using referenced CLASS tag or direct STYLE tag

Full support to the [ISO Latin-1 Character](#) Set **entities** and you can use decimal entities (&#xx;)

What tags are not supported:

- Rowspan on tables.
- CSS, Advanced Style sheet (Basic is supported) .

Additional tags not standard from HTML but useful for the aspEasyPDF:

<!-PAGE BREAK> and **<!-PDF.ADDPAGE>**

Forces to add a new page.

<!-PDF.MOVETOPAGE>

Moves to a specific page, use the **Value** to set the page to jump.

F.E. to jump on second page: `<!-PDF.MOVETOPAGE Value="2">`

<!-PDF.ADDBARCODE>

Draws a Barcode. You have parameters for **X, Y, Height, Type** and **Value**. If you don't set the Cursor position for X, Y then it will use the actual position.

Draw a barcode: `<!-PDF.ADDBARCODE X=100 Y=200 Height=30 Type=2 Value=123>`

<!-PDF.ADDTEXTPOS>

Powerful function to add text anywhere. Parameters **X, Y** for cursor and **Value** for the text to display.

Example: `<-PDF.ADDTEXTPOS X=400 Y=400 Value="Hello world">`

<!-PDF.SETPROPERTY>

Alters an internal property of the aspEasyPDF. See instructions on how to use the [SetProperty](#) . Parameters are **ID** and **Value**.

Example that sets the author document to MITData: `<!-PDF.SETPROPERTY ID=302 Value="MITData, S.C.P.">`

<!-PDF.PAGENUMBER>

Returns the actual page number

<!-PDF.PAGECOUNT>

Returns the total pages added when using this tag. If you add more pages after using this tag then it will be no updated. This tag is to be used with the [AddPattern](#) function

Non standard CSS supported by aspEasyPDF:

Syntax :	Description:
{ adjust-width : no auto (default) }	Tables will adjust it's size by it's contents to make it see proportional and improve the page contents. If you don't want to use the improvements set the value to no.

Some examples to see how it works:

```
<%  
' Create the component  
set PDF = server.createObject("aspPDF.EasyPDF")  
' Set the actual font, indeed you can also set it by HTML tags  
PDF.SetFont " F1 ", 12, "# 000000 "  
PDF.AddHTML " <p>The html command are implemented to allow a fast and easy use to draw on <b>PDF</b>. " & _  
" It has not been done to make an <u>exact view</u> as you see on the Internet Explorer</p> "  
'PDF.AddHTML " <p><font face=""F1"" size=""2"">This only a demo to see how it works.</font><br>HTML Commands  
supported:</p> "  
PDF.AddHTML " <p>Only paragraphs, bold, italic, color and a bit more, this is only the <u>beginning</u> as this  
demonstrates " &_  
" html commands that are supported, now until the final release It will have the most important commands</p> "  
PDF.AddHTML " <p align=""center"">ENJOY!</p> "  
PDF.AddHTML " <p><a href=""www.mitdata.com"">Visit our page</a></p> "  
PDF.BinaryWrite  
' destroy it  
set pdf = nothing  
>%>
```

Other easy functions:

```
<%  
' Create the component  
set PDF = server.createObject("aspPDF.EasyPDF")  
' Adds html from an URL, it also enables you to run ASP pages dynamically  
PDF.AddHTML "http://www.mitdata.com/AspEasy/pdf_html_Support.htm"  
PDF.BinaryWrite  
' destroy it  
set pdf = nothing  
>%>
```

And now direct from a server file:


```
<%  
' Create the component  
set PDF = server.createObject("aspPDF.EasyPDF")
```


```
' Adds html from a File  
PDF.AddHTML "file:///C:/inetpub/AspEasy/pdf_html_Support.htm"  
PDF.Save "c:\inetpub\inetpub\aspEasyPDF.pdf"  
' destroy it  
set pdf = nothing  
%>
```

License overview

aspEasyPDF

Quick help about using the license on aspEasyPDF

aspEasyPDF uses Crippleware technique to guaranty from being hacked; so shareware version and registered version are different. So make sure that after registering your copy, you uninstall the shareware version and install the registered version previously downloaded from the registered area. 

Once you have installed the registered version you should copy the license file that was emailed with your order, if not you can request a new license file from the registered area site. 

Note: The license codifies the TCP/IP address of the machine that will be used for running the aspEasyPDF, if it does not have an unique IP address then you should use the Unique ID identification. To know which is the Unique ID of your machine that will use aspEasyPDF you can use the [PDF.Version](#) property to know which is or the aspEasyREG tool which will give you also the same information.

You should copy the license to the system32 path, which is the default path that aspEasyPDF will look for, or at the same place which was installed aspEasyPDF.

Remember to give read permissions to the anonymous IIS user for reading the license, if not, you will get a message that the easypdf.lic is not found.

If you need to store the license in a different place, (web host providers uses home users directory for each user), then you should load the license with the [PDF.License](#) function

Note: Site License users may use the [PDF.SiteLicense](#) property to unlock the site license version. If you have a special compilation then you don't need to use any license function.

Prices & Conditions

All products when registered have the following features:

- " Free email support for all of our products
- " Minor upgrades and bug fixes
- " Major upgrades, see table to see which products are under those conditions
- " Notification by email when newer versions are released

Prices and conditions

To order one product go to <http://www.mitdata.net/online>
 We process orders in 5 minutes with Visa Credit card

Product	Code to Order it	Price per/ Server	Site License	Full Mail Support	Free Updates for minor releases	Free Updates for major releases	Special Prices for major releases	Bug fixes e-mail notifications
aspEasyPDF	38546	89 US\$	400 US\$	Yes	Yes	No	Yes	Yes
aspEasyPDF Pro	44318	189 US\$	9 99 US\$	Yes	Yes	Yes	N/A	Yes
aspEasyPDF Enterprise	44970	399 US\$	1.9 99 US\$	Yes	Yes	Yes	N/A	Yes
easyReportPDF	54970	299 US\$	1.4 99 US\$	Yes	Yes	Yes	N/A	Yes
VisualEasyPDF	58293	5 9 US\$		Yes	Yes	Yes	N/A	Yes

N/A = Not applicable

[Email us](#) for special conditions and quantities prices. For price differences between versions, contact us to get more information.
 Schools and Universities (Educational Services) have 35% off from regular price

What's per Server License

Allows you to run the program on different computers depending on number of licenses purchased. Each license is for each computer that runs the program.

What's Site License

SITE licensing option helps you achieve significant savings on volume purchase of MITData products. SITE license entitles you to install and use the software on unlimited number of workstations.

History

aspEasyPDF

Last modified : 30/01/2006

3.30 (30-January-2006)

- " AddHTML - Caption tag added with no parameters at all
- " AddHTML - Styles for TABLE and TD with WIDTH property
- " AddHTML - Styles with % values were not correctly calculated
- " AddHTML - After using the AddGraphicPos is always adding a border
- " AddPDF - Referred images are correctly imported
- " AddPDF - Colorspace without external object was getting an invalid handler
- " AddPDF - Added new compression type for graphics palette and structure
- " AddPDF - Added DeviceCMYK for graphics using a different color palette than RGB
- " AddPDF - Partial Fonts were not correctly loaded.
- " AddPDF - Improved for PDF created in UNIX systems
- " AddPDF - Now manages different fonts for different pages in the same PDF with the same name.
- " AddPDF - Can now load from HTTP and HTTPS sites
- " ENTERP - 1.05 and 1.06 version added
- " ENTERP - SSL and Authentic method validation for all http request
- " New property constant for ignoring HTTP errors
- " On JS Events the \n was translated to a wrong code format

3.21 (29-September-2005)

- " csHTML_TRFullPage was not working with big images in HTML tables.
- " Table across pages with images were adding an extra page.
- " Footer Patterns with big images that didn't fit was causing loop bug error crash.
- " Special characters (,) and \ are ignored in form captions and in the AddTextWidth function
- " AddHTML - Added absolute font size for the size property in font (ex. size=8px)
- " AddHTML - Decrement font size was adding instead of subtracting
- " AddPDF - Removed the font optimization because if you were adding multiple PDF you may get unreadable text
- " AddPDF - Setting the csPageAutoAdd to false will not insert a new first page when loading a PDF and sees that there are contents on that page.
- " ENTERP - Added properties to control an automatic Grow for a field
- " ENTERP - Graphics was using it's original path instead of using the VEP memory structure.
- " ENTERP - Shapes with the same color background of page was drawing it in black.
- " ENTERP - Objects can be placed in a relative position that will print the object correctly for a grow field
- " ENTERP - Database Fields with HTML contents can be printed on the document
- " ENTERP - Functions can now return Variant types which

3.20 (09-September-2005)

- " New Enterprise version which adds the reports management
- " New error management in LastError message property for all functions
- " LoadPDF - extGStates ignores unsupported command which makes a readable document
- " LoadPDF - extGStates with object referred name is correctly imported
- " LoadPDF - Removed a bug in ColorSpace which would not read correctly the item information
- " AddHTML - Wrong Width calculation if one cell got a width specification with a colspan set
- " AddHTML - H1 H2 H3 H4 H5 H6 font size tags were ignored
- " AddHTML - Support incremental o decrement specification on font Size property. Ej: and (-)x ; Increments or decrements actual font.

3.15 (22-July-2005)

- " Corrected the image format PNG for 2 bits which was inverted
- " Row size height was not correct
- " When loading existent PDF now uses correctly the flat encoding for type1 fonts
- " Http requests on image tags are not more case sensitive
- " You can set a rotate flag for every page. csPageRotate
- " Loading PDF uses the correct rotate property for every page set.
- " VEP object can change Fontname and Fontunderline dynamically

3.14 (28-June-2005)

- " Improved the width calculations for tables when big phrases were in cell without a specific size
- " Images in Cells right alignment was wrong
- " Added ™ entity
- " Pattern page parameters were not correctly used

3.13 (13-June-2005)

- " Wrong image import for PNG
- " Right aligned patterns was adding an additional line when using internal commands.

3.12 (02-June-2005)

- " addHTML has a better resize width for small columns
- " Small images in html paragraphs was not correctly positioned the next paragraph
- " Sometimes the csHTML_TRFullPage constant may let 1 row in a single page, now it's fixed.
- " Text rendering has being improved for big paragraphs

3.11 (30-May-2005)

- " Shareware version got an exception when using the security settings in aspEasyPDF

3.1 (28-May-2005)

- " New Tables management for standard and PRO version
- " JPG Improvement
- " HTML improved Cell alignment
- " Removed a bug that may get out the contents of an html cell if it was center alignment and it was not too much size in the column
- " Added height property to the table.
- " Bug removed when saving twice the document
- " New csPropIntHTTPQuery constant to control HTTP request (0 default for GET and 1 for POST)
- " Added the new csPropIntDebugView constant for kernel debugging with aspEasyPDF
- " AddPDF fixes a bug when loading internal Acrobat fonts, manage a better compatibility with 3rd party tools
- " AddPDF works with embedded fonts and compressed
- " AddPDF handles multiple documents with the same referenced font
- " AddPDF can load ascii85decode compressed documents.
- " AddPDF manages correctly the first insertion page in pages without contents and images.

3.0 (06-March-2005)

- " Charts capabilities in combination of VisualEasyPDF
- " SetFont with 0 size was corrupting the document
- " FDF forms fails when the content value is null
- " You can generate multiple documents with different contents with the same instance
- " Euro entity was wrong coded
- " Barcodes were all inverted
- " Encryption and compress goes faster
- " Interpolate constant for images; csPropGraphInterpolate
- " Setting and align property and then a valign, was completely ignoring the align property
- " csPropGraphPDF417_Mode, csPropGraphPDF417_SecurityLevel,
- " csPropIntHTTPTimeOut
- " Cell space was wrong when it didn't fit.
- " Fixed Cell were scaled when the text was two lines height.
- " HTML bold and underline ending tag in certain condition added a white space.
- " Indexed images on the same TIFF file was not drawing it by a fail on the cache name
- " PRO: csHTML_FontSize property was being lost after a paragraph tag was found
- " PRO: Fonts with styles could corrupt the PDF document by a bad pointer allocation
- " Added Angle property to the graphic images
- " Stress version included on just one binary compilation

2.24 (10-Agust-2004)

- " PRO: Centered text was not right calculated on nested tables
- " IP license was not being checked on the registered version

2.23 (30-July-2004)

- " PRO: Multiple paragraph format while centered, keeps a correct formatting Text.
- " PRO: addPDF imports ColorSpaces from body contexts. Improved fonts reading.
- " PRO: FDF values can be without forms notation (forms.name now accepts only name)
- " Bug in the AddText function when using a branch return tag at the end on the line.

2.22 (09-July-2004)

- " PRO: Nested table with row valign or cell valign was not correctly getting the total height
- " PRO: Using internal instructions on HTML tables were not being used for cell size calculations.
- " PRO: Improved white/blank removal chars on HTML documents
- " PRO: Images with a specific size property was reserving a bigger space for a cell than it was.
- " PRO: Better auto adjust table width when there is no width parameter specification for the table.
- " PRO: Setting the table border to 1 and all cells to 0 was not drawing correctly the border of the table.
- " PRO: Added a new CSS property to control the width calculation of the table; adjust-width
- " PRO: A TD tag without a TR tag specified first was interrupting the execution
- " PRO: Barcodes now doesn't have the transparent flag set.
- " PRO: AddPDF imports only known images, if there is a problem the skips the image and it displays correctly the document
- " PRO: AddPDF added support for small embedded fonts
- " PRO: Last styles values in style tag was ignored if there wasn't end by a ;
- " CharSpacing in AddText and AddTextWidth function
- " was not adding a blank space on the calculation logic for a correct alignment.

2.21 (05-May-2004)

- " PRO: True Type font with WinAnsi type was not correctly displayed
- " PRO: AddPDF with ColorSpace inside the contents is ignored
- " PRO: AddPDF fonts without name are being loaded
- " PRO: AddPDF was not loading second pages
- " AddTextPos can now use branch tags (
)
- " Euro font is missing
- " Unicode wrong font format

2.20 (30-April-2004)

- " PRO: HTML added Margins tag
- " PRO: HTML colors without # character are processed
- " PRO: Nested centered tables were ignored
- " PRO: AddPDF Imports TrueType fonts and Type1 fonts
- " PRO: AddPDF Imports correctly Annotations types.
- " PRO: script inside tables were giving a wrong size in cells.
- " PRO: <IL> tag now aligns correctly if number gets higher than 9 and uses correct font size for the paragraph
- " PRO: <!-PDF.SETPROPERTY tag was ignored
- " PRO: Security was giving random fails by a bad pointer address
- " PRO: Basic CSS support (font-family font-weight font-size color page-break-after) using referenced CLASS tag or direct STYLE tag
- " PRO: csPropIntProxyUser, csPropIntProxyPass - New properties for setting HTTP basic authentication
- " csPropIntDebugTCP, csPropIntDebugLevel - New properties for debugging
- " Word spacing, render mode is stored for pages, this makes consistent changes when moving trough pages and restoring default drawing modes per page.
- " UnicodeCheck is set to false by default. Some countries like Germany was adding it automatically when there was no need.

2.11 (23-March-2004)

- " PRO: AddPDF was not correctly loading multiple fonts
- " PRO: AddPDF didn't get the image in the correct format mask
- " PRO: AddPDF supports compression
- " PRO: AddPDF supports linerazed PDFs
- " PRO: AddPDF supports Unix coded PDFs
- " PRO: [AddPDF](#) loads unknown graphics formats, like Fax TIFF
- " PRO: Entities coded in decimal or hexadecimal were ignored if using a WinAnsi font.
- " PRO: AddFont adds correctly win98 external fonts
- " PRO: Complex HTML with graphics was adding a blank new page at the final document
- " PRO: HTML tables with a page body background color different was redering it in blank
- " AddFonts performs a better error check and set the LastError information

2.10 (12-March-2004)

- " PRO: Big tables across multiple pages were losing primary width information
- " PRO: [AddPDF](#) ; new function to import native PDF documents
- " PRO: VAlign for cells is supported
- " PRO: TR accepts alignment default for cells
- " PRO: Better support for centered contents with different format styles
- " PRO: Rewritten some classes to allow read access to existent PDF files (item under work)
- " PRO: Embedding fonts with added security PDF Reader was displaying a warning message and didn't use the embedded font
- " PRO: Embedded fonts was not removing a memory pointer
- " PRO: Hexadecimal is working for unicode
- " PRO: HR tag was not correctly rendered if used twice between a simple text
- " PRO: AutoAddPage was not working when the option was set to false and adding html tables
- " PRO: Added support for TTF unicode fonts
- " PRO: Better support for right alignment
- " PRO: Added new constant [csPropGraphBCRatio](#) to control de BC Ratio
- " PRO: Moving from one page to another updates cursors
- " Takes care of GIF transparency
- " When setting the csPropGraphDashLine, the property will not turn off with a zero value.
- " Added support for native CJK fonts (Chinese, Japanese and Korean) using Unicode
- " SetMargins accepts -1 parameters to only affect the margins you really want to set
- " New function [SaveVariant](#) for use on .NET applications, now its fully compatible with .NET (unsafe pointers)
- " New property that controls if an image can be broken up by a page break or should create a new page if it doesn't fit. Only AddGraphic Function. csPropGraphFullPage (True by default, if it doesn't fit it will add a new page)
- " Rewritten all debug functions, now you can debug remotely your application by using TCP/IP connection with the aspEasyReg tool

2.06 (01-Feb-2004)

- " PRO: HTML was not getting the correct styles
- " PRO: Graphics on cells aligned right were not correctly positioned
- " PRO: Title tag left a blank space that was printed on the next paragraph
- " Using binarywrite function with debug to screen automatically aborts post request to not corrupt the PDF document

2.05 (23-Jan-2004)

- " PRO: STYLES are ignored and they do not print the styles on the document as text.
- " PRO: Tables with a bigger width than the page is adjusted automatically to make it fit to the page.
- " PRO: Cell center alignments with more than two lines are correctly aligned
- " PRO: Fixed some internal tags to allow merging with 3rd party utilities
- " PRO: Skip blanks spaces Improved on html pages
- " PRO: Adding object forms with inverted coordinates would not drawn the object.
- " PRO: To mark by default a checkbox you should pass "On" as a parameter
- " PRO: Font sizes greater than 7 were ignored
- " PRO: External Fonts in html referred with F1x were ignored
- " Compatible release of VisualEasyPDF 2.0 forms
- " Version 2.04 was not correctly loading the VEP forms
- " Was not using the correct color background when loading VEP files
- " Adding text on page jump with the csPropTextVertSpace activated could lost the last line
- " Shadows are correctly positioned inside the rect frame and not outside.
- " AddText and AddTextWidth (with csPropAddTextWidth=2 constant) now accepts angles

2.04 (05-Jan-2004)

- " PRO: Table cell was not removing a space at the beginning of the paragraph

- " PRO: Tables with a bigger width than the page is adjusted automatically to make it fit to the page.
- " PRO: Tables across pages in certain circumstances was not write in the next page and was overwriting all pages
- " PRO: Future bookmarks were not working
- " PRO: Sometimes the page break was not adjusting the correct Top margin
- " PRO: Headers tags were not adding a branch return
- " PRO: Headers have the align property added
- " PRO: Table with image ratios where not right scaled
- " PRO: URL & Bookmarks for page jumps can be defined even if the page does not exists.
- " PRO: URL & Bookmarks for page jumps now you can specify just the page information and is not necessary to specify the coord.
- " Graphics memory leaks
- " Save function controls exceptions and prints the error information when debugging it (done to prevent locked files)
- " Inserting text in different format just when there is a page break was misplacing the text on the next page
- " AddGraphic imports WMF and EMF vector graphics on the correct background color
- " Two new constants to control the import size of a vector graphic; csPropGraphWidthImport, csPropGraphHeightImport
- " GetProperty added for some of the constants that were lost
- " If forms fields has blank text or blank hint then it was corrupting the document
- " /UL and /OL was not closing the Order list
- " csPropIntDebugTime
- " Debug information when there is an error on AddHTML function
- " Save debug information when the document has been saved.

2.03 (17-11-2003)

- " PRO: TD Align=right was adding a carriage return
- " PRO: Nested tables with a table of 100% and some fixed columns was not getting the full 100% of the cell
- " PRO: Security was not working for old Acrobat versions (3.0 to 5.0)
- " Saving several times the PDF document on the same instance was corrupting the final document.

2.02 (13-11-2003)

- " PRO: Security (Encryption) was not working when adding images.
- " AddGraphic function was converting 24 bits images to 8 bits by error.

2.01 (10-11-2003)

- " AddGraphic was inserting twice the graphic
- " For licensed users: gives more information on debug license
- " For licensed users: use of MAC address if the HD serial fails

2.0 Final version (07 -1 1-2003)

- " PRO: 128 bits strong security added
- " PRO: Securities on 40 bits was requesting an user password if left blank. Corrected.
- " PRO: AddNote now returns a name that can be used on the SetPropObj
- " Corrected some fails on the help file
- " Ellipse now has shadows
- " Added all constants definition inside the library
- " SetPropObj now accepts Notes, now you can change the default positions and sizes
- " csPropTextOverGraph added again for graphic priority

2.0 Release Candidate 2 (27 -10-2003)

- " PRO: Improved html rendering and detecting white spaces bugs from IE and using it to produce the same output.
- " PRO: Added a new function called AddPattern, to add footers and headers
- " PRO: Added a new constant for HTML rendering csHTML_ImageRatio (default: 1.35)
- " PRO: Added two internal tags: <!--PDF.PAGENUMBER> and <!--PDF.PAGECOUNT>
- " PRO: Multiformat center accepted for HTML
- " Bookmarks failed if there were subchilds added. Also added a new property to specify if it's opened
- " JPG quality at 100% by default new constant csPropGraphJPGQuality to choose a different one (100 best 0 worst)
- " Trial text reminder redesigned and removed the limitation of only two pages (Released a RC1a for shareware users)
- " Added a new item to control the open status of the open outlines: csPropObjOpened & csPropObjText
- " Improved graphic render when is 8 bits or lower
- " Possibility to load an individual TIFF multipage or GIF frame image (csPropGraphImageIndex)
- " The library does not store GIF or TIF (with LZW compression) images on the PDF, this doesn't vulnerable the Unisys Patent.

If you use compression you may vulnerable this license if the Patent still available on your country, so you should contact Unisys to make an agreement with them.

We strongly suggest to use the PNG format instead of the GIF format.

- " Added more importing graphic formats see AddGraphic function

2.0 Release Candidate 1 (13-10-2003)

- " Added VEP loadable forms, draw as a template, and dynamic change of VEP objects before insertion (Needs VisualEasyPDF version 1.01)
- " Improved nested tables
- " Finished the help file
- " If graphic was not found the library was crashing
- " Border, CellSpacing, Cellpadding can be decimal variables
- " Coded all 1.72 features and corrections (Coded for big stress conditions and for multiple CPU / load balancing servers)

2.0 Beta 1 (11-09-2003)

- " First beta version of 2.0 ;-)

This is the first Beta published, please report any error on the forum. Thanks.

At this moment we are still working on some issues that will be fixed on the final release of 2.0;

- Nested Tables
- Scroll list doesn't print the current list, you must select one to display it correctly
- WinXP and MacOSX design objects
- Full support to the Unicode
- CKJ Fonts
- To finish the Constant help file section

General news from version 2.0:

- AddGraphic, PNG graphic format added
- AddGraphic, TIFF graphic format added (there is no option of multi page)
- AddGraphic release correctly the memory when there was an error loading the graphic
- AddGraphic is no more limited to 100 per page, now is unlimited and has faster load process(memory handling)
- AddGraphic returns Boolean value to know if the graphic has been added
- SetTrueTypeFont -- > AddFont
- Fonts are unlimited, there is no more limitation of 15 fonts.
- You can use any name to define the TTF fonts, the F1 to F14 are reserved for PDF internal fonts
- Unique identification of each PDF file
- Internal pointers is not more using words (limited to 65k def) now we can use 4.294.967.295 objects internally, this means that maximum number of pages, graphics, fonts... etc.
- Password protection (user and owner)
- Permissions (printing, changing, notes...)
- Dash Lines properties
- Can use 40 bits encryption and is prepared for the 128 bits
- New Function: AddEllipse for ellipses and circles
- Advanced functions grph_xxxx
- AddNote
- URL jump labels can be set before defining it
- Faster memory release when destroying the PDF
- Outlines (bookmarks)
- Rounded Rectangles with or without shadows
- Annotations for Text popups and marks
- Different country different legal sizes...add - (US Legal?) 8 1/2 * 14
- CYMK support for text, html, primitives and advanced graphics
- csPropTextVertSpace property to control vertical space, default 0
- csPropGraphShadowPos
- csPropGraphWidthShadow is float number and fills boxes
- Forms.
- Buttons, Edit boxes, combo boxes, list boxes, check boxes and radio buttons.
- Forms File default creations, FDF files.

- JavaScript for object forms. Event programming.
- Table inside table
- License now checks multiple IP address and has a better debug technique. Now supports Windows 2003 Server
- License return true or false if it works or fails to load the license.
- Help file revised and got a new interface to better seek information.

1.71 (05-08-2003)

" Corrects a big problem found by Russ Perna that was generating additional fonts information and that was creating a corrupted PDF. I just left a compile define tag misplaced on the latest release which was using a feature of the 2.0

1.70 (01-08-2003)

- " PRO: Add SUB and SUP tag for html text
- " PRO: Images in cells with Align CENTER was not respecting the left margin
- " PRO: TD tag now accepts HEIGHT value
- " PRO: Using P tag inside a cell was adding an additional CR
- " PRO: Left blank spaces removed when using addhtml and combining different tags on a paragraph
- " PRO: Added two new properties csHTML_FontName = 252; csHTML_FontSize = 253; for default fonts when using html
- " PRO: Table alignment
- " PRO: URL are no more limited to 80 characters size, can be any size.
- " PRO: Html has a better support for nonsense blanks characters that are correctly removed
- " PRO: Added the TH tag, works equally to the TD tag
- " PRO: When using the DIV or P tag in the first cell was adding some extra lines, removed.
- " PRO: added more protection to the html tables, if you miss some tags on the html it doesn't trough an exception
- " PRO: Using the http request now detects the type graphic format, so it will use the correct graphic when using img with .asp, .apx or whatever extension you pass with params
- " The csPropTextAngle now accepts angles see help to see how to use it
- " Added a new constant to update cursors when using AddTextWidth or AddTextPos(csPropAddText_UpdPos)
- " Added unlimited page support
- " Width lines can have decimal units, now supports 0.5, 0.1 weakness lines
- " Adding a single text on the right breaks margin
- " SaveString
- " Delphi sample was wrong, it was an old project test that didn't have nothing to do with the component.
- " Saving twice the document or in combination of BinaryWrite with graphics was corrupting the PDF document
- " SaveStream now returns a memory pointer to the PDF in memory.
- " AddGraphic, Bitmap 24bits supported (it will convert it automatically to JPG)
- " AddGraphic was not recognizing the JPEG files as a JPG graphic
- " Adding an unknown graphic was corrupting the PDF file, this has been corrected.

1.61 (01-05-2003)

- " PRO: UL & OL Tags without parameters corrupted the document
- " PRO: Consecutive Branches
 were ignored
- " PRO: The AddHTML restored the AlignGraphic to the left, this was wrong and there were unexpected results when calling several times the AddHTML method with graphics.
- " Positioning graphics was not working anymore, there was a bug on the width, height calculation

1.60 (30-04-2003)

- " PRO: Colspan feature for tables
- " PRO: Increment the maximum external font from 5 to 15 different fonts. (F15-F29)
- " PRO: True Type fonts didn't use the correct right margins
- " PRO: True Type not embedded was not using the styles
- " PRO: added the BGCOLOR for Rows when using tables
- " PRO: added HR tag with size and color property
- " PRO: Empty Rows are sized to the correct height
- " PRO: Tables between tables are joined
- " PRO: Values with px properties was not right converted
- " PRO: Added a new property csPropGraphBCText to control the display of a label on the Bar Code graphic
- " PRO: BarCodes were flipped vertically, it was no problem for scanning them, but the text appear reversed
- " PRO: Barcodes uses JPG, avoids conflicts from old graphic cards when using bitmaps
- " PRO: New barcode property to set the angle. It supports only the 0 , 90 and 270 degrees for drawing it vertically.

- " PRO: URL, type 4 was using adobe coordinates, now is using the same coordinates as stipulated on the component
- " PRO: AddURL for type 2 was not working
- " PRO: <!--PDF.ADDPAGE>
- " PRO: <!--PDF.MOVETOPAGE>
- " PRO: <!--PDF.ADDBARCODE>
- " PRO: <!--PDF.ADDTEXTPOS>
- " PRO: <!--PDF.SETPROPERTY>
- " PRO: New function to retrieve the Barcode width: GetBarCodeWidth
- " Setting margins will update the Y cursor and the right position, it has been also added debug information
- " Border images for JPG was drawn after the graphic not inside.
- " Image Borders were not scaled correctly when using zooms
- " When adding the same image using relative path it was not reusing it with the previous one.
- " Using the
 entities doesn't interpret like a branch return

- " The border box of DrawBox function it was drawn with a width line set to 0
- " On the AddText function you can use the file:/// property to load a text file
- " Optimized for big texts to be inserted on the AddText function
- " Added a new function GetTextHeight
- " AddTextWidth now has different options for cutting or wrapping the text with the csPropAddTextWidth property
- " AddTextWidth was not placing correctly the text with the Y cursor (not solved on version 1.53)
- " Add Debug information on AddTextWidth function
- " Positioning functions were not getting the correct page size
- " Letter size was wrong, it showed 8.46" x 10.98" instead of 8.5" x 11"
- " Better stress work condition

1.53 (14-02-2003)

- " PRO: BarCodes didn't work on version 1.52, corrected and uses the new priority for graphics
- " PRO: Corrected the problem with the border property when it was 0 for the table
- " PRO: cells across pages where printed on the same page
- " PRO: cells across pages were bigger than expected
- " PRO: csHTML_TRFullPage constant was not working
- " PRO: Table border from the right was not well drawn
- " PRO: The ordered list accepts two properties, the start and the Type (for type 1, a, A, i, I)
- " PRO: The debug information for AddHTML has been improved by replacing the < > by < and > this will prevent the explorer to render the debug information.
- " Added results on the GetProperty when debugging it.
- " AddTextWidth was not placing correctly the text with the Y cursor.
- " The character \ wasn't print on the document with the AddText function
- " Underlines that were break by the end of a line wasn't drawn on the next line
- " Margins on the bottom of the page wasn't correct when inserting text with the AddText function and forcing a page break
- " AddGraphic was not positioning at the right place on the beginning of a page insertion
- " Use of entities in AddText and AddLink, you can change the default action with the [csPropTextEntityConv](#)
- " Bug on csPropIntCoord used in conjunction with some cursor positioning functions
- " Decimal conversions when the users inputs an error the component raises with an exception, now by default it assigns a 0 to the value.
- " Added a new constant property to control relative paths, [csPropIntRelativePath](#)

1.52 (01-02-2003)

- " PRO: Added relative paths when using ADDHTML with a file:/// clause, then it assumes that the directory which contains the file is the actual directory to load graphics and so on.
- " The (AddGraphic was not returning the right Y position) from version 1.52 was not working with html images in tables

1.52 (31-01-2003)

- " PRO: HTML files are interpreted faster
- " PRO: Blockquote tag was not working fine
- " PRO: MoveToPage can be used in conjunction with AddGraphics
- " PRO: New property to control TR tag for tables, [csHTML_TRFullPage](#) will force that every row is printed in one page, you don't get a row in two pages.
- " PRO: Skips trash characters from the HTML that doesn't have to be render on the document, improves size and speed for the PDF document.
- " PRO: When adding a graphic on a table cell it was displaced lower than it should be
- " PRO: On table cells using a different font size for the beginning of a text was not correctly draw
- " PRO: Using AddLink of type 4 (page jump) and using float values, the PDF get corrupted.

- " PRO: Added full support to the ISO Latin-1 Character Set entities in html
- " PRO: Convert decimal entities to character set (&#xx;)
- " PRO: Optimized annotation (AddLink)
- " Debug to log file when using VB or another language that is not ASP now works
- " Licensed version can debug license problems from VB or another language.
- " AddGraphic was not returning the right Y cursor position
- " AddGraphic takes priority on the same way that has been inserted, now the csPropTextOverGraph has no effect.
- " Using IIS relative path for adding graphics when it doesn't find the file. Uses the same path that has been call for the asp script. (ASP only)
- " Possibility to add graphics with fixed pixel size zoom independent from width and height. (So if it founds -1 on the value then it will use the same size of the picture)
- " The csPropGraphYAlign was not working, it has been corrected.

1.51 (09-12-2002)

- " PRO: Script tag is ignored and it does not display the contents of the script
- " PRO: Added
 tag for XLS transformation
- " PRO: <i> and for Italic font
- " PRO: When using <center> it doesn't add a new line if the cursor is at the beginning of the line.
- " PRO: Bullets supported with the AddHTML
- " PRO: Adding a table with TR tags and without TD tag was causing an exeception.
- " PRO: Tables across pages was not correctly drawing the border cells and the background color.
- " AddText, AddHTML with parenthesis was not calculating the right size.

1.5 (31-10-2002)

- " PRO: Script tag is ignored and it does not display the contents of the script
- " PRO: Added the property of Border on the IMG tag
- " PRO: Text compression level. csPropIntLZDocLevel (ID: 507)
- " PRO: Fonts now have properties to set bold and italic.
- " PRO: Bug when using more then 2 external fonts on the same PDF
- " Charspace does not reset when using the AddText and setting it previously to 0
- " New function called [AddTextWidth](#) , which controls the text into a position with a width size and you can use the alignment property.
- " [csPageAutoAdd](#) property to controll if the component add automatically pages, by default is true. (ID: 211)
- " [csPageCount](#) property to get how many pages you have created.
- " 16bits and 24bits on BMP graphics are converted to 8bits, it doesn't raise an error anymore
- " Code Improved to enable web stress process
- " BorderGraphic property when adding graphics images. [csPropGraphBorder](#) (ID: 413)
- " Calling twice or more times BinaryWrite or SaveStream was increasing memory and not releasing it
- " When using OnEndPage or Destroy without using create or OnStartPage was causing a crash, now it ignores the error and do nothing

1.41 (26-08-02)

- " PRO: [MoveToPage](#) was not controlling correctly the page number, it was getting wrong.
- " [MoveToPage](#) now returns true or false if it was moved or not
- " Better debug information for [MoveToPage](#) and [GetTextWidth](#)

1.4 (16-08-02)

- " PRO: Adding a second table on a HTML was losing the right margin on the second table.
- " PRO: New function [MoveToPage](#) , to move between page after creating them.
- " Help document changed for csPropGraphZoom, the correct constant number is 412 and not 411. Added also on the include ASP file.
- " New property to set the [Debug](#) output to a file
- " Wrong position when adding a JPG Graphic and then a text without specifying absolute position
- " Gif graphics where giving problems on old graphic cards, this has been solved with the help of Florain Langner who help me to find this bug.
- " The HTML font face tag was ignoring the internal fonts or the True type external fonts, use it with font face="F15"

1.33 (07-06-02)

- " PRO: Align method works for html tables (cells)
- " PRO: When fixed size for a cell, it breaks up in multiple lines, it grown down and not on the right
- " PRO: Added a new tag to do Page breaks from HTML pages; <!-PAGE BREAK->

- " PRO: Removed more than one spaces on the text if it's not specified with on html
- " PRO: Images on table cell are positioned
- " PRO: Added entities (" < > &)
- " GIF Color reduction has been set to none
- " Fixed a bug when using the BinaryWrite and using reentrant code, it was generating an exception every time the object was closed.
- " When adding twice the same graphic on the first page, it was loading it twice and it didn't use the cache. This has been corrected.
- " If file is not found it shows an error on debug mode, but it never raise the program
- " HTTP graphic now try to get it, if you have an error then displays it (only with debug property) and it does not displays any error message on the PDF.
- " New property for controlling site licenses (only registered users)

1.32 (03-05-02)

- " PRO: Codebar height was getting the zoom size and not the scale factor
- " Flag memory when using a lot of pages
- " Memory was not been releasing when encoding with LZ
- " When using left and right alignment in just on line, the right was not aligned
- " Top margin was not adjusted when setting a new margin

1.31 (03-04-02)

- The http request was wrong when there was no proxy server enabled.

1.3 (29/03/2002)

- " New installer, auto registers the component and installs samples
- " New tool to help registering and detect error with the component and the IIS
- " Setting individual margins affects the actual page, this makes the ability to change dynamically the left and right margins.
- " PRO: Added tables support for HTML, read help to see what tags are supported
- " PRO: Added IMG tag for HTML
- " PRO: Support for proxy when retrieving from http
- " PRO: Now you can add graphics from http: with the addgraphic and addgraphicPos
- " PRO: Named colors using html
- " PRO: Paragraph end tag is optional </p>
- " Properties that set colors can use the 16 named standard colors, see docs.
- " Zoom factor less than 100% for JPG causes the graphic to print out of the page
- " Zooms can now be specified in pixel sizes. csPropGraphZoom (0 to use percent zoom and 1 to use fixed width and height pixel size)
- " Help file include on the installation, now you can see it offline
- " Internal Error message in English, there is no Spanish messages
- " Internal Floating conversions does not assume you use ',' as decimal separator, now retrieves your default regional configuration
- " Added GIF resolutions for 5,6,7 Bits (32, 64, 128 colors)
- " Barcode does not use shadows anymore, the scanning devices were getting bad readings
- " csPropGraphBOXOnBack

1.2 (PRO RC5) (15/02/2002)

- " PRO: When retrieving a html file it will skip the hexadecimal chars 0xA and 0xD
- " PRO: HTML Anchor tag <A> has full support to jump inside the document
- " PRO: Links can now be wrap when it finds a margin break (AddURL. not for the AddURLPos)
- " PRO: Links could cause an exception when using it on multiple pages.
- " PRO: Added Body tag with the background color for the document
- " PRO: Added full support to the links
- " GetProperty now returns html colors
- " GetProperty csPropPosX and csPropPosY was returning an integer and was a Float variant
- " Angle in radians of the text to display for the addTextPos
- " 3D Shadow text on the addTextPos
- " New setting for the BackGround colors for the page or for the whole document
- " You have a new option to designate when opening the document to fit to the whole page on the window. (const csPropIntFitWin = 503)
- " Properties to control the spaces between characters and words. Warning as it does not work for the addtext method, only for addtextpos. Also with gettextwidth you will not get the correct size width when using those properties.
- " Another option to open the document to full screen (const csPropIntFullScreen = 504)

1.10a (31/01/2002)

" Big bug found with big documents, for more than 64k, that the offset variables from inside the document where handling no more than 64k it has been resized to allow a maximum of 2GB, so this is the limit of a PDF file created with the aspEasyPDF, I really don't know if PDF reader would handle this. ;-)

1.10 (25/01/2002)

" Create and Destroy methods

" Bug when adding two different graphics on each page that was mixing the information

" **PRO:** AddHtml now can use url from different sites or files

1.10 RC2 (14/01/2002)

" **PRO:** Now the Addhtml remembers all fonts used on the html. So when you add 4 more fonts tags when finishing it it will remember the 4 different fonts.

" **PRO:** Add on the html the blockquote and anchor links

" Underline was not using the margins, this has been fixed.

" **PRO:** There where some blank spaces that where not drawn on html.

" **PRO:** Fixed the bug for the align="center", it was general fault for which was releasing more memory that was not allocated.

1.10 RC1a (13/01/2002)

" **PRO:** Bug found on the True Type font, it was adding the Arial font even when specify another

" **PRO:** Code Bars now works fine

" **PRO:** Bold tag implemented , only work with F1 font

" **PRO:** Found a bug with the align="center" that causes an exception, I didn't found where it comes, hope to get it fixed on next release.

1.10 RC1 (11/01/2002)

" It uses the fonts that you use on the document not the all 14 fonts, this reduce the sizes and draws fast.

" blank spaces at the beginning of the word and in the end was trimmed when using the AddText method.

" The csPropPar margins has been replaced by the page margins and is defined for each page.

" The csPropIntCoord now affects alls coordinates, graphics and positions.

" Complete underline for addparagraph (not supported for justified align)

" Underline now draws with the same color of the text

" Underline bug fix when AddTextPos with -1 argument

" Added some new methods, SetFont, PageSize

" International charset size

" Concatenated branches when using the addParagraph

" Underline height varies from the size of the font

" When adding bitmaps and Gifs there was a bug that didn't release the work buffer memory when destroying the object. The JPG was not affected.

" **PRO:** Coded all features inside the 1.10

1.03 (8/12/2001)

" The BinaryWrite it now fully supports graphics, so you don't have to care about saving files everywhere and also about the privileges to be write enable.

1.03 (6/11/2001)

" Removed the bug from the Right Paragraph margin.

" When adding the margins from right to left it was not well justified when drawing text.

1.02 (20/10/2001)

" Reentrant creation of the component was not releasing all the memory and crashed when creating more than 2 times

1.01 (2/10/2001)

" Small bug on the registered version and handling the license file

1.0 (01/10/2001)

" A lot of changes, please make a backup of the last binary version because you can get a different look from this version.

" Removed the PDF message that the file was damaged and was being repairing, this correction has made a lot of internal changes. It has been improved the speed on the PDF reader, the document is generated in the fastest way for PDF.

" Priority for text / graphics, text over graphics or down the graphics.

" Removed the limitation of 10 graphics per page to 100 graphics per page.

" When adding two same file graphics, then it only adds one with two different positions. Optimize the size of the PDF document.

" Now you can specify the color of the boxes and lines, and also the fill color for the boxes.

" Added shadow boxes

" Measure the text length with the GetTextWidth method

" Added more properties to control the page number, width and height.

" For registered users, now you can specify where is the license file allowing you a different location of the license file. Better control on different IP address. Add the unique ID machine for multiple address.

" and added internal functions that improves speed and reliability.

0 .99 Release Candidate 8 (02/08/2001)

" New license module for registered users, this will provide standalone updates.

" Added new properties to control the graphic position and size, csPropGraphXAlign, csPropGraphWZoom and csPropGraphHZoom .

0 .99 Release Candidate 7 (20/07/2001)

" Compatible with Visual Basic and Scripting technologies (JavaScript and Basic Script)

" Corrected the bottom and top properties that were ignored

0 .99 Release Candidate 6 (17/07/2001)

" Corrected the addText when changing color property was adding a new line

" The addGraphic was not positioning to the actual position of the cursor

0 .99 Release Candidate 5 (16/07/2001)

" Corrected the addText paragraph for auto adding the next page with the corrected position without overlaying it.

" Fixed the BottomEnd property for the margin

" Corrected the addText function to add special characters " () ".

" AddBox was buggy

0 .99 Release Candidate 4 (13/07/2001)

" Changed the way that it works the addText, now it doesn't add paragraphs. See documentation to know more about this change.

" Added two new functions to quick control the positions and the margins of the page.

0 .99 Release Candidate 3 (29/06/2001)

" Added JPEG support whit 24 bits!

0 .99 Release Candidate 2 (5/06/2001)

" Added the right paragraph space (csPropParRight)

0 .99 Release Candidate 1 (4/06/2001)

" Graphic support. Bitmaps and Gifs in 8 bit resolution (256 colors)

" Bitmaps are compressed inside the PDF with the LZ (the same used by the gif format)

" 2 more fonts added, symbol and Zap

" Underline option with AddtextPos method

" Page properties, now you can set orientation and page size

" Properties to control the paragraphs position

0 .95 (17/05/2001)

" Small bug found on PDF reader 5 when drawing lines or boxes it was displaying a message about an illegal m operation on text box. On PDF reader 4 and previous was working ok.

0 .92 (14/05/2001)

" Added a lot of request from the users

" String errors translated to English, (they were in Spanish ;-)

" Added colors to the AddText and the AddTextPos method, it looks beautiful and you can specify directly the RGB color, like the font color in HTML.

" Added space between paragraphs. (see SetProperty csPropParSpace in docs)

" Added the header information, you can specify the title, author and other PDF properties. On PDF Reader press CTRL+D to see the window properties.

0 .91 (14/05/2001)

" Small bug removed.

0 .9 (13/05/2001)

" First version, full functional and released in Beta state to see how it works and what the users says in productivity .

General page information

e asyReportPDF

Last modified : 23/12/2005 (Version 1.06)

Information

easyReportPDF is a library for developers to create **Reports in PDF** from any data source. It's based on the award winning aspEasyPDF library for rendering fast PDF documents through the WEB and being optimized to use it on any application development that needs to use **real-time Data Reporting** by connecting to a database server.

But what really differs from aspEasyPDF library? We have simplified aspEasyPDF, added database connection, fast scripting interpretation and finally report rendering features; The result is that you design the report with VisualEasyPDF, which is a product designed for data reporting (*included on the license*), saves the report and run it from your application without coding a line for the report. No other library makes it easier, take a look on the tutorials to learn more.

Key features

PDF Reports

Native PDF rendering by including the aspEasyPDF library inside the easyReportPDF which results a **fast real-time PDF Data Reporting library**.

Design reports

easyReportPDF includes a license of **VisualEasyPDF** to develop the reports. You don't need to code your application to render the report, just set the report file and render it directly to the printer.

Parameters

Create user parameters and automatically ask the user to input the value. The report will react with the user input data and will show the correct information that has requested for.

Grouping

Create up to **10 grouping level bands**.

Scripting

You need that the report reacts to certain values when rendering it, then just code with our fast scripting technology and debug it with VisualEasyPDF before posting the report to avoid any error.

Fast Maintenance

Found an error on your report and need to solve it fast; the reports may be altered directly through the server and the changes will be effective immediately for your users after pressing save button in VisualEasyPDF.

Databases

It includes **msSQL** (© Microsoft Corporation) , **MySQL** (© MySQL AB.) , **Oracle** (© Oracle Corporation.) , **PostgreSQL** (© Portions. The PostgreSQL Global Development Group) , **SyBase** (© Sybase Inc.) , **FireBird** (© Firebird Project.) , **Access** (© Microsoft Corporation) and **ADO** native drivers to retrieve the data faster than any simple ODBC connection. Other databases may be accessed by using ADO native drive connection.

Versions

The library is **Shareware** which has one limitation; it displays a message on each page of the PDF document, the **Cripleware** technique is used to guaranty piracy protection. **The price? Starts at just US\$ 159 per machine!**

The **Professional** version adds functions that helps you to alter object properties and connections settings in real-time; see help file to see which are those functions that are only available with the Professional version.

- aspEasyPDF -

The SITE licensing option helps you achieve significant savings on volume purchase of MITData products. SITE license entitles you to install and use the software on unlimited number of workstations.

If you need all the aspEasyPDF functionality in the easyReportPDF then get the aspEasyPDF Enterprise version which adds all the features of easyReportPDF inside the aspEasyPDF.

Different versions of easyReportPDF

The different versions are all coded on the same DLL (reportpdf.dll) and the license file that you purchase sets the library to use one version or other. This way you may upgrade your license with just using a different license file.

Standard version is the basic version which loads the report and renders it. You can not use compression, encryption or other internal functions which interact with the report on your code. All those functions are marked with an asterisk in the help file.

Professional version does what the standard version do but adds compression properties, encryption and internal function that interacts with the objects from the report. Those functions are marked with an asterisk in the help file.

Feature	Standard	Professional
aspEasyPDF library coded	Yes	Yes
Support for Bitmaps, GIF, TIFF, PNG, JP2000 and JPEG	Yes	Yes
Chart capability	Yes	Yes
Use code bars (EAN, Code39, Code128, etc)	Yes	Yes
Supported Databases (ADO, msSQL, mySQL, FireFox, ODBC, Access and more)	Yes	Yes
Render reports directly to the printer or to the browser	Yes	Yes
Free license of VisualEasyPDF	Yes	Yes
Load VEP files done with the VisualEasyPDF design tool	Yes	Yes
Loads reports created by VisualEasyPDF	Yes	Yes
Adds all technology features from easyReportPDF into aspEasyPDF	Yes	Yes
Create dynamically reports	Yes	Yes
Add dynamic script to your report	Yes	Yes
Manage user parameters directly	Yes	Yes
Compression and security encryption in 40 bits or 128 bits	No	Yes
Functions that interacts with VisualEasyPDF objects; setDBConnection ; Changes connection properties, this is useful to setDBSQL ; Changes the SQL dynamically before running the report. getFirstObject , getNextObject; retrieve objects from the report getPropObj , getProperty ; gets property objects setPropObj , setPoperty ; sets property objects	No	Yes

Release History

1.06 (29-November-2005)

- " HTML DBFields now uses the default font and size specified for text
- " Added the **DBExecSQL** script function
- " Added the **setParameter** and **getParameter** script functions
- " Added **@Param** field for printing directly the parameters in the page
- " Text inside functions can be positioned at any place and any times
- " If you now close the status window it will cancel the report (only for visual run)
- " MS Access, ODBC and Oracle databases are full supported
- " Now correctly closes all open connection after drawing the report
- " Blank parameters are correctly set to the report

1.05 (21-October-2005)

- " Shows Connections errors in the PDF inside a round rectangle in red. This will avoid connection errors with blank reports without knowing the error.
- " Displays internal script errors error messages
- " Parameter dialog now the OK button has default focus
- " Corrected some errors on the Expandable field that didn't work if it wasn't set with HTML contents.
- " New property to control the maximum pages that will print the report, this will enable you to control large reports errors that takes too much time to generate. (**MaxReportPage**)
- " Added the **DBQuerySQL** script function to create a new query for a given connection.
- " Added the **AddTextWidth** and **AddGraphicPos** script function to write text and draw graphics from the script.
- " Skip some errors after a connection break, it will display just one error, no consecutive errors for each record.

1.04 (19-October-2005)

- " TimeOut property in connection
- " SetDBConnection accepts [ALL] has object name to affect all connections object defined in the report
- " Scripts returns variants and reads variants, this is helpful to manage decimal values.
- " Small bug fixes.



1.01 (19-September-2005)

- " Added the Clear function
- " Added NVersion function
- " Added Connection constants (see Constants help)

1.00 (1-September-2005)

- " Public release

Debug

Property explain	
Name	Debug
Syntax in VB	PDF.Debug = True / False
Parameter type	Boolean
Read / Write	 / 

Sets the library in debug mode. This is very useful for debugging your application if you get errors and wish to see what's happening.

If you activate this option when you made any operation on your program it will displays what's going on into a file or the screen. Default is set to False, so there is no debug information when running up.

To active the Debug to a file use the [csPropIntDebugFile](#) , by default it will output all information on the screen.

Note: Do not use the debug property in conjunction with ASP and BinaryWrite, it will output the information directly to the PDF reader and it will fail with an error. If you want to use the Debug option then use the Save function.

Example in ASP

```
<%  
PDF.Debug = True  
%>
```

Lic_Debug

Property explain	
Name	Lic_Debug
Syntax in VB	PDF.Lic_Debug = True / False
Parameter type	Boolean
Read / Write	✔ / ✔

This properties is only for registered users and outputs debug information on how it loads and checks the license. This is useful to check if there is a problem with the IP address of the machine and the licensed file or if the license is corrupted.

If it displays any problem, please send this debug information to support@mitdata.net to repair and send a new license file.

Example in ASP



```
<%  
dim PDF  
' Debug the license routine on the PDF version  
' This is done when you get problems on the license machine  
set PDF = server.createobject("aspPDF.EasyPDF")  
PDF.DEBUG = True  
PDF.LIC_DEBUG = True  
' Loads the license file, change it if you use a different path  
PDF.License("easypdf.lic")  
response.write "<br>Version Information:<br>" &  
PDF.Version  
set pdf = nothing  
%>
```

See also

 [License](#)  [SiteLicense](#)

LastError

Property explain



Name	LastError
Parameter type	WideString
Read / Write	 / 

Gets the last error from the library. The Error is a string that contains the description in English of the error that was issued by the last command executed.

To reset the error just assign it to an empty string.

NVersion

Property explain

Name	NVersion
Parameter type	Integer
Read / Write	 / 

Returns the version number of the library, you can check it to provide compatibility between different versions of the library.

Returning values

100 - this is version 1.0

110 - for version 1.10

132 - for version 1.32

200 - for version 2.00

Example in VB



```
if PDF.NVersion < 200 then
    msgbox " You have an old version of aspEasyPDF installed on your machine, this not supported by our
program "
end if
```

See also

 [Version](#)

Version

Property explain

Name	Version
Parameter type	String
Read / Write	 / 

Returns the version information of the library and the IP address from your network machine, this last information will be needed when buying the library.

Please support us and register it for developing better and new programming libraries.

See also

 [NVersion](#)

PosXCursor

Property explain

Name	PosXCursor
Parameter type	Double
Read / Write	✔ / ✔
From version	2.00

This property controls the cursors inside the page, this is quite useful to use it in combination with the [AddText](#) method. You have two options to position text on the document, the [AddTextPos](#), that gives you the option to print everywhere with cursor position and the [AddText](#), that adds consecutive blocks of text on the page.

If you work with the [AddText](#) you may need some times to control the cursors, knowing where we are putting the text and changing it to another position to take it faster to that place.

This property controls and changes each time we insert text on the page the X Cursor, Horizontal position. This affects to the [AddText](#), [AddGraphic](#), [AddLink](#) and [AddHTML](#).

Example in VB:

```
' From the X position move it 10 mm left  
PDF.PosXCursor = PDF.PosXCursor + PDF.cnvUnitmm  
(10)  
PDF.AddText "Hello world"
```

See also

[PosYCursor](#) [csPropPosX](#) [csPropPosY](#)

Applies to:

[AddText](#) [AddHTML](#) [AddLink](#) [AddGraphic](#)

PosYCursor

Property explain

Name	PosYCursor
Parameter type	Double
Read / Write	✓ / ✓
From version	2 2.00

This property controls the cursors inside the page, this is quite useful to use it in combination with the [AddText](#) method. You have two options to position text on the document, the [AddTextPos](#), that gives you the option to print everywhere with cursor position and the [AddText](#), that adds consecutive blocks of text on the page.

If you work with the [AddText](#) you may need some times to control the cursors, knowing where we are putting the text and changing it to another position to take it faster to that place.

This property controls and changes each time we insert text on the page the Y Cursor, vertical position. This affects to the [AddText](#), [AddGraphic](#), [AddLink](#) and [AddHTML](#).

Example:

```
' Jump to 100 mm from the top  
PDF.PosYCursor = cnvUnitmm(100)  
PDF.AddText "Hello world"
```

See also




[PosXCursor](#) [csPropPosX](#) [csPropPosY](#)

Applies to:

[AddText](#) [AddHTML](#) [AddLink](#) [AddGraphic](#)

PageNumber

Property explain

Name	PageNumber
Parameter type	Integer
Read / Write	 / 
From version	 2.00

Returns the Page number of the actual page that is being drawing. For PRO users this property can be set to move from page to another.

Note: First page is number 0

Example in ASP

```
<%  
set PDF = server.createobject("aspPDF.EasyPDF")  
NPage = PDF.PageCount  
For P = 0 To NPage  
    PDF.PageNumber = P  
    PDF.AddTextPos 10, 10, " Page: " & P+1 & " of " & NPage +  
1  
next  
set pdf = nothing  
>%
```

See also




 [PageCount](#)

 [csPageNumber](#)

 [csPageCount](#)

PageCount

Property explain



Name	PageCount
Parameter type	Integer
Read / Write	 / 
From version	 2.00

Returns the number of pages that has been created on the PDF document

See also

 [PageNumber](#)  [csPageNumber](#)  [csPageCount](#)

SiteLicense

Property explain	
Name	SiteLicense
Parameter type	String
Read / Write	 / 

When you purchase the library you have two licensing methods, one is that you pay per server license (one license for one computer) and the other you pay once and you can install it everywhere or distributed it with your application without paying any royalties for that.

If you decide to purchase the SiteLicense option, then we can send you two different versions of the library, using the license file or compiling the DLL just for you with your information encoded inside the library:

Using License file:

This is the default method that will use our eCommerce system to compose the license, what it means is that you will have a license file encoded with the name of your company that must match on the source code by setting the SiteLicense property. See this example, imagine that your company is called MITData and you just purchased a site license this is what you should change on your code:

Example in ASP

```
<%  
set PDF = server.createobject  
("aspPDF.EasyPDF")  
PDF.SiteLicense = " MITData "  
' Loads the license file  
PDF.License("easypdf.lic")  
response.write "<br>Version Information:<br>" &  
PDF.Version  
set pdf = nothing  
%>
```

Notice that you must set the SiteLicense property after creating the library object and before calling the License function.

The license won't check any IP address or Unique Identification of the machine, just if the name matches the license.

Using a compiled DLL:

This is useful if you don't want that the library does not check any license file, you must contact us to compile a special version that will have encoded the company name inside the Library.



The bad part of this option is that if you want to upgrade the library to the latest version you must contact us, with the first option you only download the latest version from the registered users site.

See also

 [License](#)  [Lic_Debug](#)

WEB_APP

Property explain

Name	Web_App
Syntax in VB	<i>PDF . Web_App = True / False</i>
Parameter type	Boolean
Read / Write	 / 

Sets the library to a running server environment, like an Web server or a batch process which doesn't need any user interaction.

Note: Is very important to set this property accordantly to your application environment, this will prevent further error on your application.

When you got an error while working with the library and the WEB_APP is set to false, then you will get a popup message with the error. This may be an inconvenient for web server application where you want a user to give ok to all error messages that appears on the server.

Also for batch process where you don't want to have any message appearing in your application control, set the property to true, then on your application check always the error property to see if there was an error on the last command.

Brief aspEasyPDF functions description by Groups



Name	Description
Internal	
cnvUnitInch	Returns the adobe unit measurement from an inch unit.
cnvUnitMM	Returns the adobe unit measurement from a millimeter unit.
Create	Forces the creation of all initialization variables of aspEasyPDF. Is automatically called, just override if necessary.
Destroy	Forces the destruction of all pointer memories. Is automatically called, just override if necessary.
GetProperty	Gets an internal property - See constants to see which property can return information.
GetPropObj	Gets an internal property for an object set - See constants to see which object property affects
SetProperty	Sets an internal property - See constants to see which property affects the correct objects
SetPropObj	Sets an internal property for an object set - See constants to see which object property affects
Charts	
AddChart	Starts defining a chart to be drawn later
AddChartSeries	Adds a chart serie to the current chart definition
AddChartXYValue	Adds a value to the chart series for X and Y
AddCharXValue	Adds a value to the chart series for X
AddChartYValue	Adds a value to the chart series for Y
DrawChart	Draws the chart to the current page
DrawChartPos	Draws the chart to the current page and the specified position
Document	
AddOutline	Adds an outline / bookmark reference to the document.
AddPage	Adds a new page to the document
AddPattern	Defines a pattern to be used when generating the document, the contents are in HTML and it can be defined which pages affects.
AddPDF	Imports a new PDF document into the current document.
AddProtection	Sets the protection/security of the document.
MoveToPage	Moves to another page in the document, making it current.
Page	Defines a standard page size for the current page and the new ones.
PageSize	Defines a user page size for the current page and the new ones.
SetMargins	Sets the margins for the current page and for the new pages
SetEvenMargins	Sets the margins for the current page and for the new pages that are even.
SetPos	Sets the current cursor positions
Fonts	
AddFont	Adds a new TTF font in the document to be used later.
SetFont	Sets the current font to be used.
Forms	

AddFDFValue	Sets a value for an object form. Use it in combination of SaveFDF or BinaryWriteFDF
AddFormObj	Adds a new form object to the current page
AddEventObj	Adds an event script to the form object
Graphic	
AddBarCode	Draws a barcode to the current page cursors
AddBox	Draws a box to the specified position and size to the current page.
AddBoxRounded	Draws a rounded box to the specified position and size to the current page.
AddEllipse	Draws a circle or an ellipse to the current page
AddGraphic	Loads an image to the current page cursors
AddGraphicPos	Loads an image to the current page at the specified cursors.
AddLine	Draws a line to the specified position and size at the current page.
Reports (Enterprise version which includes easyReportPDF)	
AddDBBand	Adds a band to the report
AddDBBarCode	Adds a field connection which is a barcode to the report
AddDBConnection	Adds a connection definition
AddDBText	Adds a text field to the report
AddDBGraphic	Adds a graphic field to the report
AddParameter	Adds an user parameter
AddScript	Adds script code to the report
AddTextField	Adds a text which can be manipulate trough the script
RenderReport	Renders the report, creates all pages but does not save it.
SetDBBandActive	Sets the active band to add all linked functions and properties
SetDBConnection	Changes the connection string
SetDBSQL	Set the SQL for the given connection
SetReportPage	States aspEasyPDF to use the actual page as a report page
SetParameter	Sets the parameters value for a given parameter.
<u>Saving or generating PDF</u>	
BinaryWrite	Sends the generated document to the client Browser. Specific for ASP - IIS programmers.
BinaryWriteFDF	Send the FDF to the client Browser. Specific for ASP - IIS programmers.
Save	Saves the document into a file.
SaveFDF	Saves FDF variables into a file.
SaveStream	Saves the document into a stream memory.
SaveString	Returns the document in string argument (not binary allowed).
SaveVariant	Returns a pointer structure to be managed in .NET applications. (File saving or being streamed to the client)
Tables	

AddTable	Adds a new Table to report to be filled.
AddRow	Adds a new row to the table.
SetCell	Fills the cell with the contents specified.
SetCellHTML	Fills the cell with HTML contents.
SetCellTable	Links another table to the cell (nested tables)
GetCell	Returns the cell contents.
DrawTable	Draws the table at the current cursor position
DrawTablePos	Draws the table at the specified position
<u>Text and HTML</u>	
AddNote	Add a note box to the current page
AddHTML	Add an html code to the current page cursors.
AddHTMLPos	Add an html code at the specific position to the current page.
AddHTML	Add an html code to the current page cursors.
AddLink	Adds a link to an internal position or an external PDF / URL.
AddTextPos	Add text at the specific position to the current page with the default font style and size.
AddTextWidth	Add text at formatted size and alignment text position
GetTextHeight	Returns the total height size of a given text with the current font
GetTextWidth	Returns the total width size of a given text with the current font
<u>VisualEasyPDF</u>	
DrawVEP	Draws the VisualEasyPDF that was imported with LoadVEPFile to the pages specified.
LoadVEPFile	Loads and import a VisualEasyPDF page

Create

Function explain

Name	Create
Parameters	 (none)
Result	 (none)
From version	1.1

The create function is designed for programming languages that does not use the OnStartPage method from the IIS, like VB, C++ or Delphi. This method will initialize the memory for use the EasyPDF.
Warning, do not try to call this method in ASP and even call twice or more times in other languages, if you do so you will get an exception.

Note: From version 2.0 there is no need to call Create, this is done automatically.

Syntax in VB

PDF.Create

Example in VBS

```
set moPDF = createObject("aspPDF.EASYPDF")
moPdf.Create
moPdf.SetPos 72, 72
moPdf.AddText " 1313 Mocking Bird Lane "
moPdf.save " e:\temp\envelope.pdf "
```

Example in Delphi



```
// Create the COM object
PDF := CreateComObject(CLASS_EASYPDF) as
IEASYPDF;
// Init the memory
PDF.Create();
PDF.SetPos( 72, 72 );
PDF.AddText(' 1313 Mocking Bird Lane' );
// Save it to a file
PDF.Save(' e:\temp\envelope.pdf ');
// Destroy object
PDF.Destroy();
```

See Also

 [Destroy](#)

Destroy

Function explain

Name	Destroy
Parameters	 (none)
Result	 (none)
From version	1.1
Until version	2.0

Note: From version 2.0 this function has been removed because it's called automatically when the object frees it self.

When you create the object you must destroy it when finishing working with, this will release all the memory that has take for the process when generating the document.

This is for programming languages that does not use the OnEndPage method inside, like VB, C++ or Delphi.

Warning, do no try to call this method in ASP and even call twice or more times in other languages, if you do so you will get an exception.

Syntax

PDF.Destroy

See Also

 [Create](#)

AddBarcode

Function explain

Name	AddBarcode
Parameters	✔ XPos as Double , YPos as Double , Height as Double , Type as Integer , TextToCode as String
Result	✘ (none)
From version	1.0



" Professional feature "

Allows you to draw a Bar code on the PDF document. It has the same code bars support from the aspEasyCodeBar. The width will be adjusted automatically with the visible size of the code bar, the Zoom properties from the graphics will be used, so 50% will reduce on the half the size of the image, default is 100%.

Different types to use for coding bars:

```

const csCode_2_5_interleaved = 0
const csCode_2_5_industrial = 1
const csCode_2_5_matrix = 2
const csCode39 = 3
const csCode39Extended = 4
const csCode128A = 5
const csCode128B = 6
const csCode128C = 7
const csCode93 = 8
const csCode93Extended = 9
const csCodeMSI = 10
const csCodePostNet = 11
const csCodeCodabar = 12
const csCodeEAN8 = 13
const csCodeEAN13 = 14
const csCodeUPC_A = 15
const csCodeUPC_E0 = 16
const csCodeUPC_E1 = 17
const csCodeUPC_Supp2 = 18
const csCodeUPC_Supp5 = 19
const csCodeEAN128A = 20
const csCodeEAN128B = 21
const csCodeEAN128C = 22

```

' From 3.0 version

```

const csCode_2_5_datalogic = 23
const csCode_2_5_IATA = 24
const csCode_2_5_Invert = 25
const csCode_2_5_Coop = 26
const csCodeABCCodabar = 27
const csCodeITF = 28
const csCodeISBN = 29
const csCodeISSN = 30
const csCodeISMN = 31
const csCodeOPC = 32
const csCode11 = 33
const csCodePZN = 34
const csCodePDF417 = 35

```

Syntax

PDF.AddBarCode XPos as Double , YPos as Double , Height as Double , Type as Integer , TextToCode as String

Example

```
<%  
' Define csCodeEAN128A constants  
const csCodeEAN128A = 20  
' Create the component  
set PDF = server.createObject(" aspPDF.EasyPDF ")  
' Sets  
PDF.AddBarCode PDF.cnvUnitmm(10), PDF.cnvUnitmm(10), 50, csCodeEAN128A, " 540010"  
PDF.BinaryWrite  
' destroy it  
set pdf = nothing  
%>
```

See also

 [csPropGraphBCAngle](#)

 [csPropGraphBCRatio](#)

 [csPropGraphBCText](#)

 [GetBarCodeWidth](#)

AddBox

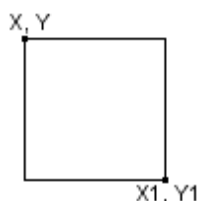
Function explain

Name	AddBox
Parameters	✔ X as Double , Y as Double, X1 as Double , Y1 as Double
Result	✘ (none)
From version	1.0

Draws a Box on the coordinates of the actual page, see properties that affect the appearance of the [shadow](#) , [colors](#) and fill colors.

To set the filled color use the constant  [csPropGraphFillColor](#) with the  [csPropGraphFilled](#) to indicate that the box is filled, if you don't set  [csPropGraphFilled](#) the box will be transparent.

X,Y is the first coordinate point and X1,Y1 is the second coordinate point, you can not specify the height and width of the graphic but you can do a small function to calculate it. See example.



Syntax

PDF .AddBox X, Y, X1, Y1

Example

```
<%
Sub DrawBox( X, Y, Width, Height )

    PDF.AddBox X, Y, X + Width, Y + Height

End Sub

' Create the component
set PDF = server.createObject(" aspPDF.EasyPDF ")
' Sets
DrawBox 10,10, 50, 50
PDF.BinaryWrite
' destroy it
set pdf = nothing
%>
```

See Also

 [AddBoxRounded](#)  [AddLine](#)  [AddEllipse](#)
 [csPropGraphFillColor](#)  [csPropGraphFilled](#)  [csPropGraphLineColor](#)  [csPropGraphDashLine](#) 
[csPropGraphWL](#)

AddBoxRounded

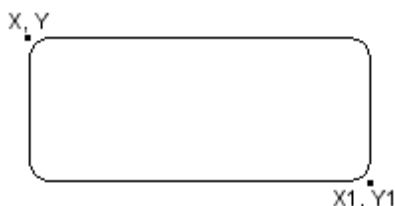
Function explain

Name	AddBoxRounded
Parameters	✔ X as Double , Y as Double, X1 as Double , Y1 as Double , Size as Double
Result	✘ (none)
From version	🕒 2.00

Draws a rounded Box on the coordinates of the actual page, see properties that affect the appearance of the [shadow](#) , [colors](#) and fill colors.

To set the filled color use the constant [csPropGraphFillColor](#) with the [csPropGraphFilled](#) to indicate that the box is filled, if you don't set [csPropGraphFilled](#) the box will be transparent.

X,Y is the first coordinate point and X1,Y1 is the second coordinate point the size specifies the rounded corner size.



Syntax

PDF.AddBoxRounded X, Y, X1, Y1, Size

Example in ASP

```
<%
' Create the component
set PDF = server.createObject(" aspPDF.EasyPDF
")
PDF.AddBoxRounded 10,10, 50, 50, 2
PDF.BinaryWrite
set pdf = nothing
%>
```

See Also

- [👉 AddBox](#)
- [👉 AddLine](#)
- [👉 AddEllipse](#)
- [🔑 csPropGraphFillColor](#)
- [🔑 csPropGraphFilled](#)
- [🔑 csPropGraphLineColor](#)
- [🔑 csPropGraphDashLine](#)
- [csPropGraphWL](#)

AddChart

Function explain

Name	AddChart
Parameters	✔ Name as String , Properties as String
Result	✔ Chart as String
From version	3.0

Initializates the chart with its properties definition (see constants) and returns the chart to use for drawing.

Syntax

Chart = PDF.AddChart Name, Properties

Properties definition

 [All Chart constants definition](#)

See Also

 [AddChartSeries](#)  [AddChartYValue](#)  [DrawChartPos](#)

Function explain	
Name	AddChartSeries
Parameters	✔ Chart as String , Properties as String , SeriesType as Integer
Result	✔ Serie as Integer
From version	3.0

Adds a serie to the chart, in PRO version you can add several series for the same chart. Standard version can only add one serie.

Series Type	
Series	Type
0	Line
1	Fast Lines
2	Vertical Bars
3	Horizontal Bars
4	Areas
5	Pie
6	Bubbles
7	Gantt

Syntax

Serie = PDF.AddChartSeries Chart, Properties, SeriesType

Function explain

Name	AddChartXYValue
Parameters	✔ Chart as String , Serie as Integer , X as Double , Y as Double ,Title as String , Color as String
Result	✔ Serie as Integer
From version	3.0

Adds a value to the series chart.

Syntax

PDF .AddChartXYValue Chart, Serie, X, Y, Title, Color

Function explain

Name	AddChartXValue
Parameters	✔ Chart as String , Serie as Integer , X as Double , Title as String , Color as String
Result	✔ Serie as Integer
From version	3.0

Adds a value to the series chart.

Syntax

PDF.AddChartXValue Chart, Serie, X, Title, Color

Function explain

Name	AddChartYValue
Parameters	✔ Chart as String , Serie as Integer , Y as Double ,Title as String , Color as String
Result	✔ Serie as Integer
From version	3.0

Adds a value to the series chart.

Syntax

PDF .AddChartYValue Chart, Serie, Y, Title, Color

AddFont	
Function explain	
Name	AddFont
Parameters	✔ FontName as String , Font as String , Encoding as String , Parameters as Integer , Embedded as Integer
Result	✔ Boolean
From version	2 2.00

Will add more fonts to the PDF document. PDF documents supports 14 standard fonts (see a list of supported fonts on [SetFont](#) instruction), if you need to add more fonts or use Unicode support, then you should use this function to add manually fonts into the document.

Enables you to add more fonts to the PDF document, the PDF comes with 14 standard fonts and this may be some users requires to use additional fonts and most common the Windows True Type font. This options allows you to add a True type and CJK fonts on the document and use it.

Note:This function works only for professional if you use TTF fonts, standard version can only use CJK fonts.

The font is just linked to the document, so you must to take care that on the client side has the same font, if not the PDF will replace it with standard F1 - Helvetica font.

The **FontName** is the internal font name to use in PDF, you can use any name but from **F1 to F14**, are reserved to internal fonts.

The **Font** is the font name to load on the document, if you use CJK fonts you should specify the exact font name, see Encoding table for a list of available fonts.

The **Encoding** parameter of the font to use.

Encodings		
Locale	Font Name	Supported Encodings
True Type Font	(all)	WinAnsi
True Type Font (unicode)	(all unicode)	Unicode Identity-H
Encodings CJK		
Simplified Chinese	STSong-Light STSongStd-Light-Acro	UniGB-UCS2-H UniGB-UCS2-V
Traditional Chinese	MHei-Medium MSung-Light MSungStd-Light-Acro	UniCNS-UCS2-H UniCNS-UCS2-V
Japanese	HeiseiKakuGo-W5 HeiseiMin-W3 KozMinPro-Regular-Acro	UniJIS-UCS2-H UniJIS-UCS2-V UniJIS-UCS2-HW-H UniJIS-UCS2-HW-V
Korean	HYGoThic-Medium HYSMyeongJo-Medium HYSMyeongJoStd-Medium-Acro	UniKS-UCS2-H UniKS-UCS2-V

The **Parameters** specifies the font style, to combine two or more properties just sum the values:

Fonts	
Number	Definition
0	None (Default)

1	Bold
2	Italic

The **Embedded** parameter means that the font will be embedded to the PDF document, making it cross platform and does not care if you have it install on all machines. This is quite important for use with special fonts or others international fonts that are sending it to different platforms that does have it installed on their machines, PDF will use the embedded font and make it readable.

When you embedded fonts the PDF Document will increase on size, between 120k to 500k, depends on the font to used.

Note: This feature is only available for the Professional version.

Embedded	
Number	Definition
0	None (Default)
1	Embedded

Syntax


PDF.AddFont FontName as string, TrueType as string, Parameters as Integer, Embedded as Integer

Example in ASP:

```
<%  
' Font Styles  
const fsNone = 0  
const fsBold = 1  
const fsItalic = 2  
' Create the component  
set PDF = server.createObject(" aspPDF.EasyPDF ")  
' Set the True Type font Tahoma to use it with the internal F15 name  
PDF.AddFont " F15 ", " Tahoma ", " WinAnsi ", fsNone, 0  
' Set the True Type font Tahoma in BOLD to use it with the internal F16 name and embedded in the document  
PDF.AddFont " F16 ", " Tahoma ", " WinAnsi ", fsBold, 1  
' Set the font  
PDF.SetFont " F15 ", 16, ""  
' Add a text in Tahoma style  
PDF.AddText " This is a text with Tahoma font<br> "  
PDF.SetFont " F16 ", 16, ""  
PDF.AddText " This is a text with Tahoma font with a Bold style and the font is embedded to the  
document<br> "  
' Destroy it  
setpdf = nothing  
%>
```

CJK Example in VB:


```
setPDF = createObject(" aspPDF.EasyPDF ")  
WScript.Echo PDF.Version  
PDF.SetMargins 20, 20, 20, 20  
PDF.AddFont " FntJapan ", " HeiseiKakuGo-W5 ", " UniJIS-UCS2-H ", fsNone, 0  
PDF.SetFont " FntJapan ", 20, " #000000 "  
PDF.AddText " This is a unicode string, here are some Japan chars: &#4ea0; and  
&#4ea4; "  
PDF.Save " unicode_cjk.pdf "
```


See also
 [SetFont](#)

AddFormObj

Function explain

Name	AddFormObj
Parameters	✔ X as Double , Y as Double , X1 as Double , Y1 as Double , Name as WideString , Caption as WideString , Hint as WideString , Type as Integer
Result	✔ Boolean
From version	2 2.00


" Professional feature "

Adds an object forms to the document, you can add Labels, Input Boxes, Combo and Check boxes, all can easily interact with each other with JavaScript.

X, Y is the first coordinate point and X1, Y1 is the second coordinate point.

Name is the object name that must be unique, is mandatory to specify the form name and then the object name separated by a dot, for example *form.object*

Caption is the title that will appear on the document if the object supports captions.

Hint is the help that appears when the cursor passes over the object.

Type is the object type that you want to add on the actual page:

Type Object

Number	name	Description
0	foButton	Adds a button
1	foCheckBox	Adds a check mark To control if the check box is marked or not, set the caption to Off to be unmarked and a different value to mark it by default
2	foRadioButton	Adds a radio button options. To add different values on the radio button you must use the name object on this way: form.radioobject.value1 and the next one must be form.radioobject.value2, which means that values will be inserting on the form.radioobject On the caption you set if you want the option to be marked or not, if you set it to Off then it will not marked, to be marked insert the same value as the name value.
3	foTextField	Adds a text field
4	foScrollList	Adds a scroll list Note: Beta version does not draw correctly the scroll list
5	foComboBox	Adds a combo box

Syntax

PDF.AddFormObj X, Y, X1, Y1, Name, Caption, Hint, Type

Example in ASP

```

<%
' Create the component
set PDF = server.createobject(" aspPDF.EasyPDF ")
' Constants
const foButton = 0
const foCheckBox = 1
const foRadioButton = 2
    
```


```
const foTextField = 3
const foScrollList = 4
const foComboBox = 5

' Add button that will send the results to an internet page
PDF.AddFormObj 120, 100, 220, 150, " form.buton ", " push me ", " Press me ", foButton
' Adds a text field editable with a default value
PDF.AddFormObj 120, 300, 220, 315, " form.editme ", " My Value ", " Introduce the value ", foTextField
' Adds a combo box and fulfill the data with SetPropObj function
PDF.AddFormObj 320, 600, 420, 615, " form.combos ", " Combo value ", " Select a credit card type ", foComboBox
PDF.SetPropObj " form.combos ", csPropObjCbxValues, "[ (AMX)(American Express)] [ (CBL)(Carte Blanche)] [ (DCL)(Diners Club)] [ (DSC)(Discover)] [ (ENR)(EnRoute)] (JCB)[ (MSC)(MasterCard)] [ (VIS)(Visa)] "
' Adds a Scroll list and introduces the values
PDF.AddFormObj 320, 400, 420, 450, " form.listbox ", " test ", " Introduce the value combo value ", foScrollList
PDF.SetPropObj " form.listbox ", csPropObjCbxValues, " [ (AMX)(American Express)] [ (CBL)(Carte Blanche)] [ (DCL)(Diners Club)] [ (DSC)(Discover)] [ (ENR)(EnRoute)] (JCB)[ (MSC)(MasterCard)] [ (VIS)(Visa)] "
' Adds a radio button
PDF.AddFormObj 150,700,170,720, " form.radiobutton.option1 ", " option1 ", " Mark ", foRadioButton
PDF.AddFormObj 250, 700, 270, 720, " form.radiobutton.option2 ", " Off ", " Mark me ", foRadioButton
' Adds a check box
PDF.AddFormObj 450,700,470,720, " form.checkbox ", " Off ", " Mark it ", foCheckBox
' Adds an event
PDF.AddEventObj aaOnMouseUp, " form.buton ", "
SubmitForm('http://test/pdf_print_vars.asp','form.checkbox','form.editme') ", " A "
PDF.BinaryWrite
set pdf = nothing
%>
```

See Also

 [AddEventObj](#)

AddFDFValue

Function explain	
Name	AddFDFValue
Parameters	✔ Name as String , Value as String
Result	✘ (none)
From version	2 2.00
 " Professional feature "	

Adds form data field object to the FDF document (Form Data Format).

This is very useful to change fields on a PDF document, for example, you may have a PDF documents that have some input fields and you want to use it as template, the user introduces the name, this is added to the FDF document and it's saved. When it opens the FDF document it will replace the objects by the value specified.

This is a powerful function that gives you the option to use PDF as templates and fulfill the form objects with the values you specify on the FDF file.

Note:FDF files is only a list of form object values that must match with the one that is specified on the PDF template document that is defined when saving it.

Syntax

PDF .AddFDFValue Name, Value

Example in ASP

First create the PDF template file and save it:

```
<%
' Create the PDF document
set PDF = server.createobject(" aspPDF.EasyPDF ")
' Add a field text
PDF.AddFormObj 120, 400, 220, 415, " frm.edtName ", "", " Introduce your name ",
foTextField
PDF.AddTextPos 120, 380, " Introduce your name: "
PDF.Save " names.pdf "
set pdf = nothing
%>
```

Now get the name of the user and create a FDF document linked to names.pdf :

```
<%
' Create the PDF document
set PDF = server.createobject(" aspPDF.EasyPDF
")
' Add a field text
PDF.AddFDFValue " frm.edtName ", request
("name")
PDF.SaveFDF " setnames.fdf" , " names.pdf "
' You can also use the binarywrite from ASP
' PDF.BinaryWriteFDF "http://xxxx/names.pdf"
set pdf = nothing
%>
```

If you double click the setnames.fdf file it should open the names.pdf and fulfill the data with the contents of

the FDF file.

See Also

 [SaveFDF](#)

 [BinaryWriteFDF](#)

AddGraphic

Function explain

Name	AddGraphic
Parameters	✔ Filename as String
Result	✔ Boolean
From version	1.0

Adds the graphics at the last position were you added the text.

Supported formats:

TYPE	Max. Resolutions
BMP	2,16,256 or 16M colors
GIF	x - 8 bits (Maximum set by the GIF specification)
JPEG	JPEG (RGB, GrayScale, YCbCr, CMYK, YCbCrK) 24 bits
JPEG200	JPEG2000: JP2, J2K and JPC code stream formats (JPEG-2000 Part-1 standard, ISO/IEC 15444-1)
PCX	2,16,256 or 16M colors
PNG	with various compression levels.
TIFF	TIFF (rev.6.0 and Tech.Note #2, Packbits, JPEG, LZW, CCITT G.3 and G.4) with RGB, CMYK, B/W, CIELab color spaces to file or stream. Also FAX (CCITT3), G3F and G3N (Zetafax) formats supported for loading. Supported 4, 8, 16, 32, 64, 128 and 256 colors paletted images
Others	DIB, RLE, TGA (TARGA, VDA, ICB, VST, PIX)
Portable Bitmap	PBM, PGM and PPM
Vector	WMF, EMF
Windows	ICO and CUR

Note between different versions: On the professional you can retrieve remote graphics from other servers, on the standard version you can only get local graphics. Remember that using http connections for getting files will reduce the speed when creating the PDF document, please try to use local files to avoid this problem.

Syntax

Result = *PDF*.AddGraphic FileName as String

Notes

The bitmaps are being compressed in the PDF document with LZ compression level.

Adding a lot of graphics will increase the document size and will decrease the performance when generating dynamically the PDF document.

When inserting graphic the component uses 72dpi, which is a bad quality but good for internet connection because file is smaller.

To came across with a better image quality you should use the zoom factor:

100% --- > 72 dpi (normal quality)

50% --- > 144 dpi (good)

25% --- > 288 dpi (very good)

Examples

```
<% ' Create the component
set PDF = server.createobject
("aspPDF.EasyPDF")
' Adds a graphic on the page
```

```
PDF.AddText " My first graphic on a PDF<br> "  
PDF.AddGraphic " C:\inetpub\images\test.jpg "  
PDF.BinaryWrite  
' destroy it  
setpdf = nothing  
%>
```

Only for professional version

```
<!--#include file ="easypdf.inc"-->  
<% ' Create the component  
set PDF = server.createObject("aspPDF.EasyPDF")  
' Adds a graphic on the page  
PDF.AddText " The Google logo:<br> "  
' Sets the proxy if we have a proxy server  
PDF.SetProperty csPropIntProxyServer, " 192.1.1.1:80 "  
PDF.AddGraphic "  
http://www.google.com/images/logo.gif "  
PDF.BinaryWrite  
' destroy it  
setpdf = nothing  
%>
```

See Also


 [AddGraphicPos](#)

 [csPropGraphZoom](#)

 [csPropGraphWZoom](#)

 [csPropGraphHZoom](#)

 [csPropGraphJPGQuality](#)

 [csPropGraphImageIndex](#)

AddGraphicPos

Function explain

Name	AddGraphicPos
Parameters	✔ X as Double , Y as Double , Filename as String
Result	✔ Boolean
From version	1.0

Draws an image to the x, y coordinate on the actual page.

See full information on [AddGraphic](#) function along with some samples.

Syntax

PDF.AddGraphicPos X, Y, FileName

See Also

[AddGraphic](#)

[csPropGraphZoom](#)

[csPropGraphWZoom](#)


[csPropGraphHZoom](#)

AddEllipse

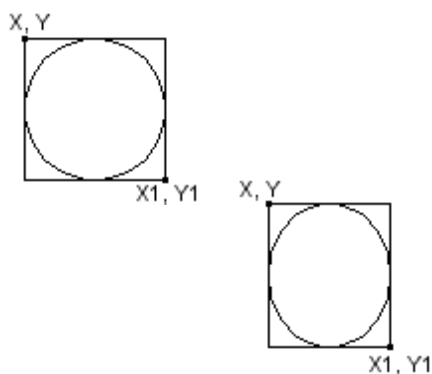
Function explain

Name	AddEllipse
Parameters	✔ X as Double , Y as Double, X1 as Double , Y1 as Double
Result	✘ (none)
From version	📌 2.00

Draws an Ellipse / Circle on the coordinates of the actual page, see properties that affect the appearance of the [shadow](#) , [colors](#) and fill colors.

To set the filled color use the constant  [csPropGraphFillColor](#) with the  [csPropGraphFilled](#) to indicate that the ellipse is filled, if you don't set  [csPropGraphFilled](#) the ellipse will be transparent.

X,Y is the first coordinate point and X1,Y1 is the second coordinate point, you can not specify the height and width of the graphic but you can do a small function to calculate it. See example.



Syntax


PDF.AddEllipse X, Y, X1, Y1

Example in ASP

```
<%
' Create the component
set PDF = server.createObject(" aspPDF.EasyPDF
")
PDF.AddEllipse 10,10, 50, 50
PDF.BinaryWrite
set pdf = nothing
%>
```

See Also

-  [AddLine](#)
-  [AddBox](#)
-  [AddBoxRounded](#)
-  [csPropGraphFillColor](#)
-  [csPropGraphFilled](#)
-  [csPropGraphLineColor](#)
-  [csPropGraphDashLine](#)
- 
- [csPropGraphWL](#)

AddEventObj	
Function explain	
Name	AddEventObj
Parameters	✔ EventType as Integer , Name as String, Script as String , ScriptType as String
Result	✔ Boolean
From version	2 2.00
 " Professional feature "	

Adds an event to an object, the object must have been created before adding the event.

Event Type		
Number	name	Description; when is called?
0	aaOnMouseIn	When the mouse enters on the object zone
1	aaOnMouseOut	When then mouse leaves the object zone
2	aaOnMouseDown	When the user press the mouse button
3	aaOnMouseUp	When the user release the mouse button
4	aaOnFocusEnter	When the user focus the object (TAB key)
5	aaOnFocusExit	When the user leaves the object (this is the best event to perform actions)
6	aaOnVisible	When the object is visible to client
7	aaOnUnVisible	When the object is being hidden (moving from one page to another)
8	aaOnKeyPress	When the user press a key on the object
9	aaOnDrawField	When the object is being drawn
10	aaOnValueChange	When the object has been changed

The function has been programmed in a way that it can be add more useful script languages or even types, at the moment there are two languages supported; JavaScript and aspEasyPDF Script.

JavaScript

It fully supports the JavaScript, please read the Adobe® documentation to know how to use the language in combination of a PDF document. You should download the Acrobat JavaScript document from adobe.

Some samples:

```
PDF.AddEventObj aaOnMouseUp, " form.buton ", " app.alert({cMsg: ""Error! Try again!"" ,cTitle: ""Acme Testing Service""}); ", " JavaScript ";
```

```
PDF.AddEventObjaaOnMouseUp, " form.buton ", " this.getField("""form.editme""").value=""MITData, S.C.P."""; ", " J ";
```

aspEasyPDF

This is combination of Acrobat® and JavaScript language in an easy language to interact with the document. The best part is that is very easy to remember and use, and the worse is that is quite limited and it only accepts one line. JavaScript can how big you want but the aspEasyPDF script is just one instruction. (This will change on future version)

Some samples:

```
PDF.AddEventObj aaOnMouseUp, " form.buton ", " SubmitForm("http://127.0.0.1/pdf_print_vars.asp",  
"form.checkbox", "form.editme") ", " aspEasyPDF Script ";
```

```
PDF.AddEventObjaaOnMouseUp, " form.buton ", " Hide("form.checkbox", true) ", " A ";
```

Funtions explanations:

FINDDIALOG

Show the Find Dialog.

Syntax: FindDialog()

HIDE

Hides or Shows an object

Syntax: Hide(objectname, booleanvalue)

Sample: Hide(" frm.UserName ", false)

MOVETO

Moves to from the current location to a different place, the values can be:

FirstPage
NextPage
PrevPage
LastPage
GoBack
GoForward

Syntax: MoveTo(wherestring)

Sample: MoveTo(" FirstPage ")

PRINT

Show the Print Dialog. For security reasons Acrobat® will never print automatically a document.

Syntax: Print()

RESETFORM

Set the default values of all objects

Syntax: ResetForm()

SAVEAS

Show the SaveAs Dialog.

Syntax: SaveAs()

SUBMITFORM

Submits the objects forms values to an URL to be processed. You must specify the URL address and the list of all objects to be send on the POST command.

Syntax: SubmitForm(URLaddress, list of form names)

Sample: SubmitForm("http://127.0.0.1/pdf_print_vars.asp", "form.checkbox", "form.editme")

ZOOM

Zooms the page, the values can be:

FitPage
FitVisible
ZoomViewIn
ZoomViewOut

Syntax: Zoom(zoomstring)


Sample: Zoom(" FitPage ")

See Also

 [AddFormObj](#)

NOTICE: Adobe® Acrobat® is copyrighted of Adobe Systems Incorporated.

AddHTML

Function explain	
Name	AddHTML
Parameters	✔ HTMLCode as String
Result	✘ (none)
From version	1.10
 " Professional feature "	

This is the most powerful option from the Pro version, it enables you to read HTML language and render it directly on the PDF without using concatenated methods to make the same draw. We have seen codes reduced from the standard version to the PRO version in 60% to 80% less sentences as the HTML offers you to render it directly with just one sentence.

[See supported HTML tags](#)

The parameter can be a string containing html sentences, a file or an url that retrieves remote information and the it process it. Processing URL is quite slow as it has to get it trough the network but you can linked to an ASP page that produces and output and the PDF interprets it to render it to PDF.

Some notes, you can not use the [AddHTML](#) method as the [AddText](#) as it works a little different. [AddText](#) and [AddHTML](#) writes directly to the document without knowing what will be on the next sentence, this makes a powerful way on working on fast a real docs. But what about tables on html?, you can not use the [AddHTML](#) to work in different sentence, you must pass all the table contents on a variable and then to the [AddHTML](#) .

Syntax

PDF.AddHTML Html as String

Example

```
<%
' Create the component
set PDF = server.createObject(" aspPDF.EasyPDF ")
' Adds html from an URL
PDF.AddHTML " http://www.mysite.com/asp/clauses.asp?data=10
"
PDF.BinaryWrite
' destroy it
setpdf = nothing
%>
```

Tables example

```
<%
' Create the component
set PDF = server.createObject(" aspPDF.EasyPDF ")
' Do not use different addHtml sentence to compose the table, just use a variable
s = " <table width=""100%""> "
s = s & " <tr> "
s = s & " <td>Mi field 1</td><td>Mi result</td> "
s = s & " </tr> "
s = s & " </table> "
PDF.AddHTML s
```

PDF.BinaryWrite

' destroy it

setpdf = nothing

%>

See Also

 [AddHTMLPos](#)

 [csHTML_FontName](#)  [csHTML_FontSize](#)

AddHTMLPos

Function explain

Name	AddHTMLPos
Parameters	✔ X as Double , Y as Double , HTMLCode as String
Result	✘ (none)
From version	1.10



" Professional feature "

Same as the [AddHTML](#) but with cursor positioning.

Syntax

PDF.AddHTML XPos as Double , YPos as Double , Html as String

See Also

[AddHTML](#)

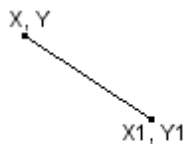
[csHTML_FontName](#) [csHTML_FontSize](#)

AddLine

Function explain

Name	AddLine
Parameters	✔ X as Double , Y as Double , X1 as Double , Y1 as Double
Result	✘ (none)
From version	1.0

Draws a line on the coordinates of the actual page.



Syntax

PDF.AddLine X, Y, X1, Y1

See Also

-  [AddBoxRounded](#)
-  [AddBox](#)
-  [AddEllipse](#)
-  [csPropGraphLineColor](#)
-  [csPropGraphDashLine](#)
-  [csPropGraphWL](#)

AddLink

Function explain

Name	AddLink
Parameters	✔ Text as WideString , Link as String , Type as Integer
Result	✘ (none)
From version	1.10



" Professional feature "

Link the document with internet page or email address that launches your favourite internet explorer or your email messenger, just add this sentence on the ASP and it will generate a link text, or an annotation commonly known in PDF.

It will add the link on the current position and will update the cursor at the end of the link. It uses the current font and also the paragraph property. If the test is wrapped in two lines, it will maintain the link for both lines.

You can specify the type link to use, you have 5 different choices:

Link Type		
Number	name	Description
0	InkToURL	means an internet link or url, like http://www.mitdata.com
1	InkToPDF	PDF file, notifies the document to open another PDF from a physical drive
2	InkDefineBookmark	Sets a bookmark name to jump with the identifier 3, the link property has no effect.
3	InkToBookmark	Jumps to the bookmark name specified in the link property. If it doesn't find the name it will jump at the beginning of the current page. The name is case sensitive.
4	InkToPosition	Jumps to page number with the X Pos and Y Position, the format to pass to the link is "PageNumber,X,Y", 3 parameters inside the link string separated with commas.

Syntax

PDF.AddLink TextToDisplay as string , Link as string , TypeLink as integer

Examples

```
<%
' Create the component
set PDF = server.createobject(" aspPDF.EasyPDF ")
' Set the font
PDF.SetFont "F1", 12, "#0000FF "
' Add a wonderful site ;-)
PDF.AddLink " Go To MitData ", " http://www.mitdata.com ", 0
PDF.Addtext"<br>"
' Jump to another PDF file on the local drive
PDF.AddLink " See our prices ", " Prices.PDF ", 1
PDF.Addtext "<br>"
' Sets this position as a bookmark on page 1g
PDF.AddLink "", " #Jumping ", 2
PDF.AddPage
' Link to jump to a bookmark
PDF.AddLink " Jump to page 1 ", " #Jumping ", 3
```

```
' Link to jump to a bookmark  
PDF.AddLink " Jump to page 1 at Beginnig of the page ", " 1,0,0 ",  
4  
PDF.BinaryWrite  
' Destroy and free memory allocated by the component  
set pdf = nothing  
%>
```

See Also

 [AddLinkPos](#)  [AddNote](#)

AddLinkPos

Function explain

Name	AddLinkPos
Parameters	✔ X as Double , Y as Double , Text as Wide String , Link as String , Type as Integer
Result	✘ (none)
From version	1.10



" Professional feature "

As the [AddLink](#) but with cursor position

Syntax

PDF.AddLinkPos XPos as double , YPos as double , TextToDisplay as string , Link as string , TypeLink as integer

See Also

[AddLink](#)

[AddNote](#)

AddNote

Function explain

Name	AddNote
Parameters	✔ X as Double , Y as Double , Title as WideString , Text as WideString , Date as String , Type as Integer
Result	✘ (none)
From version	2 2.00



" Professional feature "

Adds an icon note to the actual page.

The date must be inserted in string format composed by four digits year, 2 digits for the month and 2 more digits for the day, the total length must be 8, if not then date will no be printed.

Note Type		
Number	name	Description
0	noteIcon	Draws an Icon
1	noteCross	Draws a boxed cross
2	noteTextBoxed	Draws the text in a box

Syntax

PDF.AddNote XPos as double , YPos as double , Title as WideString , Text as WideString , Date as string , Type as integer

Example in ASP:

```
<%  
' Create the library  
set PDF = server.createObject(" aspPDF.EasyPDF ")  
' Adds a note  
PDF.AddNote 500, 200, " Title1 ", " This is simple note ", " 20030811", 0  
PDF.BinaryWrite  
' Destroy and free memory allocated by the library  
set pdf = nothing  
%>
```


See Also

[AddLink](#)

[AddLinkPos](#)

AddOutline

Function explain

Name	AddOutline
Parameters	✔ ReferredID as String , Text as WideString , Link as String , Type as Integer
Result	✔ ItemID as String
From version	2.00
 " Professional feature "	

Outlines are an hierarchical *tree line* showing the document s internal structure.

Each item in the outline can then be associated with a corresponding *destination* in the document, allowing the user to jump directly to that destination by clicking with the mouse.

When adding the first item being referred or not it will be added first and all the next items will be consecutive, there is no function to alter the order or to remove an item. After adding an item it will return a string identifier, use this a parameter to add an item inside this item making it the child of the Referred.

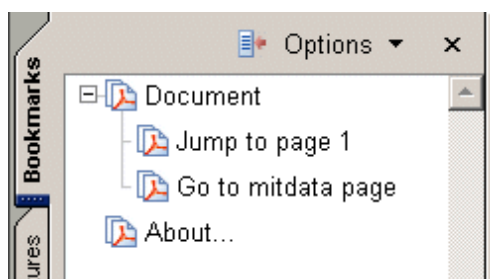
The Outline uses the same technique as the link function to make the destinations jumps.

Link Type		
Number	name	Description
0	InkToURL	means an internet link or url, like http://www.mitdata.com
1	InkToPDF	PDF file, notifies the document to open another PDF from a physical drive
2	InkDefineBookmark	Sets a bookmark name to jump with the identifier 3, the link property has no effect.
3	InkToBookmark	Jumps to the bookmark name specified in the link property. If it doesn't find the name it will jump at the beginning of the current page. The name is case sensitive.
4	InkToPosition	Jumps to page number with the X Pos and Y Position, the format to pass to the link is "PageNumber,X,Y", 3 parameters inside the link string separated with commas.

Syntax

PDF.AddOutline ReferredID as String , Text as WideString , Link as String , Type as integer

Example in ASP:



```

<%
' Create the library
set PDF = server.createObject(" aspPDF.EasyPDF ")
' Adds a node
main_node = PDF.AddOutline ("", " Document ", "", 3 )
PDF.AddOutline main_node, " Jump to page 1 ", " 1 ", 4
    
```



```
PDF.AddOutline main_node, " Go to MITData page ", " http://www.mitdata.com", 0
PDF.AddOutline "", " About ", " 1 ", 4
PDF.BinaryWrite
' Destroy and free memory allocated by the library
set pdf = nothing
%>
```

See Also

 [AddLink](#)  [AddLinkPos](#)
 [csPropObjOpened](#)

AddPage

Function explain

Name	AddPage
Parameters	 (none)
Result	 (none)
From version	1.0

Adds a new page to the document and position the cursor on the top of the document.
When you create the component it automatically creates the first page with the standard size DIN A4. (you can change it with the Page method).

When adding text paragraphs and reaching the end of the page it will add a new page automatically.

Syntax

PDF.AddPage

Example

```
<%  
' Create the component  
set PDF = server.createObject(" aspPDF.EasyPDF  
)  
' Add 10 pages  
for f = 0 to 10  
    PDF.AddText " This is the " & (f + 1) & " page. "  
    PDF.AddPage  
Next  
PDF.BinaryWrite  
' destroy it  
setpdf = nothing  
%>
```

See Also

 [SetReportPage](#)

AddPattern

Function explain

Name	AddPattern
Parameters	✔ Type as Integer , FromPage as Integer , ToPage as Integer , Offset as Double , HTML as String
Result	✘ (none)
From version	2 2.00



" Professional feature "

The use of pattern is to add on each page an HTML text. This function help to write headers and footers for the document. If you use the 0 value for the FromPage parameter then it will mean from the beginning and if you use it on the ToPage then it will mean to the end. So if you use 0,0 for the FromPage, ToPage parameter will mean to write on all pages.

The offset value is for altering the default position, you can use negative and positive values to up and down the text. HTML can be any valid HTML syntax.

You can use as many patterns as you want to construct the document, there is no limitation on the number of patterns.

Type

Number	name	Description
0	patHeader	Use the top of the page for writing the pattern
1	patFooter	Uses the footer of the page minus the margin of the actual page to position the text, you can use the Offset value to move the text.

Syntax


PDF.AddPattern Type as Integer , FromPage as Integer , ToPage as Integer , Offset as Double , HTML as String

Example

```
<%
' Create the component
set PDF = server.createObject(" aspPDF.EasyPDF ")
PDF.SetMargins 50,50,50,50
PDF.AddPage
PDF.AddPage
PDF.AddPage
' Add a Pattern from page 2 to the end
strHTML = " <p align="center">Page Nr: <font size=4><!--PDF.PAGENUMBER></font> of
<b><!--PDF.PAGECOUNT></b></p> "
PDF.AddPattern 1, 2, 0, 10, strHTML
PDF.BinaryWrite
' destroy it
setpdf = nothing
%>
```

See Also

[HTML syntax on aspEasyPDF](#)

Function explain	
Name	AddPDF
Parameters	✔ Filename as String , FromPage as Integer , ToPage as Integer , Password as String
Result	✔ Boolean
From version	2 2.10
 " Professional feature "	

Imports an existence PDF document and adds it's contents at the end of you new document. You can import as many times and pages combinations as you want.

The FromPage and ToPage vars are used to import a range of pages, if one of this var as a 0 then it's assumed as all. For example "FromPage is 4 and ToPage is 0" it will start importing from page 4 to the end of the document.

What will import:

1. Text contents of the page
2. Links (extenal)
3. Compressed documents
4. Graphics (8 - 24 bits)

Known limitations that will be improved on newer releases:

1. Password is ignored, it only imports unencrypted documents
2. Filename can only be phisical files
3. Support for Acrobat exchange and other 3rd party PDF exporters. (Colorspace command)

Syntax


PDF.AddPDF Filename as String, FromPage as Integer , ToPage as Integer, Password as String

Example

```
<%  
' Create the component  
set PDF = server.createobject(" aspPDF.EasyPDF  
")  
PDF.AddText "Importing PDF document"  
PDF.AddPage  
' Imports all pages from PDF the document  
PDF.AddPDF " c:\temp\doc.pdf ", 0,0, ""  
PDF.AddText "Imported"  
PDF.BinaryWrite  
' destroy it  
setpdf = nothing  
%>
```


AddProtection

Function explain

Name	AddProtection
Parameters	✔ OwnerPass as String , UserPass as String , Key_Length as Integer , Flags as Integer
Result	✘ (none)
From version	2 2.00
 " Professional feature "	

Adds security to the document. The document will be encrypted using the owner password, if it's left blank then it will use a random key and protect the document. On the flags you can specify what to protect and allow. The key length is encryption length that will use the system

Note: At this moment it only supports 40 bits on the future it will support 128 bits keys.

You can also set an user password to ask the user for a password.

Flags

Number	name	Description
4	scAllowPrint	Allows to print the document. If you want to let the user to print to the top quality use the scPrintQuality flag, if not then it will print a draft with low quality.
8	scModifyContents	Modify the contents of the document
16	scCopyContents	Copy or otherwise extract text and graphics from the document, including extracting text and graphics
32	scAllowNotesForms	Add or modify text annotations, fill in interactive form fields, and, if scModifyContents is also set, create or modify interactive form fields
256	scAllowForms	Fill in existing interactive form fields
512	scExtractText	Extract text and graphics
1024	scAssembleDoc	Assemble the document (insert, rotate, or delete pages and create bookmarks or thumbnail images), even if it is not set the scModifyContents
2048	scPrintQuality	Print the document to a representation from which a faithful digital copy of the PDF content could be generated.



Syntax


PDF.AddProtection OwnerPass as String, UserPass as String , Key_Length as Integer , Flags as Integer

Example

```
<%
' Create the component
set PDF = server.createobject(" aspPDF.EasyPDF ")
' Constant definition
const scAllowPrint = 4
const scModifyContents = 8
const scCopyContents = 16
const scAllowNotesForms= 32
const scAllowForms = 256
const scExtractText = 512
const scAssembleDoc = 1024
const scPrintQuality = 2048
' Use 40 bit encryption, set auto owner password and allow print and extract text to the clipboard
```

```
PDF.AddProtection "", "", 40, scAllowPrint+scPrintQuality+scExtractText
PDF.BinaryWrite
' destroy it
setpdf = nothing
%>
```

Function explain	
Name	AddRow
Parameters	 TableID as String , Property as String
Results	 (none)
From version	3 .10




Adds a new row to the table. Table ID is the returned value from  [AddTable](#) function and Property are the properties set for the row inserted.

Properties available for AddRow function		
Name	De scription	Default
Align	Alignment of the table	Left
BGColor	Background color for the Row, use HTML syntax	Transparent
Hieght	Fixed Size of the Row Size	Variable
Introduce the properties in the same syntax as you introduce the properties in an HTML tag.		


See Also

-  [AddTable](#)  [SetCell](#)  [SetCellHTML](#)  [SetCellTable](#)  [DrawTable](#)

Function explain	
Name	AddTable
Parameters	✔ Properties as String , Columns as Integer
Results	✔ Pointer as String
From version	3 .10

Initializes the memory for drawing a table structure in the PDF document; by using addTable it will not draw the table immediately, you need to use  [DrawTable](#) for that purpose. To fill the table with contents use the  [AddRow](#) and  [SetCell](#) functions.

AddTable function will return an identification number that should be used as a pass argument for the other table functions.

You should set the maximum number of columns that will have the table, rows are dynamically and you need to use the  [AddRow](#) function to add the rows for the table.

Properties available for AddTable function		
Name	De scription	Default
Align	Alignment of the table	Left
Border	Set the border size	1
BorderColor	Set the border color if it has a border size	Black
BGColor	Background color for the table	Transparent
CellSpacing	Space between each cell	0
CellPadding	Cell space for the text	0
Height	Fixed Size for the table height	Variable
Width	Fixed Size of the table, can be percent or pixel size	Variable
Introduce the properties in the same syntax as you introduce the properties in an HTML tag.		

Syntax

PDF .AddTable Properties as String, Columns as Integer

Example in ASP:

```
<%
' Create the component
set PDF = server.createobject(" aspPDF.EasyPDF
")
' table = PDF.AddTable "border=0 width='100%'", 2
PDF.SetCell table, 2,1, "", "HI"
PDF.DrawTable table
PDF.BinaryWrite
' destroy it
setpdf = nothing
%>
```

See Also

-  [AddRow](#)  [SetCell](#)  [SetCellHTML](#)  [SetCellTable](#)  [DrawTable](#)

AddText

Function explain

Name	AddText
Parameters	✔ Text as String
Results	✘ (none)
From version	1.00

Adds a new text on the document it starts at the top of the page and it will add consecutive text until it finds a carriage return or the internal command
, used on html to add a new line.

From version 2.0 you can load a full text file, (does not accept http requests), just add and the full path of the filename and it will load the text file and add it on the PDF.

You can set the way it looks setting some properties with the SetProperty Method, fonts, color, align, bottom start, left start and right end.

Syntax

PDF.AddText Text as String

Example in ASP:

```
<%  
MrName="Tony"  
' Add Big text on the document  
PDF.AddText "Hello Mr. " & MrName & "<br>We are very happy to announce that the new version of  
aspEasyPDF has been released to version 1.0.<br> <br>Please download it at www.mitdata.com<br>  
<br> Thank you very much, John "  
%>
```

See Also

 [AddTextPos](#)

 [AddTextWidth](#)



 [AddHTML](#)

AddTextPos

Function explain

Name	AddTextPos
Parameters	✔ X as Double , Y as Double , Text as String
Result	✘ (none)
From version	1.00

Adds a new text in the position specified on the active page. If you set the x to -1 value then you will center it to the page. All properties affect in the way that is displayed on the document.

You can position it to an inch or metric position with the  [cnvUnitMM](#) or  [cnvUnitInch](#) method

Syntax

PDF.AddTextPos X, Y, Text

See Also

 [AddText](#)  [AddTextWidth](#)  [AddHTML](#)  [cnvUnitMM](#)  [cnvUnitInch](#)

AddTextWidth

Function explain

Name	AddTextWidth
Parameters	✔ X as Double , Y as Double , Width as Double , Text as String
Result	✘ (none)
From version	1.50

Adds a new text in the position specified on the active page and using a maximum width size. This function is quite powerful because you can also specify the align method to use for the text, which such feature you can easily done columns aligns.

You can alter the aspect of the text with the `csPropAddTextWidth` constant property.

You can position it to an inch or metric position with the [cnvUnitMM](#) or [cnvUnitInch](#) method

Syntax

`PDF.AddTextWidth X as Double , Y as Double , Width as Double , Text as String`

Examples in ASP:

```

dim PDF
set PDF = server.createObject(" aspPDF.EasyPDF ")

const csPropAddTextWidth = 113
PDF.Debug = True
PDF.page " Letter ",0
PDF.setmargins 50,50,50,50
str = " This is a test to show the cut of a text with different options for the AddTextWidth
"
' Cut the text if it doesn't fit, you can also use the csPropTextAlign to align the text
PDF.SetProperty csPropAddTextWidth , 0
PDF.AddTextWidth 300,100,100, str
' Cut the text if it doesn't fit and add three suspensions points
PDF.SetProperty csPropAddTextWidth , 1
PDF.AddTextWidth 300,200,100, str
' Wrap the text down
PDF.SetProperty csPropAddTextWidth , 2
PDF.AddTextWidth 300,300,100, str
' Generate the PDF document
PDF.Save server.MapPath(" pdf_addtextwidth.pdf ")
set pdf = nothing

```

```

' At position 10,10 with a size of 200, the text is centered
PDF.SetProperty csPropTextAlign, algCenter
PDF.AddTextWidth 10,10,200, "Text Centered"
' This one is aligned to the right
PDF.SetProperty csPropTextAlign, algRight
PDF.AddTextWidth 10,60,200, "Right align"
' This aligned to the left
PDF.SetProperty csPropTextAlign, algLeft
PDF.AddTextWidth 10,120,200, "Left align"

```

See Also

 [csPropTextAlign](#)

 [AddText](#)

 [AddTextPos](#)



 [AddHTML](#)

 [cnvUnitMM](#)

 [cnvUnitInch](#)

BinaryWrite

Function explain

Name	BinaryWrite
Parameters	 (none)
Result	 (none)
From version	1.03

Warning: For using a similar function in .NET you should use the SaveVariant function and not the BinaryWrite

This is the same function as the Save method, but instead to save it to disk it will save it on memory and forwards it directly to the Internet Browser which will open the PDF reader. This is the preferred way to work on interned developments because you don't have to establish security permissions and you don't have to maintain PDF files.

Syntax

PDF.BinaryWrite

See also

 [Save](#)  [SaveStream](#)  [SaveString](#)  [SaveVariant](#)

BinaryWriteFDF

Function explain

Name	BinaryWriteFDF
Parameters	✔ PDFFileName as String
Result	✘ (none)
From version	2 2.00



" Professional feature "

Warning: This function **ONLY** works on ASP language for internet developments, does not work for ASP.NET

Forwards the FDF document to the user Internet explorer.

PDFFileName is the linked PDF document that will use it to fulfill the data. Please make sure the document is reachable, you can use http URLs.

Note: FDF files is only a list of form object values that must match with the one that is specified on the PDF template document that is defined when saving it.

Syntax

PDF.BinaryWrite PDFFileName as String

Example in ASP

See it on [AddFDFValue](#) sample

See Also

[SaveFDF](#) [AddFDFValue](#)

cnvUnitMM

Function explain

Name	cnvUnitMM
Parameters	✔ mmValue as Double
Result	✔ UserSpace as Double
From version	1.0

Converts a Millimeter number to an internal PDF number (*user space*).

To avoid the device-dependent effects of specifying objects in device space, PDF defines a device-independent coordinate system that always bears the same relationship to the current page, regardless of the output device on which printing or displaying will occur. This device-independent coordinate system is called *user space* .

The user space coordinate system is initialized to a default state for each page of a document. The positive x axis extends horizontally to the right and the positive y axis vertically downward. The length of a unit along both the x and y axes is 1D72 inch.

Syntax

vUnit = PDF.cnvUnitMM vMM as Double

Example

```
<%  
csx = PDF.cnvUnitmm( 10 )  
csy = PDF.cnvUnitmm( 20 )  
' Draws the graphic a 10 mm from the left page and 20 mm from the top page  
PDF.AddGraphicPos csx, csy, " C:\TEMP\MyImage1.gif "  
%>
```

See also

[c nvUnitInch](#)

cnvUnitInch

Function explain

Name	cnvUnitInch
Parameters	✔ Inch Value as Double
Result	✔ UserSpace as Double
From version	1.0

Converts an Inch number to an internal PDF number (*user space*).

To avoid the device-dependent effects of specifying objects in device space, PDF defines a device-independent coordinate system that always bears the same relationship to the current page, regardless of the output device on which printing or displaying will occur. This device-independent coordinate system is called *user space* .

The user space coordinate system is initialized to a default state for each page of a document. The positive x axis extends horizontally to the right and the positive y axis vertically downward. The length of a unit along both the x and y axes is 1D72 inch.

Syntax

vUnit = PDF.cnvUnitInch vMM as Double

Example



```
<%  
csx = PDF.cnvUnitInch( 3 )  
csy = PDF.cnvUnitInch(5 )  
' Adds a Text at 3 inches from the left page and 5 inch from the top page  
PDF.AddTextPos csx, csy, " Hello World "  
%>
```

See also

 [c nvUnitmm](#)

Clear Chart Values

Function explain

Name	Clear ChartValues
Parameters	 Chart as String, Series as Integer
Result	 (none)
From version	 3.00

Deletes all chart contents from the serie.

Syntax

PDF.ClearChartValues Chart, Series

See also

 [AddChart](#)  [DrawChartPos](#)

DrawChart

Function explain

Name	DrawChart
Parameters	✔ Chart as String Width, Height as Double
Result	✘ (none)
From version	2 3.00

Draws the chart that was defined on the document at current cursor position on the actual page document.

Syntax

PDF .DrawChartPos Chart, Width, Height

See also

✔ [AddChart](#) ✔ [DrawChartPos](#)

DrawChartPos

Function explain

Name	DrawChartPos
Parameters	✔ Chart as String X, Y, Width, Height as Double
Result	✘ (none)
From version	2 3.00

Draws the chart that was defined on the document at some position on the actual page document.

Syntax

PDF.DrawChartPos Chart, X, Y, Width, Height

See also

✔ [AddChart](#) ✔ [DrawChart](#)

Function explain	
Name	Draw Table
Parameters	✔ TableID as String
Results	✘ (none)
From version	3.10

Draws the table at the current position of the cursor and page.

Syntax

PDF.DrawTable TableID as String

Example in ASP:

```
<%  
' Create the component  
set PDF = server.createObject(" aspPDF.EasyPDF  
")  
' table = PDF.AddTable "border=0 width='100%'", 2  
PDF.DrawCell table, 2,1, "", "HI"  
PDF.DrawTable table  
PDF.BinaryWrite  
' destroy it  
setpdf = nothing  
%>
```

See Also

-  [AddRow](#)
-  [SetCell](#)
-  [AddTable](#)
-  [DrawTablePos](#)

Function explain	
Name	Draw Table
Parameters	✔ TableID as String , X as String , Y as String
Results	✘ (none)
From version	3.10

Draws the table at the position specified on X, Y of the current page.

Syntax

PDF.DrawTablePos TableID as String

Example in ASP:

```
<%  
' Create the component  
set PDF = server.createObject(" aspPDF.EasyPDF  
)  
' table = PDF.AddTable "border=0 width='100%", 2  
PDF.DrawCell table, 2,1, "", "HI"  
PDF.DrawTablePos table, 100, 100  
PDF.BinaryWrite  
' destroy it  
setpdf = nothing  
%>
```


See Also

-  [AddRow](#)
-  [SetCell](#)
-  [AddTable](#)
-  [DrawTable](#)

DrawVEP

Function explain

Name	DrawVEP
Parameters	✔ PageNR as Integer
Result	✘ (none)
From version	2 2.00

After loading a VEP form (VisualEasyPDF template file) you can choose to draw all pages or a specific page. To add all pages just use 0 for page number and it will insert the correlative pages automatically. If you use the Page Number the you should take care of the page dimension. See the  [LoadVEPFile](#) for sample.

Syntax

PDF.DrawVEP pageNr

See also

 [LoadVEPFile](#)  [SetPropObj](#)

GetBarcodeWidth

Function explain

Name	GetBarcodeWidth
Parameters	✔ BarType as Integer , BarValue as String
Result	✔ Width as Double
From version	1.60



" Professional feature "

Returns the width in PDF units of the barcode. This can be useful to know if the barcode will fit on the page.

Syntax

var = *PDF*.GetBarcodeWidth BarType as Integer , BarValue as String

See also

✔ [AddBarcode](#)

Function explain	
Name	GetCell
Parameters	✔ Table ID as String , Lin as String , Col as Integer
Result	✔ Value as String
From version	3.10

Returns the contents from the specified cell position.

GetProperty

Function explain

Name	GetProperty
Parameters	✔ PropertyID as Integer
Result	✔ Value as String
From version	1.0

Returns the specified property. The PropertyID is the parameter identifier that we want to know, it's an integer value and we normally uses constants or defines to easily use names in one include file.

Remember to add the include instruction if you want to use name identifiers, if not, then use directly the value for the property you want to get.

Properties are defined in constant groups; text, graphics, documents, information, report, internal, etc. We call each property with the constant name, you should find an information on each constant defined in this help file under the constants help tree.

Info: When we started to design the library, we searched an easy way to change parameters for modifying the look of the PDF document, without using thousands of variables that could change between different versions without losing compatibilities with old version. Some libraries uses properties that on version x a property is n wide length and for example a string type, then it came version 2 and this version uses a double precision variable, making the old code being incompatible.

This is why we decided to implement two powerful functions that gets and sets values that modifies and interacts with the library. The value is always treated as a string, making the conversion inside the library and forgetting on incompatible version from version 1 to x.

Syntax

var = *PDF*.GetProperty PropertyID as Integer

Properties

[See this link to read all properties that can be altered](#)

Example in ASP:


```
const csPropTextFont = 100
' Write with Helvetica Font
Actual_Font = PDF.GetProperty csPropTextFont
PDF.AddText " This font is " & Actual_Font
```

See also

 [SetProperty](#)  [SetPropObj](#)

SetPropObj

Function explain

Name	GetPropObj
Parameters	✔ ObjectID as WideString , PropertyID as Integer , Value as WideString
Result	✔ Result as WideString
From version	3.20
 " Professional feature "	

The GetProperty gets a global parameter, the GetPropObj gets an object parameter.

Syntax

Result = PDF.GetPropObj ObjectID as string, PropertyNumber as Integer

Properties

[See this link to read all properties that can be altered](#)

Example in ASP

```
<%  
' Create the component  
set PDF = server.createobject(" aspPDF.EasyPDF ")  
' Adds a Scroll list and introduces the values  
PDF.AddFormObj 320, 400, 420, 450, " form.listbox ", " test ", " Introduce the value combo value ", foScrollList  
PDF.SetPropObj " form.listbox ", csPropObjCbxValues, " [ (AMX)(American Express)] [ (CBL)(Carte Blanche)] [ (DCL)(Diners Club)] [ (DSC)(Discover)] [ (ENR)(EnRoute)] (JCB)[ (MSC)(MasterCard)] [ (VIS)(Visa)] "  
  
response.write PDF.GetPropObj " form.listbox ", csPropObjCbxValues  
PDF.BinaryWrite  
set pdf = nothing  
>%
```

See also

 [GetProperty](#)  [SetPropety](#)  [SetPropObj](#)  [DrawVEP](#)  [LoadVEPFile](#)

GetTextHeight From version 1.6

Function explain

Name	GetTextHeight
Parameters	✔ Text as WideString
Result	✔ Height as Double
From version	1.60

Returns the height in PDF units (*user space*) of the text passed as argument.

Syntax

var = *PDF*.GetTextHeight Text as String

Example

```
Text = " Where it will stop this text if it's so large, we need to know the height on a virtual 100px column wrap
"
' Place the text as you where printing it on the document
PDF.SetPos 100, 100
' Change the margins if you want a column look of 100px wide
PDF.SetMargin 100, PDF.GetProperty( csPageWidth ) - 100, 10, 10
pos = PDF.GetTextHeight ( Text )
' Draw a line where will end the text
PDF.AddLine 100, 100 + pos,200, 100 + pos
PDF.SetPos 100, 100
PDF.AddText Text
```

See also

✔ [GetTextWidth](#)

GetTextWidth

Function explain

Name	GetTextWidth
Parameters	✔ Text as WideString
Result	✔ Width as Double
From version	1.00

Returns the length in PDF units (*user space*) of the text passed as argument.

Syntax

var = *PDF*.GetTextWidth Text as String

See also

✔ [GetTextHeight](#)

License

Function explain	
Name	License
Parameters	✔ FileName as String
Result	✔ Result as Boolean
From version	1.00

NOTE: Only for registered customers.

After purchasing the library you will receive a license file that should be used in conjunction with the library. What it does the license is to check if the machine is allowed to run the library by checking the IP address or the Unique identification.

You do not have to use the license method if you copy the license on the system32 path, the library will search on the actual path and on the system32 path if it exists the license, then it will load it.

The license function is an alternative way to load the license from a different location of different license name, this could be useful for multi host environments or because you don't have read permissions on the system32 path.

Call the license function after creating the library process.

Syntax

PDF.License LicenseFile as String

Example in ASP:

```
<%
' Create the component
set PDF = server.createObject(" aspPDF.EasyPDF ")
' Loads the license from a different place
PDF.License " C:\inetpub\wwwroot\licenses\leasypdf.lic
"

' .... your code
%>
```

Example in Delphi:


```
var
  pdf : IEASYPDF;
begin
  // Create the COM object
  PDF := CreateComObject(CLASS_EASYPDF ) as IEASYPDF
;
  // Init the memory
  PDF.Create();
  // Load license
  PDF.License( ' C:\inetpub\wwwroot\licenses\leasypdf.lic ' );
  // .... your code
end;
```

See also

 [SiteLicense](#)  [Lic_Debug](#)

Function explain	
Name	LoadVEPFile
Parameters	✔ FileName as String
Result	✔ Result as Boolean
From version	🔢 2.00

Loads a VisualEasyPDF form file into memory, you will need a registered copy of VisualEasyPDF to allow you to save the designed forms, on the freeware edition you can not save and load forms.

What is really powerful of this option is that you can change any aspect of any object from the form before inserting it on the document. You draw the form by pages or by inserting all pages on your project and you can change it with the  [SetPropObj](#) function.

You can easily give to your customer the VisualEasyPDF to design their documents and from the library you can alter in real-time the data inside. For example you customer can draw a default invoice template and you can easily add and change it on your project.

Using VEP forms increases the speed when drawing it, if you use the source code that generates the VEP program this will take more time that loading directly the VEP form.

Syntax

result = *PDF*.LoadVEPFile(FileName)

Example in ASP:

```
<%  
' Create the component  
set PDF = server.createObject(" aspPDF.EasyPDF ")  
' Loads the VEP file ( Generate under VisualEasyPDF 1.01 )  
PDF.LoadVEPFile ( Server.MapPath(" pdf_VEPSample.vep " ) )  
' Before drawing change the text object contents  
PDF.SetPropObj " object6 ", csPropObjText, " **** Hello  
World!!!!!! " )  
' Draw all pages  
PDF.DrawVEP 0  
  
' .... your code  
%>
```

See also

 [DrawVEP](#)  [SetPropObj](#)

MoveToPage

Function explain

Name	MoveToPage
Parameters	✔ PageNumber as Integer
Result	✔ Result as Boolean
From version	1.40



" Professional feature "

Moves to a specific page. This is very useful to first draw all pages and after filling all data, you move to the first page to make the header and footer of each page moving one by one. The function will return true if the page has been moved or false if not.

This feature is new from version 1.4

Notes: Page must exists and the first page is 0 and not 1.

From version 2.0 you can use two direct properties to manipulate pages; [PageNumber](#) and [PageCount](#) properties.

Syntax

result = PDF.MoveToPage PageNr as Integer

Example in VB:

```
' After filling all you data in the document
' Now do the footers of every page

TotalPages = PDF.PageNumber
For P = 0 To TotalPages
  ' First page is 0 not 1
  PDF.MoveToPage P
  PDF.AddTextPos PDF.cnvUnitmm(10), PDF.cnvUnitmm(205), "Page " & P +1 & " of " & TotalPages +
  1
next
```

See also

[PageNumber](#) [PageCount](#)

Function explain

Name	SetCell
Parameters	✔ TableID as String , Lin as Integer , Col as Integer , Properties as String , Contents as String
Results	✘ (none)
From version	3 .10

Sets the contents of cell.

Note: To add more contents on the same cell, just begin the contents with the reserved word tag: <+> ; this will prevent to not overwrite the previous contents of the cell.

Properties available for AddRow function		
Name	De scription	Default
Align	Horizontal alignment of the cell	Left
BGColor	Background color for the Row, use HTML syntax	Transparent
Border	Size for the border cell	1
BorderColor	Color for the border if you have set border	Black
Width	Fixed Size of the Cell width	Variable
Valign	Vertical alignment of the cell	Top
Introduce the properties in the same syntax as you introduce the properties in an HTML tag.		

Syntax


PDF.AddCell TableID as String

Example in ASP:

```
<%
' Create the component
set PDF = server.createobject(" aspPDF.EasyPDF
")
' table = PDF.AddTable "border=0 width='100%'", 2
PDF.SetCell table, 2,1, "", "HI"
PDF.SetCell table, 2,1, "", "<+><br>BYE"
PDF.DrawTable table
PDF.BinaryWrite
' destroy it
setpdf = nothing
%>
```

See Also

-  [AddRow](#)
-  [SetCellHTML](#)
-  [DrawTable](#)

Function explain	
Name	SetCellHTML
Parameters	✔ TableID as String , Lin as Integer , Col as Integer , Properties as String , ContentsHTML as String
Results	✘ (none)
From version	3.10
 " Professional feature "	

Sets the contents of cell in HTML format. See  [SetCell](#)

See Also

 [SetCell](#)  [DrawTable](#)

Function explain

Name	SetCellTable
Parameters	✔ TableID as String , Lin as Integer , Col as Integer , Properties as String , TableID as String
Results	✘ (none)
From version	3.10

Links a table to a cell. Like nested tables do in HTML.

See Also

-  [AddRow](#)
-  [SetCell](#)
-  [SetCellHTML](#)
-  [DrawTable](#)

Function explain	
Name	SetProperty
Parameters	✔ PropertyID as Integer , Value as WideString
Result	✘ (none)
From version	1.00

Returns the specified property. The PropertyID is the parameter identifier that we want to change, it's an integer value and we normally uses constants or defines to easily use names in one include file. Remember to add the include instruction if you want to use name identifiers, if not, then use directly the value for the property you want to get.

Properties are defined in constant groups; text, graphics, documents, information, report, internal, etc. We call each property with the constant name, you should find an information on each constant defined in this help file under the constants help tree.

Info: When we started to design the library, we searched an easy way to change parameters for modifying the look of the PDF document, without using thousands of variables that could change between different versions without losing compatibilities with old version. Some libraries uses properties that on version x a property is n wide length and for example a string type, then it came version 2 and this version uses a double precision variable, making the old code being incompatible.

This is why we decided to implement two powerful functions that gets and sets values that modifies and interacts with the library. The value is always treated as a string, making the conversion inside the library and forgetting on incompatible version from version 1 to x.

Syntax

PDF.SetProperty PropertyNumber as Integer , Value as variant

Properties

[See this link to read all properties that can be altered](#)

Example in ASP:

```
const csPropTextFont = 100
' Write with Helvetica Font
PDF.SetProperty csPropTextFont, " F5
"
PDF.AddText " Hello world "
```

Example in C:

```
define csPropTextFont = 100;
' Write with Helvetica Font

main()
{
PDF.SetProperty (csPropTextFont, ' F5
');
PDF.AddText(' Hello world ');
}
```

Example in Delphi:

```
const csPropTextFont = 100;  
' Write with Helvetica Font  
PDF.SetProperty (csPropTextFont, ' F5  
PDF.AddText(' Hello world ');
```

See also

 [GetProperty](#)  [SetPropObj](#)

SetPropObj

Function explain

Name	SetPropObj
Parameters	✔ ObjectID as WideString , PropertyID as Integer , Value as WideString
Result	✘ (none)
From version	2 2.00



" Professional feature "

The SetProperty changes a global parameter, the SetPropObj changes an object parameter.

You can alter forms objects and VEP objects after loading it by [LoadVEPFile](#)

Syntax

PDF.SetPropObj ObjectID as string, PropertyNumber as Integer , Value as string

Properties

[See this link to read all properties that can be altered](#)

Example in ASP


```
<%  
' Create the component  
set PDF = server.createobject(" aspPDF.EasyPDF ")  
' Adds a Scroll list and introduces the values  
PDF.AddFormObj 320, 400, 420, 450, " form.listbox ", " test ", " Introduce the value combo value ", foScrollList  
PDF.SetPropObj " form.listbox ", csPropObjCbxValues, " [ (AMX)(American Express)] [ (CBL)(Carte Blanche)] [ (DCL)(Diners Club)] [ (DSC)(Discover)] [ (ENR)(EnRoute)] (JCB)[ (MSC)(MasterCard)] [ (VIS)(Visa)] "  
PDF.BinaryWrite  
set pdf = nothing  
%>
```

See also

[GetProperty](#) [SetPropety](#) [GetPropObj](#) [DrawVEP](#) [LoadVEPFile](#)

SetTrueTypeFont

Function explain

Name	SetTrueTypeFont
Parameters	✔ Font Name as String , Font as String , Parameters as Integer , Embedded as Integer
Result	✔ Result as Boolean
From version	1.00
 " Professional feature "	

Note : Obsolete function, please use the new AddFont function instead of the SetTrueTypeFont.

Enables you to add more fonts to the PDF document, as the PDF comes with 14 standard fonts some users requires to use additional fonts and most common the Windows True Type font. This options allows you to add a True type font on the document and use it.

The font is just linked to the document, so you must to take care that on the client side has the same font, if not the PDF will replace it with standard F1 - Helvetica font.

The **FontName** is the internal font name to use in PDF, you can use from **F15 to F29**, this allows you to define 15 different True type fonts.

The **Parameters** specifies the font style, to combine two or more properties just sum the values:

Fonts	
Number	Definition
0	None (Default)
1	Bold
2	Italic

The **Embedded** parameter means that the font will be embedded to the PDF document, making it cross platform and does not care if you have it install on all machines. This is quite important for use with special fonts or others international fonts that are sending it to different platforms that does have it installed on their machines, PDF will use the embedded font and make it readable.

When you embedded fonts the PDF Document will increase on size, between 120k to 500k, depends on the font to used.

Note: Currently it doesn't support Unicode Charest.

Embedded	
Number	Definition
0	None (Default)
1	Embedded

Syntax

PDF .SetTrueTypeFont FontName as string, TrueType as string, Parameters as Integer, Embedded as Integer

Example

```
<%
' Create the component
set PDF = server.createObject("aspPDF.EasyPDF")
' Set the True Type font Tahome to use it with the internal F15 name
```

```
PDF.SetTrueTypeFont "F15", "Tahoma", 0, 0
' Set the True Type font Tahome in BOLD to use it with the internal F16 name and embedded in the document
PDF.SetTrueTypeFont "F16", "Tahoma", 1, 1
' Set the font
PDF.SetFont "F15", 16, ""
' Add a text in Tahoma style
PDF.AddText "This is a text with Tahoma font<br>"
PDF.SetFont "F16", 16, ""
PDF.AddText "This is a text with Tahoma font with a Bold style and the font is embedded to the
document<br>"
' Destroy it
set pdf = nothing
%>
```

See also

 [AddFont](#)

Page

Function explain

Name	Page
Parameters	✔ PageForm as String , Orientation as Integer
Result	✘ (none)
From version	1.00

Sets the page size and orientation, valid options are:

For page " A3 ", " A4 ", " A5 ", " Letter ", " Legal ", " Envelope "
For the orientation you have: 0 - Vertical ; 1 - Horizontal

If you want to set up a non standard page size, use the 📄 [PageSize](#) function.

Syntax

PDF .Page type as String , orientation as Integer

See also

📄 [PageSize](#)

PageSize

Function explain

Name	PageSize
Parameters	✔ mmWidth as Double , mmHeight as Double , Orientation as Integer
Result	✘ (none)
From version	1.10

Enables you to set an user page size without using the standard ones. It must be set before using any other command and consecutive pages will use this size.

Syntax

PDF.PageSize mmWidth as Double , mmHeight as Double, Orientation as Integer

Note:

The Width and Height is in millimeters, to convert it from inches use this formula: inch * 25.4

For the orientation you have to options: 0 - Vertical ; 1 - Horizontal

Some page sizes in mm

Name	mm Width	mm Height
A3	297	420
A4	210	297
A5	148	210
Letter	215.9	279.4
Legal	215.9	305
USLegal	215.9	355.6
Envelope	210	110

Example

```
<%  
set PDF = server.createObject("aspPDF.EasyPDF")  
' Sets a A4 page manually ( It's a substitution of the Page function )  
PDF.PageSize 210, 297, 0  
...
```

See also

👉 [Page](#)

Save

Function explain

Name	Save
Parameters	✔ FileName as String
Result	✘ (none)
From version	1.00

Finally you write all the information to the disk and generates the pdf document. You can link the document dynamically and redirect it to display it on the user explorer.

NOTE: If you use ASP or ASP.NET development you should always check your write permissions to the IIS_users for write permissions on the folder that you will use to save your document. If the IIS anonymous user doesn't have sufficient privileges then it will fail to write to disk.

Syntax

PDF.Save FileName as String

Example in ASP:

```
<%  
set PDF = server.createobject  
("aspPDF.EasyPDF")  
' Your code  
PDF.Save Server.MapPath ("filename.pdf" )  
%>
```

See also

 [BinaryWrite](#)  [SaveStream](#)  [SaveString](#)  [SaveVariant](#)

SaveFDF

Function explain

Name	SaveFDF
Parameters	✔ FDFFileName as String , PDFFileName as String
Result	✘ (none)
From version	2 2.00



" Professional feature "

Saves the values list to the disk;

FDFFile name is the FDF file to be saved and the PDFFileName is the linked PDF document that will use it to fulfill the data. Please make sure the document is reachable, you can use http URLs.

Note: FDF files is only a list of form object values that must match with the one that is specified on the PDF template document that is defined when saving it.

Syntax

PDF.SaveFDF FDFFileName as String , PDFFileName as String

Example in ASP



See it on  [AddFDFValue](#) sample


See Also

 [BinaryWriteFDF](#)  [AddFDFValue](#)

SaveStream

Function explain

Name	SaveStream
Parameters	 (none)
Result	 AddrPtr as Pointer
From version	1.00

This is the same function as the  [Save](#) to disk, but it saves it on memory and returns a string containing the PDF information.

This function has been set for non-script programmers that wants to work directly with the contents of the PDF in memory.



Script programmers that doesn't allow you to use pointer manipulation then use the alternative SaveString function.


See also

 [Save](#)  [BinaryWrite](#)  [SaveString](#)  [SaveVariant](#)

SaveString

Function explain

Name	SaveString
Parameters	 (none)
Result	 Result as WideString
From version	1.70

This is the same function as the  [Save](#) to disk, but it saves it on memory and returns a string containing the PDF information.

This function has been set for script programmers that can no use memory pointers.

Syntax is ASP

```
Response.ContentType = "application/pdf"
```

```
Response.BinaryWrite PDF.SaveString
```




```
set PDF = nothing
```

See also

 [Save](#)  [SaveStream](#)  [BinaryWrite](#)  [SaveVariant](#)

SaveVariant

Function explain

Name	SaveVariant
Parameters	 (none)
Result	 Result as OleVariant (byte[])
From version	 2.10

This is the same function as the Save method, but instead to save it to disk it will return an array of bytes which you can redirect this contents directly to the client or save it to disk. This is the preferred way to work on interned developments because you don't have to establish securities permissions and you don't have to maintain PDF files on Disk.

If you code in .NET you should add a Reference to your project, from the COM window list browse for the aspEasyPDF library, then add it. It should create the Interop dll that you will find it on your project.

You can also create manually your project, create a bin directory and copy from the installation directory the **Interop.aspPDF.dll** to the bin folder of your project.

Syntax in .NET

Byte[] = PDF.SaveString

Example in ASP.NET using language C#:

```
<% @Page Language=" C# " %>
<% @Import Namespace=" System " %>
<% @Import Namespace=" System.Web " %>
<% @Import Namespace=" aspPDF " %>
<script language=" C# " runat=" server ">
private void Page_Load(Object sender, EventArgs e) {

    aspPDF.EASYPDF pdf = new aspPDF.EASYPDF();
    pdf.Create();
    Response.Clear();
    Response.ContentType=" application/pdf ";
    Response.AddHeader( " content-disposition ", " attachment; filename=MyPDF.PDF ");
    Response.BinaryWrite( (byte[]) pdf.SaveVariant());

}
</script>
```

See also

 [Save](#)  [SaveStream](#)  [SaveString](#)  [BinaryWrite](#)

SetFont

Function explain	
Name	SetFont
Parameters	✔ FontName as String , FontSize as Double , FontColor as String
Result	✘ (none)
From version	1.10

Sets the font to use for the text, it allows you to additionally specify the size and the color to use. You can use the F1 to F14 standard fonts and also use any font that has been added with the [AddFont](#) instruction. If you need that one of the parameters doesn't alter the actual properties, then you should leave it blank or zero.

Standard Internal PDF Fonts	
Internal Name	Font Description
F1	Helvetica
ABCDEF GHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz0123456789 áéíóú à è ì ò ù	
F2	Helvetica-Bold
ABCDEF GHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz0123456789 áéíóú à è ì ò ù	
F3	Helvetica-Oblique
<i>ABCDEF GHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz0123456789 áéíóú à è ì ò ù</i>	
F4	Helvetica-BoldOblique
<i>ABCDEF GHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz0123456789 áéíóú à è ì ò ù</i>	
F5	Courier
ABCDEF GHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz0123456789 áéíóú à è ì ò ù	
F6	Courier-Bold
ABCDEF GHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz0123456789 áéíóú à è ì ò ù	
F7	Courier-Oblique
<i>ABCDEF GHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz0123456789 áéíóú à è ì ò ù</i>	
F8	Courier-BoldOblique
<i>ABCDEF GHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz0123456789 áéíóú à è ì ò ù</i>	
F9	Times-Roman
ABCDEF GHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz0123456789 áéíóú à è ì ò ù	
F10	Times-Bold
ABCDEF GHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz0123456789 áéíóú à è ì ò ù	
F11	Times-Italic
<i>ABCDEF GHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz0123456789 áéíóú à è ì ò ù</i>	

F12	Times-BoldItalic
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789áéíóúàèìòù	
F13	Symbol
ΑΒΧΔΕΦΓΗΙΘΚΛΜΝΟΠΘΡΣΤΥΖΩΞΨ Ζαβγδεφγηιφκλμνοπθρστυωξψζ0123456789{ } [\]	
F14	ZapfDingbats
(See PDF_Fonts.pdf on the examples directory to see a F14 charset demo)	

Syntax

PDF.SetFont FontName as String , FontSize as Double , FontColor as String

Example

```
<%  
' Set the actual font  
PDF.SetFont " F1 ", 10, "#000000"  
"  
' Add Some text  
PDF.AddText " Hi this is a small  
text<br> "  
' Change the size and the color  
PDF.SetFont "", 14, "#0000FF "  
PDF.AddText " Hi this is a small  
text<br> "  
' Change only the color  
PDF.SetFont "", 0, "#FF00FF "  
PDF.AddText " Hi this is a small  
text<br> "  
' Change the font  
PDF.SetFont " F5 ", 0, ""  
PDF.AddText " Hi this is a small  
text<br> "  
%>
```

See also

-  [csPropTextFont](#)
-  [csPropTextSize](#)
-  [csPropTextColor](#)
-  [AddFont](#)

SetPos

Function explain

Name	SetPos
Parameters	✔ X as Double , Y as Double
Result	✘ (none)
From version	1.10

Sets the position of the text on the document, this has the same effect than changing individually the PosXCursor and PosYCursor properties.

Syntax

PDF.SetPos X as Double , Y as Double

Example

```
<%  
' Set Position  
PDF.SetPos 20, 50  
' Add Some text at 20,50 user space position  
PDF.AddText " Hi this is a small text "  
%>
```

See also

 [PosXCursor](#)  [PosYCursor](#)

SetEvenMargins

Function explain

Name	SetEvenMargins
Parameters	 Left as Double , Right as Double , Top as Double , Bottom as Double, Active as Boolean
Result	 (none)
From version	2.10

Add a new margin specification for Even pages, Odd pages uses default one. This is quite useful for book style reports.

Syntax

PDF.SetMargins Left as Double , Right as Double , Top as Double , Bottom as Double, Active as Boolean

Example

```
<%  
' Set Margins for ODD pages  
PDF.SetMargins 20, 50, 20, 20  
' Set Margins for Even pages  
PDF.SetEvenMargins 50, 20, 20, 20,  
True  
' Add Some text and respect the margins  
PDF.AddHTML " report.asp "  
%>
```

SetMargins

Function explain

Name	SetMargins
Parameters	✔ Left as Double , Right as Double , Top as Double , Bottom as Double
Result	✘ (none)
From version	1.00

Sets the margins of the text on the document, this has the same effect than changing individually the csPropMarLeft, csPropMarRight, csPropMarTop, csPropMarBottom property

Syntax

PDF.SetMargins Left as Double , Right as Double , Top as Double , Bottom as Double

Example

```
<%  
' Set Margins 20,20,20,20  
PDF.SetMargins 20, 20, 20, 20  
' Add Some text and respect the margins  
PDF.AddText " Hello world "  
%>
```

See also

 [csPropMarLeft](#)  [csPropMarRight](#)  [csPropMarTop](#)  [csPropMarBottom](#)

Advanced functions

These set of instructions are only for advanced users that wants to get a step behind of the EasyPDF library to alter, modify or add unsupported options that can not be done with the standard functions. It is required a good acknowledge of the Adobe® Acrobat® Reference document.



" Professional feature "

All the advanced functions can only be used on the Professional version

NOTICE: Adobe® Acrobat® is copyrighted of Adobe Systems Incorporated.

Advanced Graphic Function	
Name	Grph_AddToContents
Parameters	✔ Text as WideString
Result	✘ (none)
From version	🕒 2.00

Adds a PDF instructions for the graphic priority inside the contents page.

Advanced Graphic Function	
Name	Grph_CurveToC
Parameters	✔ X1 as Double , Y1 as Double , X2 as Double , Y2 as Double , X3 as Double , Y3 as Double
Result	✘ (none)
From version	🕒 2.00

Draws a Curve C

Advanced Graphic Function	
Name	Grph_CurveToV
Parameters	✔ X2 as Double , Y2 as Double , X3 as Double , Y3 as Double
Result	✘ (none)
From version	🕒 2.00

Draws a Curve V

Advanced Graphic Function	
Name	Grph_CurveToY
Parameters	✔ X1 as Double , Y1 as Double , X3 as Double , Y3 as Double
Result	✘ (none)
From version	

2 2.00

Draws a Curve Y

Advanced Graphic Function	
Name	Grph_InitDraw
Parameters	✗ (none)
Result	✗ (none)
From version	2 2.00

Initiate the graphic definition, line color and type and fill information.

Advanced Graphic Function	
Name	Grph_LineTo
Parameters	✓ X1 as Double , Y1 as Double
Result	✗ (none)
From version	2 2.00

Draws a line to that position, remember to use first the MoveTo.

Advanced Graphic Function	
Name	Grph_MoveTo
Parameters	✓ X1 as Double , Y1 as Double
Result	✗ (none)
From version	2 2.00

Initial graphic point.

Advanced Graphic Function	
Name	Grph_EndDraw
Parameters	✓ ClosePath as Boolean
Result	✗ (none)
From version	2 2.00

End and draw the graphic.

If you want to close the graphic path then set pass true as argument.

Basic information of aspEasyPDF enterprise and easyReportPDF

The enterprise version of **aspEasyPDF** adds **reportEasyPDF** library into **aspEasyPDF** and gives you the feature of creating reports with just a connection set. On the documentation of **aspEasyPDF** you will find all methods to create a simple report with all available functions.

You have two ways to create reports in **aspEasyPDF** enterprise version:

1) Batch functions which are all described in this document.

or


2) With the LoadVEPFile function which loads the document and renders the report. This is the fastest and preferred option and is the same function as the LoadFromFile from the **easyReportPDF** library.

The process is quite complex to understand at first but when you get the trick you may create a reports in few minutes. With the enterprise version we give a free licensed copy of **VisualEasyPDF** product, which makes the creation of reports as easy as to place objects in a report in a visual interface.

As the enterprise versions embeds **reportEasyPDF** you should also use the help file from this product, which is included on this installation and you should be able to get it through this link or from the Program file help. The information included on it will help you to understand how to create dynamic reports trough scripting, change or request user parameters, manage the objects in the document and so on which are not described on this document.

Brief aspEasyPDF enterprise functions	
Name	Description
AddDBBand	Adds a band to the report
AddDBBandGroup	Adds a group break band to the report
AddDBBarCode	Adds a field connection which is a barcode to the report
AddDBConnection	Adds a connection definition
AddDBText	Adds a text field to the report
AddDBGraphic	Adds a graphic field to the report
AddParameter	Adds an user parameter
AddScript	Adds script code to the report
AddTextField	Adds a text which can be manipulate trough the script
RenderReport	Renders the report, creates all pages but does not save it.
SetDBBandActive	Sets the active band to add all linked functions and properties
SetDBConnection	Changes the connection string
SetDBSQL	Set the SQL for the given connection
SetReportPage	States aspEasyPDF to use the actual page as a report page
SetParameter	Sets the parameters value for a given parameter.

Add DBBand

Function explain	
Name	Add DBBand
Parameters	✔ Name as String , Band as Integer , Height as Double
Result	✘ (none)
From version	3.20
 " Enterprise feature "	

Adds a Band to the report page. The band property specifies the type of the band and the height is the size that it will use to print the band. For the detail type it will use the height size * number of records that will have the report.

Note: You can not add two band types with different height, it will take the last height as effective.

Band Types	
Band	Type
5	Page header
10	Group header #1
11	Group header #2
12	Group header #3
13	Group header #4
14	Group header #5
20	Detail
22	Detail total
33	Group footer #1
32	Group footer #2
31	Group footer #3
30	Group footer #4
29	Group footer #5
34	Report footer
35	Page Footer

Syntax

PDF.AddDBBand Band as Integer , Height as Double

Example

```
<%
' Create the component
set PDF = server.createObject(" aspPDF.EasyPDF
")
' Sets
PDF. SetReportPage
PDF. Page "A4", 0
PDF. AddDBBand "Band1", 5, 100
PDF. AddDBBand "Band2", 20, 15
```

```
PDF. AddDBBand "Band3", 35, 100  
' destroy it  
set PDF = nothing  
%>
```

See Also

 [SetDBBandActive](#)  [SetDBBandGroup](#)

Add DBBand Group

Function explain

Name	Add DBBandGroup
Parameters	✔ Name as String , Band as Integer , FieldName as String , Height as Double
Result	✘ (none)
From version	3.20
 " Enterprise feature "	

Adds a group break Band to the report page. The band property specifies the type of the band and the height is the size that it will use to print the band. For the detail type it will use the height size * number of records that will have the report.

Remember that the SQL from detail connection should be ordered on the same way it will render the group breaks

Syntax

PDF.AddDBBandGroup Name as String , Band as Integer , FieldName as String , Height as Double

See Also

 [SetDBBandActive](#)  [SetDBBand](#)

Add DB BarCode

Function explain

Name	Add DBBarCode
Parameters	✔ Name as String , XPos as Double , YPos as Double , Height as Double , Type as Integer , Connection , Field as String
Result	✘ (none)
From version	3.20



" Enterprise feature "

Adds a bar code, all barcode function is described in [✔ AddBarCode](#) , for the connection and and field given. If you give a name you may manage the bar code graphic trough the script, see the samples section in the easyReportPDF help file.

Example

```
<%
' Create the component
set PDF = server.createobject(" aspPDF.EasyPDF ")
' Enterprise version
PDF. SetReportPage
PDF. Page "A4", 0
PDF. AddDBBand 5, 100
PDF. AddDBBand 20, 15
PDF. AddDBBand 35, 100
PDF. AddDBConnection " connection4", "Driver={MSSQL};Host=192.168.0.7;Database=Northwind;User ID=sa ",
20
PDF. SetDBSQL " connection4 ", " SELECT I.* FROM Invoices I "
PDF. SetDBBandActive 20
PDF. SetFont " F1 ", 10, "# 000000 "
PDF. AddDBText " field5 ", 8, 8, 137, " connection4 ", " CustomerName "
PDF. AddDBText " field6 ", 152, 8, 37, " connection4 ", " OrderID "
PDF. AddDBBarCode " graphic9 ", 304, 0, 10, 5, " connection4 ", " OrderID "
PDF. RenderReport
PDF. Save Server.MapPath(" orders.pdf ")
' destroy it
set PDF = nothing
%>
```

See also

[csPropGraphBCAngle](#)
[AddBarCode](#)


[csPropGraphBCRatio](#)

[csPropGraphBCText](#)

[GetBarCodeWidth](#)



Add DB Connection

Function explain	
Name	Add DBConnection
Parameters	✔ Name as String , Connection as String , LinkBand as Integer
Result	✔ ErrorCode as Integre
From version	3.20
 " Enterprise feature "	

Sets a database connection to the report. You may add as many connection as you want but only one can be linked to a band, normally the detail band type. Setting many connection may be used for data management in the script of for printing data that do not move in the report.

The one that is linked to the detail band is used for the loop process on the report.

We have try to use the MS notation for the connection string, each parameter should be separated with a semicolon ";" and blank spaces should be always set with the double quote ";

Connection parameters	
Driver	<p>Sets the driver to be used for the connection, at this moment those are available and are all native, if you experience some problems then you may use the ADO connection which will use the ADO driver from the database. Normally Native drivers are faster than ADO.</p> <p>ADO - Sets an ADO connection which enables you to connect to practically to all databases which provides a driver for ADO.</p> <p>FireBird - Free Relation database which offers SQL-92 standard instruction sets. Runs in many platforms. Web page .</p> <p>InterBase - Borland® InterBase®, web page .</p> <p>msSQL - The Microsoft SQL server. Web information page .</p> <p>MySQL - The mySQL server from 3.12 to 5.x Web page.</p> <p>Oracle - Oracle. Web page.</p> <p>Sqlite - C library that implements a self-contained, embeddable, zero-configuration SQL database engine. Linux, Windows . Web page .</p> <p>Sybase - Sybase databaser. Web page.</p>
Host	Specifies the host server. Can be introduced by IP address or by host name.
Port	Port address of the server. Set to 0 or do not introduce any value to use the default port connection.
Database	The database to be connected. Some databases can only use one database and this parameter may be ignore. For servers with multi-databases you should use this parameter to specify which database to use.
User ID	User ID for connection to the database. Use a valid user for the database authentication process-
Password	Sets the password for the connection user.
Properties	Sets additional properties for the connection. At this moment this only works with the ADO connection

Some string examples connection;

- **NorthWind msSQL** demo database connection trough a localserver (192.168.0.7):

```
PDF. AddDBConnection " connection4", "Driver={MSSQL};Host=192.168.0.7;Database=Northwind;User ID=sa ",
20
```

- **mySQL** database connection trough an internet server (192.168.0.7):

```
PDF. AddDBConnection " connection4", "Driver={MYSQL};Host=www.abbatia.net;Database=abbatia;Port=3306;User  
ID=root;Password= ", 20
```

- ADO connection to MS Access database file:

```
PDF. AddDBConnection " connection4", "Driver={ADO};Properties=""Provider=Microsoft.Jet.OLEDB.4.0;Data  
Source=C:\Documents and Settings\John\Mis documentos\BD1.MDB;Persist Security Info=False"" ", 20
```

Note: All AddDBConnection should be set before the SetDBSQL function, and always has to be set with a SQL for that connection, if not you will get an exception error.

Add DB Text

Function explain

Name	Add DB Text
Parameters	✔ Name as String, X as Double , Y as Double , Width as Double , Connection as String , Field as String
Result	✘ (none)
From version	3.20



" Enterprise feature "

Adds a database field from the given connection to the report. If it's linked to the detail band then it will print the give field for each record. The AddDBText gets it's primary function from the [AddTextWidth](#) function so it works in the same way.

Script support

Adds support for the *[name]* _OnPrintBefore and *[name]* _OnPrintAfter event. The result of the _OnPrintBefore is taken to the final output and the first parameter contains the field value. See the script section on the easyReportPDF help file.

Example

```
<%
' Create the component
set PDF = server.createObject(" aspPDF.EasyPDF ")
' Enterprise version
PDF. SetReportPage
PDF. Page "A4", 0
PDF. AddDBBand 20, 15
PDF. AddDBConnection " connection4", "Driver={MSSQL};Host=192.168.0.7;Database=Northwind;User ID=sa ",
20
PDF. SetDBSQL " connection4 ", " SELECT I.* FROM Invoices I "
PDF. SetDBBandActive 20
PDF. AddDBText " field6 ", 152, 8, 37, " connection4 ", " OrderID "
PDF. RenderReport
PDF. Save Server.MapPath(" orders.pdf ")
' destroy it
set PDF = nothing
%>
```

See also

[csPropTextAlign](#)

[AddTextWidth](#)

[AddHTML](#)

[cnvUnitMM](#)

[cnvUnitInch](#)

Add DBGraphic

Function explain

Name	Add DBGraphic
Parameters	✔ Name as String, X as Double, Y as Double, Connection as String, Field as String
Result	✘ (none)
From version	3.20



" Enterprise feature "

Same function as the [AddTextWidth](#) but adds the Name tag to the object. This way we can change the properties of the object while rendering the report through the script section. See the script section on the easyReportPDF help file to know more about changing dynamically the object properties.

If you want to set a static graphic then set the connection string to **URL** and specify the field as the place to get the image. header for fixed paths and http:// header for remote access.

See Also

[AddGraphicPos](#)

[csPropGraphZoom](#)

[csPropGraphWZoom](#)


[csPropGraphHZoom](#)

[csPropGraphJPGQuality](#)

[csPropGraphImageIndex](#)

Add Parameter

Function explain

Name	Add Parameter
Parameters	✔ Name as String, Description as String , Default as String , Type as Integer , askUser as Integer
Result	✘ (none)
From version	3.20
 " Enterprise feature "	

Adds a parameter to the report, this may be used to request the user for a value. The parameters are useful in combination with a SQL syntax which will execute it with the correct parameter.

To use the parameter inside the SQL sentence, just add the a double point to the Parameter name.

Note: Remember always to set the WEB_APP property to true if you are running it from a Web server application on in batch process. This will avoid to request the user with a user dialog (which will appear in the server, not in the client when running in a Web server)


Example

```
PDF. AddParameter(" Order_ID ", " Introduce Order ID: ", "0", 1, 1);  
PDF. AddDBConnection " connection4", "Driver={MSSQL};Host=192.168.0.7;Database=Northwind;User ID=sa ",  
20  
PDF. SetDBSQL " connection4 ", " SELECT * FROM Invoices WHERE OrderID = :Order_ID "
```

See Also

 [SetParameter](#)  [SetDBSQL](#)

Add Script

Function explain	
Name	Add Script
Parameters	✔ Script as String
Result	✘ (none)
From version	3.20
 " Enterprise feature "	

Sets the script of the report to manage dynamic contents while rendering it.
 To add multiple script contents to the report without overriding it you should add at the first line the following charset:
 <+>





aspEasyPDF uses Pascal scripting technology to bring fast running process, using events and object oriented programming in the report. Is very easy to use and if you wish to learn how to use it see the easyReportPDF help file.

The compilation process of script takes when you issue the RenderReport function. You should always check if the renderReport returns false, if it does then check the LastError to see if you have an error on your script. Use always the VisualEasyPDF to code script inside aspEasyPDF, you may use breakpoints to debug it and you will see immediately any compilation error with just pressing a button.

Supported events:

- * For all field, text, graphic and shape object it supports the _OnPrintBefore and _OnPrintAfter
- * At this moment it only supports the detail band; Details_OnPrintBefore if returns false then it will not print the detail content of the recordset that points.

Internal commands

Script syntax which communicates with aspEasyPDF	
Methods	Description
DBDataSet	Returns the TDataSet object from the connection given:
DBField	Returns the TField object from the connection and field given.
getPropObj	This function works like the aspEasyPDF  GetPropObj
getProperty	This function works like the aspEasyPDF  GetProperty
SetPropObj	This function works like the aspEasyPDF  SetPropObj
SetProperty	This function works like the aspEasyPDF  SetPropety
Variables	Description
WEB_APP	Returns or sets the Web_APP variable
PosXCursor	Returns or sets the PosXCursor var, which is the actual position of the X Cursor on the document.
PosYCursor	Returns or sets the PosYCursor var, which is the actual position of the Y Cursor on the document.
PageNumber	Returns the PageNumber variable
PageCount	Returns PageCount variable
Constants	Description


Version	Returns Version variable
NVersion	Returns NVersion variable

Example

On this sample set a simple script which checks the Type field from connection4, if it's 1 then sets the field8 object to the X position at 50px if not, then it sets the X positions to 96px. Check that the setPropObj uses 605 constant identifier which is the same as the aspEasyPDF; **csPropObjPosX**.

```
scr =      " function details_onprintbefore; "  
scr = sc r + "var "  
scr = sc r + "  x : integer; "  
scr = sc r + "begin "  
scr = sc r + " if DBField('Connection4.Type').asInteger = 1  
then "  
scr = sc r + "  begin ");  
scr = sc r + "  setPropObj('field8', 605, '50'); "  
scr = sc r + " end "  
scr = sc r + " else "  
scr = sc r + " begin "  
scr = sc r + "  setPropObj('field8', 605, '96'); "  
scr = sc r + " end ;"  
scr = sc r + "end; "
```

PDF. **SetScript** src

Add Text Field	
Function explain	
Name	Add TextField
Parameters	✔ Name as String, X as Double , Y as Double , Width as Double , Text as String
Result	✘ (none)
From version	3.20
 " Enterprise feature "	

Same function as the [AddTextWidth](#) but adds the Name tag to the object. This way we can change the properties of the object while rendering the report trough the script section. See the script section on the easyReportPDF help file to known more about changing dynamically the object properties.

Script support

Adds support for the *[name]* _OnPrintBefore and *[name]* _OnPrintAfter event. The result of the _OnPrintBefore is taken to the final output. See the script section on the easyReportPDF help file.

It also adds some reserved syntax;

Reserved Syntax	
Band	Type
@Date()	Prints the actual date
@DateTime()	Prints the actual date time
@MAX(<i>conneciton.field, reset</i>)	Prints the maximum amount of the given connection and field notation You may also set a second parameter to true or false, which will make a reset after printing
@MIN(<i>conneciton.field, reset</i>)	Prints the minimum amount of the given connection and field notation You may also set a second parameter to true or false, which will make a reset after printing
@Page()	Prints the actual page
@SUM(<i>conneciton.field, reset</i>)	Prints the sumatory amount of the given connection and field notation You may also set a second parameter to true or false, which will make a reset after printing
@Time()	Prints the actual time

Example in C#

```

<% @Page Language="C#" %>
<% @Import Namespace="System" %>
<% @Import Namespace="System.Text" %>
<% @Import Namespace="System.Web" %>
<% @Import Namespace="aspPDF" %>
<script language="C#" runat="server">
// *****
// Remember to add a COM+ Reference of aspEasyPDF, this will create
// the Interop dll; Interop.aspPDF.dll for using it in C#
// *****
// ##Author##
// ##Date##
// ##VEPVER##
// VEP_BEGIN_AUTO_CODE
    
```

```
PDF. SetScript("<+>function details_onprintbefore; ");
PDF. SetScript("<+>var ");
PDF. SetScript("<+> x : integer;");
PDF. SetScript("<+>begin ");
PDF. SetScript("<+> if DBField('Type').asInteger = 1 then");
PDF. SetScript("<+> begin ");
PDF. SetScript("<+> setPropObj('field8', 605, '50' ); // Text X Position ");
PDF. SetScript("<+> setPropObj('field8', 100, 'F2' ); // Font F2 is for a bold font ");
PDF. SetScript("<+> setPropObj('field8', 104, '1' ); // Underline");
PDF. SetScript("<+> setPropObj('field9', 611, '0' ); // Set invidable the quantity");
PDF. SetScript("<+> setPropObj('field32', 611, '0' ); // Set invidable the price field");
PDF. SetScript("<+> end");
PDF. SetScript("<+> else");
PDF. SetScript("<+> begin");
PDF. SetScript("<+> setPropObj('field8', 605, '96' ); // Text X Position");
PDF. SetScript("<+> setPropObj('field8', 100, 'F1' ); // Font");
PDF. SetScript("<+> setPropObj('field8', 104, '0' ); // Underline off");
PDF. SetScript("<+> setPropObj('field9', 611, '1' ); // Quantity visible");
PDF. SetScript("<+> setPropObj('field32', 611, '1' ); // Price visible");
PDF. SetScript("<+> end;");
PDF. SetScript("<+> Result := True;");
PDF. SetScript("<+>end;");
private void DoPageVEP_1(aspPDF.EASYPDF PDF)
{
PDF. SetReportPage();
PDF. Page("A4", 0);
PDF. AddDBBand(5, 135);
PDF. AddDBBand(20, 15);
PDF. AddDBBand(35, 43);
PDF. AddDBConnection(" connection4 ", "Driver={MYSQL};Host=192.168.0.2;Database=""MITGestion"";Port=3306;User
ID=root;Password=", 20);
PDF. SetDBSQL("connection4", "SELECT f.*, f.Fabricacion_ID, f1.*, f1.Fabricacion_LinealID_SUB, f1.Posicion, f1.Tipo,
(f1.Precio * f1.Cantidad) PrecioToal FROM fabricaciones f, fabricaciones_detalle f1 WHERE
f.Fabricacion_ID=f1.Fabricacion_ID AND f.Fabricacion_ID = :FabricacionID AND (f1.Tipo = 1 OR f1.Tipo = 2 OR f1.Tipo =
4) ORDER BY f1.Fabricacion_LinealID, f1.Posicion ");
PDF. SetFont("F1", 12.5, "# 000000 ");
PDF. AddDBText(" field6 ", 392, 74, 107, " connection4 ", " Fabricacion_ID ", 0);
PDF. SetFont("F1", 10, "#000000");
PDF. SetDBBandActive(20);
PDF. AddDBText("field7", 8, 8, 89, "connection4", " Producto_ID ", 0);
PDF. AddDBText("field8", 96, 8, 169, "connection4", " Texto ", 0);
PDF. SetProperty(102, 1);
PDF. AddDBText("field9", 288, 8, 42, "connection4", " Cantidad ", 0);
PDF. SetPropObj("field9", 905, "%.2n");
PDF. SetFont("F1", 20, "#000000");
PDF. SetDBBandActive(5);
PDF. AddTextPos(160, 32, "ESCANDALLOS COMPLETOS");
PDF. SetFont("F1", 12.5, "#000000");
PDF. SetProperty(102, 0);
PDF. AddDBText(" field31 ", 504, 8, 82, " connection4 ", " PrecioToal ", 0);
PDF. SetPropObj(" field31 ", 905, "%.2n €");
PDF. AddDBText(" field32 ", 408, 8, 62, " connection4 ", " Precio ", 0);
PDF. SetPropObj(" field32 ", 905, "%.2n €");
PDF. SetDBBandActive(35);
PDF. AddLine(0, 3, 600, 3);
PDF. SetFont(" F1 ", 12.5, "# 000000 ");
PDF. SetDBBandActive(5);
PDF. AddTextPos(32, 74, " Referencia :");
PDF. AddTextPos(24, 90, " Descripción :");
PDF. AddDBBand(6, 6);
```

```
PDF. AddDBBand(34, 23);
PDF. SetFont("F2", 10, "#000000");
PDF. SetDBBandActive(35);
PDF. SetProperty(102, 0);
PDF. AddTextField("text8", 40, 16, 49, " @Page() ");
PDF. SetFont("F2", 10, "#000000");
PDF. SetDBBandActive(34);
PDF. AddTextField("text33", 392, 12, 196, " @Sum(connection4.PrecioToal) ");
PDF. SetPropObj("text33", 905, "%m");
PDF. SetDBBandActive(5);
PDF. AddDBGraphic("graphic36", 496, 11, "URL", "file:///K:\Archivos de Dibujo\MITData\logo_petit.bmp");
rdrrprt = PDF. RenderReport();
}
// VEP_END_AUTO_CODE
private void Page_Load(Object sender, EventArgs e) {
aspPDF.EASYPDF PDF = new aspPDF.EASYPDF();
DoPageVEP_1( PDF );
// SaveVariant is a function that just works for aspEasyPDF 2.1
Response.Clear();
Response.ContentType="application/pdf";
Response.AddHeader( "content-disposition","attachment; filename=MyPDF.PDF");
Response.BinaryWrite( (byte[]) PDF.SaveVariant());
}
</script>
```

RenderReport

Function explain

Name	RenderReport
Parameters	✘ (none)
Result	✔ Result as Boolean
From version	3.20



" Enterprise feature "

When you have ended all report definition and you wish to run it and create all the report pages you should call the RenderReport. This function will check the script for error, then create all connections and open all queries assigned to the report and the draw the report.

You should always look for the return result of this function, if doesn't succeed then you will check the LastError message to check which was the error.

After the report is rendered the same function will close all connections and wait for a new RenderReport call. If you issue another report definition then all structure will be empty to accommodate the new report.

You may render reports as many times as you want, it will create numerous pages. Then after finishing you can move to any page and add any information as it was an aspEasyPDF page.

After the report has been render you should save the PDF or redirected trough the browser with the appropriate command.

Example

```
<%
' Create the component
set PDF = server.createObject(" aspPDF.EasyPDF ")
' Enterprise version
PDF. SetReportPage
PDF. Page "A4", 0
PDF. AddDBBand 5, 100
PDF. AddDBBand 20, 15
PDF. AddDBBand 35, 100
PDF. AddDBConnection " connection4", "Driver={MSSQL};Host=192.168.0.7;Database=Northwind;User ID=sa ",
20
PDF. SetDBSQL " connection4 ", " SELECT I.* FROM Invoices I "
PDF. SetDBBandActive 20
PDF. SetFont " F1 ", 10, "# 000000 "
PDF. AddDBText " field5 ", 8, 8, 137, " connection4 ", " CustomerName "
PDF. AddDBText " field6 ", 152, 8, 37, " connection4 ", " OrderID "
PDF. AddDBBarcode " graphic9 ", 304, 0, 10, 5, " connection4 ", " OrderID "
PDF. RenderReport
PDF. Save Server.MapPath(" orders.pdf ")
' destroy it
set PDF = nothing
%>
```


SetDBBandActive

Function explain

Name	SetDBBandActive
Parameters	✔ Band as Integer
Result	✘ (none)
From version	3.20



" Enterprise feature "

When positioning fields, shapes, graphics and so on; aspEasyPDF should know to which band belongs, by setting the Active Band it will know to where it belongs.

Example

```
PDF. AddDBBand(5, 135);
PDF. AddDBBand(20, 15);
PDF. AddDBBand(35, 43);
PDF. AddDBConnection("connection4", "Driver={MYSQL};Host=192.168.0.2;Database=MITGestion;Port=3306;User
ID=root;Password=", 20);
PDF. SetDBSQL("connection4", "SELECT f.*, f.Fabricacion_ID, f1.*, f1.Fabricacion_LinealID_SUB, f1.Posicion, f1.Tipo,
(f1.Precio * f1.Cantidad) PrecioToal FROM fabricaciones f, fabricaciones_detalle f1 WHERE
f.Fabricacion_ID=f1.Fabricacion_ID AND f.Fabricacion_ID = :FabricacionID AND (f1.Tipo = 1 OR f1.Tipo = 2 OR f1.Tipo =
4) ORDER BY f1.Fabricacion_LinealID, f1.Posicion ");
PDF. SetDBBandActive (5);
PDF. AddBox(8, 56, 584, 105);
PDF. AddBox(120, 8, 480, 48);
PDF. SetFont("F1", 12.5, "#000000");
PDF. AddDBText("field6", 392, 74, 107, "connection4", "Fabricacion_ID", 0);
PDF. SetFont("F1", 10, "#000000");
PDF. SetDBBandActive (20);
PDF. AddDBText("field7", 8, 8, 81, "connection4", "Producto_ID", 0);
PDF. AddDBText("field8", 96, 8, 169, "connection4", "Texto", 0);
PDF. SetProperty(102, 1);
PDF. AddDBText("field9", 288, 8, 42, "connection4", "Cantidad", 0);
PDF. SetPropObj("field9", 905, "%.2n");
PDF. SetFont("F1", 20, "#000000");
PDF. SetDBBandActive (5);
PDF. AddTextPos(160, 32, "ESCANDALLOS COMPLETOS");
PDF. SetFont("F1", 12.5, "#000000");
```

SetDBConnection

Function explain

Name	SetDBSQL
Parameters	✔ Name as String , Connection as String
Result	✔ Result as Boolean
From version	3.20



" Enterprise feature "

Sets the a new connection string for a given object connection.


If it successes then it returns true otherwise it will be false.

To known the connection string please check the [AddDBConnection](#) function.

See also

[AddDBBand](#) [SetDBSQL](#)

SetDBSQL

Function explain	
Name	SetDBSQL
Parameters	✔ Name as String, SQL as String
Result	✘ (none)
From version	3.20
 " Enterprise feature "	

Sets the SQL sentence to the connection name. The SQL should always be in the same notation as the driver expects to be, so please, read the manual of your database to get the correct SQL sentence.

VisualEasyPDF adds a query manager to help you create the SQL with the proper syntax for driver you have selected for.

To use a parameter inside the SQL just add double point : at the first parameter name, and when running the report the aspEasyPDF will look for the value of this parameter.




Note: Please check if the RenderReport returns true, if not then check the LastError property to see if you have an error on the SQL.

Example


```
PDF. AddParameter(" Order_ID ", " Introduce Order ID: ", "0", 1, 1);  
PDF. AddDBConnection " connection4", "Driver={MSSQL};Host=192.168.0.7;Database=Northwind;User ID=sa ",  
20  
PDF. SetDBSQL " connection4 ", " SELECT * FROM Invoices WHERE OrderID = :Order_ID "
```

SetReportPage

Function explain

Name	SetReportPage
Parameters	 (none)
Result	 (none)
From version	3.20
	
" Enterprise feature "	

Sets the actual page to be rendered as report, this will initiated all memory connections and set the detail recordset to nil until get the connection structure.

Call this function always after an  [AddPage](#) has been made or when the object is initiated. You can have as many reports as you want in your document.

Example

```
<%
' Create the component
set PDF = server.createobject(" aspPDF.EasyPDF ")
' Report page 1
PDF. SetReportPage
PDF. AddDBBand 20, 15
PDF. AddDBConnection " connection4", "Driver={MSSQL};Host=192.168.0.7;Database=Northwind;User ID=sa ",
20
PDF. SetDBSQL " connection4 ", " SELECT I.* FROM Invoices I "
PDF. SetDBBandActive 20
PDF. AddDBText " field6 ", 152, 8, 37, " connection4 ", " OrderID "
PDF. RenderReport

' Report page 2
PDF. AddPage
PDF. SetReportPage
PDF. AddDBBand 20, 15
PDF. AddDBConnection " connection5", "Driver={MSSQL};Host=192.168.0.7;Database=Northwind;User ID=sa ",
20
PDF. SetDBSQL " connection5 ", " SELECT I.* FROM Invoices I "
PDF. SetDBBandActive 20
PDF. AddDBText " field7 ", 152, 8, 37, " connection4 ", " OrderID "
PDF. RenderReport

' Document page
PDF. AddPage
PDF. AddText "Hello world, this is the last page of the 2 report document"


PDF. Save Server.MapPath(" orders.pdf ")
' destroy it
set PDF = nothing
%>
```

See also

 [AddPage](#)

SetParameter


Function explain

Name	SetParameter
Parameters	✔ Name as String , Value as String
Result	✔ Result as Boolean
From version	3.20
 " Enterprise feature "	

Sets the a new value for a given parameter object.

If it successes then it returns true otherwise it will be false.

See also

 [AddParameter](#)

Constants properties definition

ID	Name	Description
Text manipulation		
100	csPropTextFont	Changes the font name, you can also use the SetFont function to change it.
101	csPropTextSize	Changes the font size, you can also use the SetFont function to change it.
102	csPropTextAlign	Align property for text
103	csPropTextColor	Changes the font color, you can also use the SetFont function to change it.
104	csPropTextUnderLine	Sets to use underline
105	csPropTextRender	Different render options
106	csPropTextOverGraph	Draws the text over the graphic, specifies order priority
107	csPropText3DColor	Draw the text in 3D
108	csPropText3DPos	3D position of the text
109	csPropTextAngle	Angle to draw the text
110	csPropCharSpacing	Char spacing
111	csPropWordSpacing	Word spacing
112	csPropTextEntityConv	Use entities on standard text, setting off can improve speed
113	csPropAddTextWidth	Different combinations for the addTextWidth function
114	csPropAddText_UpdPos	Update cursors when using Add xxxx Pos functions, by default is disabled.
115	csPropTextVertSpace	Controls the space between each tag
Page manipulation		
201	csPageMarLeft	Left margin of the actual page
202	csPageMarTop	Top margin of the actual page
203	csPageMarRight	Right margin of the actual page
204	csPageMarBottom	Bottom margin of the actual page
205	csPropPosX	X Cursor position
206	csPropPosY	Y Cursor position
207	csPageNumber	Actual page number
208	csPageWidth	Page width size
209	csPageHeight	Page height size
210	csPageBackColor	Actual back color of the page
211	csPageAutoAdd	Auto add pages when using auto text positioning functions, by default is set.
212	csPageCount	Returns the number of pages created.
HTML manipulation		
250	csHTML_TRFullPage	Rows can no not split in two different pages
252	csHTML_FontName	Default font name for the HTML, by default is F1
253	csHTML_FontSize	Default font size for the HTML
254	csHTML_TableDraw	Priority draw for the table
Information		
300	csPropInfoTitle	Title information
301	csPropInfoSubject	Subject information
302	csPropInfoAuthor	Author information
303	csPropInfoCreator	Creator information
304	csPropInfoKeywords	Keywords information
Document manipulation		
320	csDocuBackColor	Document back color
Graphic manipulation		
400	csPropGraphWidthLine	Width line, by default is 1
401	csPropGraphXAlign	Horizontal align for graphics
402	csPropGraphYAlign	Vertical align for graphics
403	csPropGraphWZoom	Width zoom

404	csPropGraphHZoom	Height zoom
405	csPropGraphLineColor	Line color
406	csPropGraphFillColor	Fill color
407	csPropGraphFilled	Is filled?
408	csPropGraphBoxShadow	Box shadow for box, ellipse and box rounded
409	csPropGraphShadowColor	Shadow color
410	csPropGraphWidthShadow	Size of the shadow
412	csPropGraphZoom	Zoom type, percent or fixed
413	csPropGraphBorder	Add Border for graphic images, size is 0 by default which means no border
414	csPropGraphBCText	Draw the bar code text inside the bar code image
415	csPropGraphBCAngle	Angle of the bar code
416	csPropGraphDashLine	Line style
417	csPropGraphShadowPos	Shadow position
418	csPropGraphJPGQuality	JPG image degradation
419	csPropGraphImageIndex	Image Index for TIFF and GIF multi-graphics
420	csPropGraphWidthImport	When importing EMF and WMF graphic format you can specify the default Width
421	csPropGraphHeightImport	When importing EMF and WMF graphic format you can specify the default Height
422	csPropGraphFullPage	Graphic should be on full page or can it be in the middle of one page to the other
423	csPropGraphBCRatio	Barcode ratios
424	csPropGraphPDF417_Mode	PD417 mode to be used
425	csPropGraphPDF417_SecurityLevel	Set the security level of a PDF417 barcode
426	csPropGraphPDF417_Truncate	Trucante PDF417 barcodes
427	csPropGraphInterpolate	Interpolate flag for the image
428	csPropGraphDPI	DPI quality of the image to be inserted on the document
429	csPropGraphAngle	Sets the angle of an image
Internal manipulation		
500	csPropIntVer	Internal PDF version
501	csPropIntLZLevel	Graphic compression level
502	csPropIntCoord	Use PDF coordinates or logical, by default is logical which mean 0 is top
503	csPropIntFitWin	When opening the PDF make it fit the page on the window
504	csPropIntFullScreen	Open the PDF in full screen mode
505	csPropIntProxyServer	Add proxy information for http request (only for the library)
506	csPropIntDebugFile	Debug information to a file, not on the screen
507	csPropIntLZDocLevel	Document compression level
508	csPropIntRelativePath	Use a relative path (only for the library)
509	csPropIntUnicodeCheck	Automatically detect if the string has Unicode characters
510	csPropIntDebugTime	Adds time information to the debug text
511	csPropIntDebugTCP	Connect to the aspEasyREG tool for TCP debugging
512	csPropIntDebugLevel	Level of information when debugging it
513	csPropIntProxyUser	Proxy User for basic proxy authentication
514	csPropIntProxyPass	Proxy password for basic proxy authentication
515	csPropIntHTTPTimeout	Sets the timeout for retrieving http requests
Objects manipulation		
600	csPropObjDesign	Object draw type
601	csPropObjBorder	Draw a border for the object
602	csPropObjBorderColor	Border color for the object
605	csPropObjPosX	X Position of a VEP object
606	csPropObjPosY	Y Position of a VEP object

607	csPropObjPosX1	X1 Position of a VEP object
608	csPropObjPosY1	Y1 Position of a VEP object
610	csPropObjText	Text contents of a VEP object
620	csPropObjOptions	Options
621	csPropObjOpened	Bookmark opened
690	csPropObjTxtMaxLen	Max length for the text filed
730	csPropObjCbxValues	Values for the combo and scroll list fields
Charts graphics		
800	csPropChartColor	Sets the background of the graphic Chart
801	csPropChartBackColor	Background color of the panel that draws the series charts
802	csPropChartBackImage	Background image of the panel that draws the series charts
803	csPropChartBackImageStyle	Style of the image to be drawn on the chart
804	csPropChartBackImageInside	Is the image inside the panel or in the chart
805	csPropChartGradientType	Gradient color direction
806	csPropChartGradientStartColor	Starting gradient color
807	csPropChartGradientEndColor	Ending gradient color
808	csPropChartGradientVisible	Is the gradient color visible?
820	csPropChart3DView	Set the chart in 3D
821	csPropChart3DElevation	Elevation of the chart
822	csPropChart3DOrthogonal	Using orthogonal view image
823	csPropChart3DPercent	Percentage of depth
824	csPropChart3DPerspective	Perspective view
825	csPropChart3DRotation	Rotation of the chart
826	csPropChart3DTilt	Tilt pos
827	csPropChart3DZoom	Zoom view
830	csPropChartAxisColor	Axis color
831	csPropChartAxisGridColor	Axis grid color
832	csPropChartAxisFontName	Font name for the Axis
833	csPropChartAxisFontSize	Size font for the axis
834	csPropChartAxisFontColor	Color font for the axis
835	csPropChartAxisLeftAngle	Axis left angle text
836	csPropChartAxisBottomAngle	Bottom angle
837	csPropChartAxisLeftType	Left Axis type
838	csPropChartAxisBottomType	Bottom Axis type
839	csPropChartAxisLeftInverted	Is the left Axis info inverted?
840	csPropChartAxisBottomInverted	Is the bottom Axis info inverted?
841	csPropChartAxisLeftGrid	Using left grid?
842	csPropChartAxisBottomGrid	Using bottom grid?
843	csPropChartAxisLeftVisible	Use left axis
844	csPropChartAxisBottomVisible	use bottom axis
860	csPropChartLegendPosition	Legend position
861	csPropChartLegendStyle	Legend style
862	csPropChartLegendContents	Display contents
863	csPropChartLegendColor	Back legend panel
864	csPropChartLegendFontName	Which font to use to print legend information
865	csPropChartLegendFontSize	Which font size to use to print legend information
866	csPropChartLegendFontColor	Text color
867	csPropChartLegendMaxRows	Max rows to display in the legend
868	csPropChartLegendInverted	Is the legend info inverted?
869	csPropChartLegendVisible	Use legend?

880	csPropChartWallVisible	Use Walls if the chart is in 3D
881	csPropChartWallLeftColor	Background color for the left wall if viewed in 3D
882	csPropChartWallBackColor	Background color for the back wall if viewed in 3D
883	csPropChartWallBottomColor	Background color for the bottom wall if viewed in 3D
884	csPropChartWallLeftSize	Wall left size
885	csPropChartWallBackSize	Wall back size
886	csPropChartWallBottomSize	Wall bottom size

ASP INCLUDE FILE

```
<%  
' INCLUDE FILE aspEasyPDF ( www.mitdata.com )  
' Global Properties definitions, see help file to known how to use it  
' Text  
const csPropTextFont = 100  
const csPropTextSize = 101  
const csPropTextAlign = 102  
const csPropTextColor = 103  
const csPropTextUnderLine = 104  
const csPropTextRender = 105  
const csPropText3DColor = 107  
const csPropText3DPos = 108  
const csPropTextAngle = 109  
const csPropCharSpacing = 110  
const csPropWordSpacing = 111  
const csPropTextEntityConv= 112  
const csPropAddTextWidth = 113  
const csPropAddText_UpdPos= 114  
const csPropTextVertSpace = 115  
' Page  
const csPageMarLeft = 201  
const csPageMarTop = 202  
const csPageMarRight = 203  
const csPageMarBottom = 204  
const csPropPosX = 205  
const csPropPosY = 206  
const csPageNumber = 207  
const csPageWidth = 208  
const csPageHeight = 209  
const csPageBackColor = 210  
const csPageAutoAdd = 211  
const csPageCount = 212  
' HTML  
const csHTML_TRFullPage = 250  
const csHTML_xxx = 251 ' for future use  
const csHTML_FontName = 252  
const csHTML_FontSize = 253  
const csHTML_TableDraw = 254  
const csHTML_ImageRatio = 255  
' Document information  
const csPropInfoTitle = 300  
const csPropInfoSubject = 301  
const csPropInfoAuthor = 302  
const csPropInfoCreator = 303  
const csPropInfoKeywords= 304  
' Document  
const csDocuBackColor = 320  
' Graphics  
const csPropGraphWidthLine = 400  
const csPropGraphXAlign = 401  
const csPropGraphYAlign = 402  
const csPropGraphWZoom = 403  
const csPropGraphHZoom = 404  
const csPropGraphLineColor = 405  
const csPropGraphFillColor = 406  
const csPropGraphFilled = 407  
const csPropGraphBoxShadow = 408  
const csPropGraphShadowColor = 409
```

```
const csPropGraphWidthShadow = 410
const csPropGraphBOXOnBack = 411
const csPropGraphZoom = 412
const csPropGraphBorder = 413
const csPropGraphBCText = 414
const csPropGraphBCAngle = 415
const csPropGraphDashLine = 416
const csPropGraphShadowPos = 417
const csPropGraphJPGQuality = 418
const csPropGraphImageIndex = 419
const csPropGraphWidthImport = 420
const csPropGraphHeightImport = 421
const csPropGraphFullPage= 421
const csPropGraphBCRatio = 421
' Internal
const csPropIntVer = 500
const csPropIntLZLevel = 501
const csPropIntCoord = 502
const csPropIntFitWin = 503
const csPropIntFullScreen = 504
const csPropIntProxyServer = 505
const csPropIntDebugFile = 506
const csPropIntLZDocLevel = 507
const csPropIntRelativePath = 508
const csPropIntUnicodeCheck = 509
const csPropIntDebugTime = 510

' Objects
const csPropObjDesign = 600
const csPropObjBorder = 601
const csPropObjBorderColor = 602
const csPropObjPosX = 605
const csPropObjPosY = 606
const csPropObjPosX1 = 607
const csPropObjPosY1 = 608
const csPropObjText = 610
const csPropObjOptions = 620
const csPropObjOpened = 621
const csPropObjTxtMaxLen = 690
const csPropObjCbxValues = 730
' Aligns
const algLeft = 0
const algRight = 1
const algCenter = 2
const algJustified = 3
' Code bars
const csCode_2_5_interleaved = 0
const csCode_2_5_industrial = 1
const csCode_2_5_matrix = 2
const csCode39 = 3
const csCode39Extended = 4
const csCode128A = 5
const csCode128B = 6
const csCode128C = 7
const csCode93 = 8
const csCode93Extended = 9
const csCodeMSI = 10
const csCodePostNet = 11
```

```
const csCodeCodabar = 12
const csCodeEAN8 = 13
const csCodeEAN13 = 14
const csCodeUPC_A = 15
const csCodeUPC_E0 = 16
const csCodeUPC_E1 = 17
const csCodeUPC_Supp2 = 18
const csCodeUPC_Supp5 = 19
const csCodeEAN128A = 20
const csCodeEAN128B = 21
const csCodeEAN128C = 22
' Link types
const lnkToURL = 0
const lnkToPDF = 1
const lnkDefineBookmark = 2
const lnkToBookmark = 3
const lnkToPosition = 4
' Pattern types
const patHeader = 0
const patFooter = 1

' Note type
const noteIcon = 0
const noteCross = 1
const noteTextBoxed = 2
' Form constants
const foButton = 0
const foCheckBox = 1
const foRadioButton = 2
const foTextField = 3
const foScrollList = 4
const foComboBox = 5
' Object Events
const aaOnMouseIn = 0
const aaOnMouseOut = 1
const aaOnMouseDown = 2
const aaOnMouseUp = 3
const aaOnFocusEnter = 4
const aaOnFocusExit = 5
const aaOnVisible = 6
const aaOnUnVisible = 7
const aaOnKeyPress = 8
const aaOnDrawField = 9
const aaOnValueChange = 10
' Security options
const scAllowPrint = 4
const scModifyContents = 8
const scCopyContents = 16
const scAllowNotesForms = 32
const scAllowForms = 256
const scExtractText = 512
const scAssembleDoc = 1024
const scPrintQuality = 2048

%>
```


General Properties			
Name	Constants	Default	Description
Color	csPropChartColor	Page Color	Sets the background of the graphic Chart
BackColor	csPropChartBackColor	Color	Background color of the panel that draws the series charts
BackImage	csPropChartBackImage	None	Background image of the panel that draws the series charts
BackImageStyle	csPropChartBackImageStyle	Stretch	Style of the image to be drawn on the chart
BackImage Inside	csPropChartBackImageInside	True	Is the image inside the panel or in the chart
GradientType	csPropChartGradientType	Top to Bottom	Gradient color direction
GradientStartColor	csPropChartGradientStartColor	White	Starting gradient color
GradientEndColor	csPropChartGradientEndColor	Yellow	Ending gradient color
GradientVisible	csPropChartGradientVisible	False	Is the gradient color visible?
3D Properties			
Name	Constants	Default	Description
3DView	csPropChart3DView	False	Set the chart in 3D
3DElevation	csPropChart3DElevation	345	Elevation of the chart
3DOrthogonal	csPropChart3DOrthogonal	False	Using orthogonal view image
3DPer cent	csPropChart3DPercent	35	Percentage of depth
3DPerspective	csPropChart3DPerspective	15	Perspective view
3DRotation	csPropChart3DRotation	345	Rotation of the chart
3DTilt	csPropChart3DTilt	0	Tilt pos
3DZoom	csPropChart3DZoom	100	Zoom view
Axis Properties			
Name	Constants	Default	Description
AxisColor	csPropChartAxisColor	True	Axis color
AxisGridColor	csPropChartAxisGridColor	Black	Axis grid color
AxisFontName	csPropChartAxisFontName	Arial	Font name for the Axis
AxisFontSize	csPropChartAxisFontSize	8	Size font for the axis
AxisFontColor	csPropChartAxisFontColor	Black	Color font for the axis
AxisLeftAngle	csPropChartAxisLeftAngle	0	Axis left angle text
AxisBottomAngle	csPropChartAxisBottomAngle	0	Bottom angle
AxisLeft Type	csPropChartAxisLeftType	0	Left Axis type
AxisBottomType	csPropChartAxisBottomType	0	Bottom Axis type
AxisLeftInverted	csPropChartAxisLeftInverted	False	Is the left Axis info inverted?
AxisBottomInverted	csPropChartAxisBottomInverted	False	Is the bottom Axis info inverted?
AxisLeft Grid	csPropChartAxisLeftGrid	True	Using left grid?
AxisBottomGrid	csPropChartAxisBottomGrid	True	Using bottom grid?
AxisLeftVisible	csPropChartAxisLeftVisible	True	Use left axis
AxisBottomVisible	csPropChartAxisBottomVisible	True	use bottom axis
Legend Properties			
Name	Constants	Default	Description
LegendPosition	csPropChartLegendPosition	Right	Legend position
LegendStyle	csPropChartLegendStyle	Auto	Legend style
LegendContents	csPropChartLegendContents	Left Value	Display contents
LegendColor	csPropChartLegendColor	White	Back legend panel
LegendFontName	csPropChartLegendFontName	Arial	Which font to use to print legend information
LegendFontSize	csPropChartLegendFontSize	8	Which font size to use to print legend information
LegendFontColor	csPropChartLegendFontColor	Black	Text color
LegendMaxRows	csPropChartLegendMaxRows	10	Max rows to display in the legend

LegendInverted	csPropChartLegendInverted	False	Is the legend info inverted?
LegendVisible	csPropChartLegendVisible	True	Use legend?
Wall Properties			
Name	 Constants	Default	Description
WallVisible	csPropChartWallVisible	False	Use Walls if the chart is in 3D
WallLeftColor	csPropChartWallLeftColor	Color	Background color for the left wall if viewed in 3D
WallBackColor	csPropChartWallBackColor	Color	Background color for the back wall if viewed in 3D
WallBottomColor	csPropChartWallBottomColor	Color	Background color for the bottom wall if viewed in 3D
WallLeftSize	csPropChartWallLeftSize	0	Wall left size
WallBackSize	csPropChartWallBackSize	0	Wall back size
WallBottomSize	csPropChartWallBottomSize	0	Wall bottom size

Constant Definition	
ID	800
Name	csPropChartColor
Type	String
Default	Page Default
GetPropObj	Yes
GetProperty	No
SetProperty	No

Sets the chart color, by default uses the page default back color when added chart is added.

Constant Definition	
ID	801
Name	csPropChartBackColor
Type	String
Default	#FFFFFF
GetPropObj	Yes
GetProperty	No
SetProperty	No

Sets the color of the panel that will draw the chart.

Constant Definition	
ID	802
Name	csPropChartBackImage
Type	String
Default	Empty
GetPropObj	Yes
GetProperty	No
SetProperty	No

Sets an image in the panel

Constant Definition	
ID	803
Name	csPropChartBackImageStyle
Type	Integer
Default	0
GetPropObj	Yes
GetProperty	No
SetProperty	No

Sets the style of the image, previously loaded with csPropChartBackImage.

0 - Stretch; Image will be resized to fit the panel

1 - Tile; Image will be tiled

2 - Center; Image will be centered and not resized.

Constant Definition	
ID	804
Name	csPropChartBackImageInside
Type	Integer
Default	0
GetPropObj	Yes
GetProperty	No
SetProperty	No

Image will be drawn inside the panle and not in the full chart graphic.

Constant Definition	
ID	805
Name	csPropChartGradientVisible
Type	Integer
Default	0
GetPropObj	Yes
GetProperty	No
SetProperty	No

Sets the gradient colors for the chart panel

Constant Definition	
ID	806
Name	csPropChartGradientStartColor
Type	String
Default	#000000
GetPropObj	Yes
GetProperty	No
SetProperty	No

Defines the start color for the gradient

Constant Definition	
ID	807
Name	csPropChartGradientEndColor
Type	String
Default	#000000
GetPropObj	Yes
GetProperty	No
SetProperty	No

Defines the end color for the gradient panel

Constant Definition	
ID	808
Name	csPropChartGradientType
Type	Integer
Default	0
GetPropObj	Yes
GetProperty	No
SetProperty	No

Sets the gradient direction;

0 - Top to bottom

1 - Bottom to top

2 - Left to right

3 - Right to left

Constant Definition	
ID	820
Name	csPropChart3DView
Type	Integer
Default	0 (False)
GetPropObj	Yes
GetProperty	No
SetProperty	No

Sets the chart in 3D view

Constant Definition	
ID	821
Name	csPropChart3DElevation
Type	Integer
Default	0
GetPropObj	Yes
GetProperty	No
SetProperty	No

Elevation describes front plane rotation by rotation degrees (0 - 360°). Increasing the value positively will bring the top of the Chart towards the viewer and the bottom of the Chart away, moving around an horizontal axis at the central vertical point of the Chart.

Note: csPropChart3DOrthogonal should be set to False for Elevation to act on the Chart.

Constant Definition	
ID	822
Name	csPropChart3DOrthogonal
Type	Integer
Default	0
GetPropObj	Yes
GetProperty	No
SetProperty	No

Sets the 3D allowing Elevation and Rotation displacement of the Chart.

Constant Definition	
ID	823
Name	csPropChart3DPerspective
Type	Integer
Default	0
GetPropObj	Yes
GetProperty	No
SetProperty	No

Perspective sets the view of the Chart with perspective effect (dimensional appearance with respect to distance from the viewer).

Values are integer from 0 -> 100%

When set to 0 perspective is disabled.

Note: csPropChart3DOrthogonal should be set to False for Perspective to act on the Chart.

Constant Definition	
ID	824
Name	csPropChart3DRotation
Type	Integer
Default	0
GetPropObj	Yes
GetProperty	No
SetProperty	No

Rotation describes front plane rotation by rotation degrees (0 - 360°). Increasing the value positively will bring the right of the Chart towards the viewer and the left of the Chart away, moving around a vertical axis at the central horizontal point of the Chart.

Constant Definition	
ID	825
Name	csPropChart3DTilt
Type	Integer
Default	0
GetPropObj	Yes
GetProperty	No
SetProperty	No

Tilt will rotate the Chart Rectangle within the Chart Panel. Positive values (from 0 to 360°) rotate the Chart anti-clockwise, negative values, clockwise.

Constant Definition	
ID	826
Name	csPropChart3DZoom
Type	Integer
Default	100
GetPropObj	Yes
GetProperty	No
SetProperty	No

Zoom will zoom the whole Chart. Expressed as a percentage, Increasing the value positively will bring the Chart towards the viewer, increasing the overall Chart size as the Zoom value increases. The Chart may be enlarged to a size greater than the Chart panel thus bringing the axes outside of the viewable area.

Negative values of Zoom will diminish the overall Chartsize until, at values less than 0 the Chart will re-increase in size.

Constant Definition	
ID	827
Name	csPropChart3DPercent
Type	Integer
Default	15
GetPropObj	Yes
GetProperty	No
SetProperty	No

The Chart3DPercent property indicates the size ratio between Chart dimensions and Chart depth when Chart.View3D is True. You can specify a percent number from 1 to 100.

Constant Definition	
ID	830
Name	csPropChartAxisColor
Type	String
Default	#000000
GetPropObj	Yes
GetProperty	No
SetProperty	No

Sets the axis colors

Constant Definition	
ID	831
Name	csPropChartAxisGridColor
Type	String
Default	#000000
GetPropObj	Yes
GetProperty	No
SetProperty	No

Sets the axis grids color

Constant Definition	
ID	832
Name	csPropChartAxisFontName
Type	String
Default	Arial
GetPropObj	Yes
GetProperty	No
SetProperty	No

Sets the Windows font to use for generating the Axis text

Constant Definition	
ID	833
Name	csPropChartAxisFontSize
Type	Integer
Default	10
GetPropObj	Yes
GetProperty	No
SetProperty	No

Sets the size font for the axis.

Constant Definition	
ID	834
Name	csPropChartAxisFontColor
Type	String
Default	#00000
GetPropObj	Yes
GetProperty	No
SetProperty	No

Sets the font color for the axis information

Constant Definition	
ID	835
Name	csPropChartAxisLeftAngle
Type	Integer
Default	0
GetPropObj	Yes
GetProperty	No
SetProperty	No

The csPropChartAxisLeftAngle property defines the rotation degree applied to the Axis Label. Valid degree angle values are 0, 90, 180, 270 and 360.

Constant Definition	
ID	836
Name	csPropChartAxisBottomAngle
Type	Integer
Default	0
GetPropObj	Yes
GetProperty	No
SetProperty	No

The csPropChartAxisBottomAngle property defines the rotation degree applied to the Axis Label. Valid degree angle values are 0, 90, 180, 270 and 360.

Constant Definition	
ID	837
Name	csPropChartAxisLeftType
Type	Integer
Default	0
GetPropObj	Yes
GetProperty	No
SetProperty	No

Defines the label type, possible values are:

0 - Chooses automatically

1 - No label

2 - Axis labeling is based on axis Minimum and Maximum values.

Constant Definition	
ID	838
Name	csPropChartAxisBottomType
Type	Integer
Default	0
GetPropObj	Yes
GetProperty	No
SetProperty	No

Defines the label type, possible values are:

0 - Chooses automatically

1 - No label

2 - Axis labeling is based on axis Minimum and Maximum values.

Constant Definition	
ID	839
Name	csPropChartAxisLeftInverted
Type	Integer
Default	0
GetPropObj	Yes
GetProperty	No
SetProperty	No

The Inverted property indicates, when True, to draw the Legend items in opposite direction.

Constant Definition	
ID	840
Name	csPropChartAxisBottomInverted
Type	Integer
Default	0
GetPropObj	Yes
GetProperty	No
SetProperty	No

The Inverted property indicates, when True, to draw the Legend items in opposite direction.

Constant Definition	
ID	841
Name	csPropChartAxisLeftGrid
Type	Integer
Default	1
GetPropObj	Yes
GetProperty	No
SetProperty	No

Displays left grid

Constant Definition	
ID	842
Name	csPropChartAxisBottomGrid
Type	Integer
Default	1
GetPropObj	Yes
GetProperty	No
SetProperty	No

Displays bottom grid

Constant Definition	
ID	843
Name	csPropChartAxisLeftVisible
Type	Integer
Default	1
GetPropObj	Yes
GetProperty	No
SetProperty	No

Is the left axis visible

Constant Definition	
ID	844
Name	csPropChartAxisBottomVisible
Type	Integer
Default	1
GetPropObj	Yes
GetProperty	No
SetProperty	No

Bottom Axis visible

Constant Definition	
ID	860
Name	csPropChartLegendPosition
Type	Integer
Default	0
GetPropObj	Yes
GetProperty	No
SetProperty	No

Where the legend will be displayed;

0 - Left

1 - Right

2 - Top

3 - Bottom

Constant Definition	
ID	861
Name	csPropChartLegendStyle
Type	Integer
Default	0
GetPropObj	Yes
GetProperty	No
SetProperty	No

Defines the style value for the legend;

0 - Auto

1 - Series

2 - Values

3 - Last values of each series

Constant Definition	
ID	862
Name	csPropChartLegendContents
Type	Integer
Default	1
GetPropObj	Yes
GetProperty	No
SetProperty	No

How to display the contents on the Legend;

- 0 - Text information
- 1 - Left value
- 2 - Right Value
- 3 - Left percent
- 4 - Right percent
- 5 - It's X and Y values

Constant Definition	
ID	863
Name	csPropChartLegendColor
Type	String
Default	#FFFFFF
GetPropObj	Yes
GetProperty	No
SetProperty	No

Back color for the legend

Constant Definition	
ID	864
Name	csPropChartLegendFontName
Type	String
Default	Arial
GetPropObj	Yes
GetProperty	No
SetProperty	No

Text font for the Legend

Constant Definition	
ID	865
Name	csPropChartLegendFontSize
Type	Integer
Default	8
GetPropObj	Yes
GetProperty	No
SetProperty	No

Size font for the Legend

Constant Definition	
ID	866
Name	csPropChartLegendFontColor
Type	String
Default	#000000
GetPropObj	Yes
GetProperty	No
SetProperty	No

Font color for the legend text

Constant Definition	
ID	867
Name	csPropChartLegendMaxRows
Type	Integer
Default	10
GetPropObj	Yes
GetProperty	No
SetProperty	No

Max rows to display in the legend

Constant Definition	
ID	868
Name	csPropChartLegendInverted
Type	Integer
Default	0
GetPropObj	Yes
GetProperty	No
SetProperty	No

Invert the information on the legend box

Constant Definition	
ID	869
Name	csPropChartLegendVisible
Type	Integer
Default	1
GetPropObj	Yes
GetProperty	No
SetProperty	No

Displays the legend

Constant Definition	
ID	880
Name	csPropChartWallVisible
Type	Integer
Default	1
GetPropObj	Yes
GetProperty	No
SetProperty	No

Displays the Wall panel

Constant Definition	
ID	881
Name	csPropChartWallLeftColor
Type	String
Default	#
GetPropObj	Yes
GetProperty	No
SetProperty	No

Sets the color for the left panel Wall.

Constant Definition	
ID	882
Name	csPropChartWallBackColor
Type	String
Default	#
GetPropObj	Yes
GetProperty	No
SetProperty	No

Sets the color for the back panel Wall.

Constant Definition	
ID	883
Name	csPropChartWallBottomColor
Type	String
Default	#
GetPropObj	Yes
GetProperty	No
SetProperty	No

Sets the color for the bottom panel Wall.

Constant Definition	
ID	884
Name	csPropChartWallLeftSize
Type	Integer
Default	0
GetPropObj	Yes
GetProperty	No
SetProperty	No



Size of the left panel wall.

Constant Definition	
ID	885
Name	csPropChartWallBackSize
Type	Integer
Default	0
GetPropObj	Yes
GetProperty	No
SetProperty	No



Size of the back panel wall.

Constant Definition	
ID	886
Name	csPropChartWallBottomSize
Type	Integer
Default	0
GetPropObj	Yes
GetProperty	No
SetProperty	No



Size of the bottom panel wall.

Constant Definition	
ID	720
Name	csPropObjCbxValues
Type	String
Default	
SetPropObj	
From version	 2.00



Sets the different values for the combo box and for the list box

Constant Definition	
ID	601
Name	csPropObjBorder
Type	Integer
Default	1
SetPropObj	
From version	 2.00

You can set the border size if you use the Boxed design method for the object forms



Constant Definition	
ID	602
Name	csPropObjBorderColor
Type	String
Default	#0000FF
SetPropObj	
From version	 2.00

You can set the border color if you use the Boxed design method for the object forms


Constant Definition	
ID	600
Name	csPropObjDesign
Type	Integer
Default	2
SetPropObj	
From version	 2.00



Indicates the drawing method for the object forms. At the moment you can only use from 0 to 2 object design methods.

Object design	
Number	Definition
0	None
1	Boxed
2	Win95
3	WinXP
4	Mac OSX



Constant Definition	
ID	621
Name	csPropObjOpened
Type	Boolean
Default	False
SetPropObj	
From version	 2.00

Specifies if the Outline object is opened or not.



Applies to:
 [AddOutline](#)

Constant Definition	
ID	620
Name	csPropObjOptions
Type	Integer
Default	0
SetPropObj	
From version	 2.00



**** Under construction ****

Constant Definition	
ID	605
Name	csPropObjPosX
Type	Float
SetPropObj	
From version	 2.00



After loading a VEP form you can alter the X position of any object.

Constant Definition	
ID	606
Name	csPropObjPosY
Type	Float
SetPropObj	
From version	 2.00



After loading a VEP form you can alter the Y position of any object.

Constant Definition	
ID	607
Name	csPropObjPosX1
Type	Float
SetPropObj	
From version	 2.00



After loading a VEP form you can alter the X1 position of any object.

Constant Definition	
ID	608
Name	csPropObjPosY1
Type	Float
SetPropObj	
From version	 2.00


After loading a VEP form you can alter the Y1 position of any object.

Constant Definition	
ID	610
Name	csPropObjText
Type	String
SetPropObj	
From version	 2.00

After loading a VEP form you can alter the contents of any text object.

Constant Definition	
ID	690
Name	csPropObjTxtMaxLen
Type	Integer
Default	256
SetPropObj	
From version	 2.00

Sets the maximum size of a text box



Constant Definition	
ID	429
Name	csPropGraph Angle
Type	Integer
Default	0
GetProperty	✔
SetProperty	✔
From version	 3.00

Sets the angle of the image to be inserted.

Note: This property is not supported in combination of the Zoom factor or the DPI property.

Applies to:

 [AddGraphic](#)  [AddGraphicPos](#)

Constant Definition	
ID	415
Name	csPropGraphBCAngle
Type	Integer
Default	0
GetProperty	
SetProperty	
From version	1.6

Sets the angle of the Barcode. Valid options are 0, 90 and 270 degree angle. The 0 is for displaying it horizontally and for 90 and 270 is when you want to display the barcode in vertical position. (The 270 degrees displays it also vertically but flipped) .

Example:

```
<% ' Create the component
set PDF = server.createObject("aspPDF.EasyPDF")
PDF.SetProperty csPropGraphBCAngle, 90
PDF.AddBarcode 100,100,30,1,"1234"
PDF.BinaryWrite
set pdf = nothing
%>
```

See also

-  [csPropGraphBCRatio](#)
-  [csPropGraphBCText](#)
-  [GetBarCodeWidth](#)

Applies to:

-  [AddBarCode](#)

Constant Definition	
ID	423
Name	csPropGraphBCRatio
Type	Integer
Default	2
GetProperty	✓
SetProperty	✓
From version	2.10

Allows you to specify a different ratio for the barcode size.

PDF Example for barcode ratio of 1 and 2:



See also



 [csPropGraphBCAngle](#)

 [csPropGraphBCText](#)

 [GetBarCodeWidth](#)

Applies to:

 [AddBarCode](#)

Constant Definition	
ID	414
Name	csPropGraphBCText
Type	Integer
Default	0
GetProperty	
SetProperty	
From version	1.6

If you want to display text on the Bar code use this constant property to automatically do it for you. You have four options:

Options	
Value	Description
0	No text is displayed, this is by default
1	Display the code that have been passed as parameter
2	Display the type bar code used
3	Display combination of 1 plus 2

See also

-  [csPropGraphBCAngle](#)
-  [csPropGraphBCRatio](#)
-  [_GetBarcodeWidth](#)

Applies to:

-  [AddBarcode](#)

Constant Definition	
ID	413
Name	csPropGraphBorder
Type	Integer
Default	0
GetProperty	✔
SetProperty	✔
From version	1.5

Size of the border to draw on the graphic that has been inserted. This property is translated from the professional version it comes from the tag on the property Border. So we have decided that can be used on the standard version using a property.

Standard version example:

```
<% ' Create the component
set PDF = server.createObject("aspPDF.EasyPDF")
' Sets the border
PDF.SetProperty csPropGraphBorder, 2
PDF.AddGraphic "C:\inetpub\images\test.jpg"
PDF.BinaryWrite
set pdf = nothing
%>
```

Professional version example:

```
<% ' Create the component
set PDF = server.createObject("aspPDF.EasyPDF")
PDF.AddHTML "<IMG src="" C:\inetpub\images\test.jpg ""
Border=""2"">"
PDF.BinaryWrite
set pdf = nothing
%>
```

Constant Definition	
ID	408
Name	csPropGraphBoxShadow
Type	Number (Bool)
Default	False
GetProperty	✔
SetProperty	✔




When drawing a box you can set a box shadow or not, by default is not enabled.

See also

 [csPropGraphShadowColor](#)  [csPropGraphWidthShadow](#)  [csPropGraphShadowPos](#)

Applies to:

 [AddBox](#)  [AddBoxRounded](#)  [AddEllipse](#)

Constant Definition	
ID	416
Name	csPropGraphDashLine
Type	Integer
Default	0
GetProperty	
SetProperty	
From version	 2.00

You can use different dash line types to draw boxes, lines, circles, etc.

See also

 [csPropGraphLineColor](#)  [csPropGraphWL](#)

Applies to:

 [AddBox](#)  [AddBoxRounded](#)  [AddEllipse](#)  [AddLine](#)

Constant Definition	
ID	428
Name	csPropGraph DPI
Type	Integer
Default	72
GetProperty	✔
SetProperty	✔
From version	3 3.00



After adding the image you can set the DPI to use for that image in your document. A greater value generates a better quality image but it get smaller in a 100% view.

DPI View Size	
DPI Value	Image Size
72	100 %
144	50 %
288	25 %

Applies to:

 [AddGraphic](#)

 [AddGraphicPos](#)

Constant Definition	
ID	406
Name	csPropGraphFillColor
Type	String (Color)
Default	#FFFFFF
GetProperty	
SetProperty	

When drawing a box you can set the color for filling it. Use it in conjunction of the [csPropGraphFilled](#) constant.

Note: To insert a RGB color use the 3 hexadecimal html notation, for CMYK use 4 hexadecimal html notation, the library will automatically switch from RGB to CYMK or vice versa.



Color Definitions	
Color Name	sRGB Value
Black	#000000
Silver	#C0C0C0
Gray	#808080
White	#FFFFFF
Maroon	#800000
Red	#FF0000
Purple	#800080
Fuchsia	#FF00FF
Green	#008000
Lime	#00FF00
Olive	#080800
Yellow	#FFFF00
Navy	#000080
Blue	#0000FF
Teal	#008080
Aqua	#00FFFFFF


See also

 [csPropGraphFilled](#)

Applies to:

 [AddBox](#)  [AddBoxRounded](#)  [AddEllipse](#)

Constant Definition	
ID	407
Name	csPropGraphFilled
Type	Number (Bool)
Default	False
GetProperty	
SetProperty	

Sets the Box filled of the  [csPropGraphFillColor](#) color constant, 1 - Filled or 0 - Transparent. (By default is set to 0)

Example in ASP:




```
<% ' Create the component
set PDF = server.createObject(" aspPDF.EasyPDF ")
' Fill the box in Cyan for CMYK
PDF.SetProperty csPropGraphFilled, 1
PDF.SetProperty csPropGraphFillColor, "#FF000000"
"
PDF.AddBox 10,10,100,100
PDF.BinaryWrite
' destroy it
set pdf = nothing
%>
```

See also

 [csPropGraphFillColor](#)

Applies to:


 [AddBox](#)  [AddBoxRounded](#)  [AddEllipse](#)

Constant Definition	
ID	422
Name	csPropGraphFullPage
Type	Boolean
Default	True
GetProperty	
SetProperty	
From version	 2.10

When adding a graphic on the document check if the image fits on the actual page, if it doesn't then it will add a new page. If you set to false this property then you will have broken images between one page to the other.

Applies to:

 [AddGraphic](#)  [AddGraphicPos](#)

Constant Definition	
ID	421
Name	csPropGraphWidthImport
Type	Integer
Default	Image size
GetProperty	✔
SetProperty	✔
From version	 2.04



When loading a vector graphic from EMF or WMF you can specify the default size to convert in raster bitmap on the document. To gain image quality you can use big sizes and then scale the graphic to increase the DPI quality.


See also

 [csPropGraphWidthImport](#)

Applies to:

 [AddGraphic](#)  [AddGraphicPos](#)

Constant Definition	
ID	404
Name	csPropGraphHZoom
Type	String
Default	100
GetProperty	
SetProperty	


Zoom factor of the graphic for the height, default is 100 (for 100% original size when set to  [csPropGraphZoom](#) to true)

See also

 [csPropGraphWZoom](#)  [csPropGraphZoom](#)

Applies to:




 [AddGraphic](#)  [AddGraphicPos](#)

Constant Definition	
ID	419
Name	csPropGraphImageIndex
Type	Integer
Default	0
GetProperty	✔
SetProperty	✔
From version	 2.00

When loading TIFF multi-pages graphics or animated GIF, you can select which page or frame to load on the document by setting the frame to load.

Applies to:

 [AddGraphic](#)  [AddGraphicPos](#)




Constant Definition	
ID	427
Name	csPropGraph Interpolate
Type	Boolean
Default	False
GetProperty	
SetProperty	
From version	 3.00

Saves the image in Interpolate format.

Applies to:

 [AddGraphic](#)

 [AddGraphicPos](#)

Constant Definition	
ID	418
Name	csPropGraphJPGQuality
Type	Integer
Default	100
GetProperty	
SetProperty	
From version	 2.00

When loading any graphic that has 24 bits the library converts it automatically to an understandable format to the PDF reader, this is the JPG format. If you want to produce small documents and don't care about the graphic quality, you can use this setting to lower the JPG quality. By default is 100 which means the best performance for graphic quality and 0 means the worst.

Applies to:

-  [AddGraphic](#)
-  [AddGraphicPos](#)

Constant Definition	
ID	405
Name	csPropGraphLineColor
Type	String (Color)
Default	#000000
GetProperty	✔
SetProperty	✔

When drawing a box or a line you can set the color for the line.

Note: To insert a RGB color use the 3 hexadecimal html notation, for CMYK use 4 hexadecimal html notation, the library will automatically switch from RGB to CYMK or vice versa.

Color Definitions	
Color Name	sRGB Value
Black	#000000
Silver	#C0C0C0
Gray	#808080
White	#FFFFFF
Maroon	#800000
Red	#FF0000
Purple	#800080
Fuchsia	#FF00FF
Green	#008000
Lime	#00FF00
Olive	#080800
Yellow	#FFFF00
Navy	#000080
Blue	#0000FF
Teal	#008080
Aqua	#00FFFFFF

See also

 [csPropGraphDashLine](#)  [csPropGraphWL](#)

Applies to:

 [AddBox](#)  [AddBoxRounded](#)  [AddEllipse](#)  [AddLine](#)

Constant Definition	
ID	424
Name	csPropGraphPDF417_Mode
Type	Integer
Default	0
GetProperty	✔
SetProperty	✔
From version	3 3.00

When using the PDF417 you can specify how to read the input bar

Mode	
Mode	Mode type
0	csPDF417_Alphanumeric
1	csPDF417_Binary
2	csPDF417_Numeric

Applies to:




 [AddBarCode](#)

Constant Definition	
ID	425
Name	csPropGraphPDF417_ SecurityLevel
Type	Integer
Default	0
GetProperty	✔
SetProperty	✔
From version	3.00

When using the PDF417 type bar you can set an additional property to add error correction, you have 9 levels of corrections, on each level you correct a number of damaged characters by reducing the capacity of the barcode.

Security Levels		
Level	Correct error	ASCII Capacity
0	0	1850
1	1	1846
2	3	1838
3	7	1822
4	15	1790
5	31	1726
6	63	1598
7	127	1342
8	255	830



Applies to:
 [AddBarCode](#)

Constant Definition	
ID	426
Name	csPropGraphPDF417_Truncate
Type	Integer
Default	0
GetProperty	
SetProperty	
From version	 3.00

Allows to remove the right lines from the PDF417 bar codes, please check that the scan reader supports it. This makes the bar code smaller.

Applies to:

 [AddBarCode](#)

Constant Definition	
ID	409
Name	csPropGraphShadowColor
Type	String (Color)
Default	#C0C0C0
GetProperty	
SetProperty	

When drawing a box shadow you can set the color for the filled shadow.


Color Definitions	
Color Name	sRGB Value
Black	#000000
Silver	#C0C0C0
Gray	#808080
White	#FFFFFF
Maroon	#800000
Red	#FF0000
Purple	#800080
Fuchsia	#FF00FF
Green	#008000
Lime	#00FF00
Olive	#0808000
Yellow	#FFFF00
Navy	#000080
Blue	#0000FF
Teal	#008080
Aqua	#00FFFFFF

See also

 [csPropGraphBoxShadow](#)  [csPropGraphWidthShadow](#)  [csPropGraphShadowPos](#)

Applies to:

 [AddBox](#)  [AddBoxRounded](#)  [AddEllipse](#)

Constant Definition	
ID	417
Name	csPropGraphShadowPos
Type	String
Default	0
GetProperty	✓
SetProperty	✓
From version	 2.00

Alters the shadows positions



Options	
Value	Description
0	Right - Down
1	Left - Down
2	Left - Up
3	Right - Up

See also

 [csPropGraphBoxShadow](#)  [csPropGraphShadowColor](#)  [csPropGraphShadowWidth](#)

Applies to:

 [AddBox](#)  [AddBoxRounded](#)  [AddEllipse](#)

Constant Definition	
ID	412
Name	csPropGraphZoom
Type	Boolean
Default	False
GetProperty	
SetProperty	

Zoom factor, True means to use the percent values and false means to use fixed value.
By default is set to True.

Examples



```
<% ' Create the component
set PDF = server.createObject(" aspPDF.EasyPDF ")
' Zoom down 50% to get a 144 dpi quality
PDF.SetProperty csPropGraphZoom, 1
PDF.SetProperty csPropGraphWZoom, 50
PDF.SetProperty csPropGraphHZoom, 50
' Adds a graphic on the page
PDF.AddGraphicPos 100, 30, " C:\inetpub\images\test.jpg
"
PDF.BinaryWrite
' destroy it
set pdf = nothing
%>
```

See also

 [csPropGraphWZoom](#)  [csPropGraphHZoom](#)

Applies to:

 [AddGraphic](#)  [AddGraphicPos](#)

Constant Definition	
ID	401
Name	csPropGraphXAlign
Type	Number
Default	0
GetProperty	
SetProperty	



Aligns the graphic in X position. Possible values are:
0 - Use cursor position; 1 - Left; 2 - Right; 3 - Center

See also

 [csPropGraphYAlign](#)

Applies to:

 [AddGraphic](#)  [AddGraphicPos](#)

Constant Definition	
ID	402
Name	csPropGraphYAlign
Type	Number
Default	0
GetProperty	
SetProperty	



Aligns the graphic in Y position .
0 - Use cursor position; 1 - Left; 2 - Right; 3 - Center

See also

 [csPropGraphXAlign](#)

Applies to:

 [AddGraphic](#)  [AddGraphicPos](#)

Constant Definition	
ID	400
Name	csPropGraphWL
Default	1
Type	Number
GetProperty	
SetProperty	



Sets the width graphic when drawing lines, boxes or the underline.
By default is set to one.

See also

 [csPropGraphLineColor](#)  [csPropGraphDashLine](#)

Applies to:

 [AddBox](#)  [AddBoxRounded](#)  [AddEllipse](#)  [AddLine](#)

Constant Definition	
ID	410
Name	csPropGraphWidthShadow
Default	2
Type	Number
GetProperty	
SetProperty	



The width size of the rendered shadow box.


See also

 [csPropGraphBoxShadow](#)  [csPropGraphShadowColor](#)  [csPropGraphShadowPos](#)

Applies to:

 [AddBox](#)  [AddBoxRounded](#)  [AddEllipse](#)

Constant Definition	
ID	403
Name	csPropGraphWZoom
Type	String
Default	100
GetProperty	
SetProperty	


Zoom factor of the graphic for the width, default is 100 (for 100% original size when set to  [csPropGraphZoom](#) to true)

See also

 [csPropGraphZoom](#)  [csPropGraphHZoom](#)

Applies to:

 [AddGraphic](#)  [AddGraphicPos](#)

Constant Definition	
ID	420
Name	csPropGraphWidthImport
Type	Integer
Default	Image size
GetProperty	✔
SetProperty	✔
From version	 2.04



When loading a vector graphic from EMF or WMF you can specify the default size to convert in raster bitmap on the document. To gain image quality you can use big sizes and then scale the graphic to increase the DPI quality.

See also

 [csPropGraphHeightImport](#)

Applies to:

 [AddGraphic](#)  [AddGraphicPos](#)

Constant Definition	
ID	252
Name	csHTML_FontName
Type	String
Default	F1
GetProperty	
SetProperty	
From version	1.70



This constant tells which font to use by default when using the AddHTML function. By default uses the F1 font, Arial alike font.

See also

 [csHTML_FontSize](#)

Applies to:

 [AddHTMLPos](#)  [AddHTML](#)

Constant Definition	
ID	253
Name	csHTML_FontSize
Type	Integer
Default	10
GetProperty	
SetProperty	
From version	1.70



This constant tells which font size to use by default when using the AddHTML function. By default uses 10.

See also

 [csHTML_FontName](#)

Applies to:

 [AddHTMLPos](#)  [AddHTML](#)



Constant Definition	
ID	255
Name	csHTML_ImageRatio
Type	Double
Default	1.35
GetProperty	
SetProperty	
From version	2.0

When adding html images on a document the images will look bigger than the text size. To make an identical look of the image size from an html page the library will divide the height and width with a constant to make it look on the correct relation with the text.

If you set this setting to 1 then you will have the same relation that got previous versions.

Applies to:



 [AddHTMLPos](#)  [AddHTML](#)

Constant Definition	
ID	254
Name	csHTML_TableDraw
Type	Integer
Default	0
GetProperty	
SetProperty	
From version	1.71

Priority that tell the library when it should draw the table structure with it's contents.

Applies to:

 [AddHTMLPos](#)  [AddHTML](#)

Constant Definition	
ID	250
Name	csHTML_TRFullPage
Type	Boolean
Default	False
GetProperty	
SetProperty	
From version	1.52

When using the AddHTML feature there are sometimes that you need that every row of the table that must be placed entirely on the page, not broken <TR></TR> between each page. If set to true this property, the component will calculate each row height and will check if it fits on the actual page position, if it doesn't fit then it will add a new page and insert the row on the beginning of the next page.

Applies to:

 [AddHTMLPos](#)  [AddHTML](#)

Constant Definition	
ID	900
Name	csPropERPPParamName
Type	String
GetPropObj	Yes
GetPropObj	Yes

Returns the parameter name of the parameter object.

Constant Definition	
ID	901
Name	csPropERPPParamValue
Type	String
GetPropObj	Yes
GetPropObj	Yes

Returns the parameter name of the parameter object.

Constant Definition	
ID	902
Name	csPropERPPParamType
Type	String
GetPropObj	Yes
GetPropObj	Yes



Returns the parameter type.

Constant Definition	
ID	903
Name	csPropERPPParamAskUser
Type	String
GetPropObj	Yes
GetPropObj	Yes

Sets the parameter to be used on the dialog parameter user to ask a value for it. Set to 1 to enable it or 0 to disable it.

Constant Definition	
ID	970
Name	csPropERPPParamName
Type	String
GetProperty	Yes
GetProperty	Yes

Enables you to activate or deactivate the dialog parameter that will display to the user to select a value. Use this parameter in visual application, not in web development.

Constant Definition	
ID	302
Name	csPropInfoAuthor
Type	String
GetProperty	
SetProperty	



Header information - Set Author

Example:

```
const csPropInfoAuthor 302  
PDF. SetProperty csPropInfoAuthor, " John Lohmeyer "  
PDF. SetProperty 301, " My own subject of the document "
```

See also



 [csPropInfoTitle](#)  [csPropInfoSubject](#)  [csPropInfoCreator](#)  [csPropInfoKeywords](#)

Constant Definition	
ID	303
Name	csPropInfoCreator
Type	String
GetProperty	
SetProperty	

Header information - Sets creator string

See also



 [csPropInfoTitle](#)  [csPropInfoSubject](#)  [csPropInfoAuthor](#)  [csPropInfoKeywords](#)

Constant Definition	
ID	304
Name	csPropInfoKeywords
Type	String
GetProperty	
SetProperty	

Header information - Sets keywords to use on the PDF doc


See also

 [csPropInfoTitle](#)  [csPropInfoSubject](#)  [csPropInfoAuthor](#)  [csPropInfoCreator](#)

Constant Definition	
ID	301
Name	csPropInfoSubject
Type	String
GetProperty	
SetProperty	

Header information - Set Subject



See also

 [csPropInfoTitle](#)

 [csPropInfoAuthor](#)

 [csPropInfoCreator](#)



 [csPropInfoKeywords](#)

Constant Definition	
ID	300
Name	csPropInfoTitle
Type	String
GetProperty	
SetProperty	

Header information - Set title of the document



See also

 [csPropInfoSubject](#)  [csPropInfoAuthor](#)  [csPropInfoCreator](#)  [csPropInfoKeywords](#)

Constant Definition	
ID	502
Name	csPropIntCoord
Type	Number
Default	1
GetProperty	
SetProperty	

Using the addTextPos, AddLine, AddBox and AddGraphic you set x,y position coordinates of the page. PDF starts with 0,0 at the end of the page, if you set it to 1 then 0,0 coordinates gets the start of the page.

On version 0.95 and higher this will be set to 1, so when drawing a bitmap to 10,10 you will draw it on the start of the actual page but in PDF will be 10, 830 (depends on the page size property)

Constant Definition	
ID	506
Name	csPropIntDebugFile
Type	String
Default	
GetProperty	
SetProperty	
From version	1.4

Sometimes you need to debug your script but you don't want to see it on the Browser, or you can not use the Internet Explorer or Netscape to debug it because you redirect the page to the PDF reader, so you can not debug it. To allow you to debug pages without this problem you can redirect all the debug information to a file that can be read later.

This can be useful if you are in production and want to see what's happening with that nasty error, and debug it without knowing the final user that you are debugging your script.

Note:Remember that for the aspEasyPDF to create a log file must have permissions to a directory or a file, if not, you wont get the log file. Also this option only works when working with IIS, with other programming language will display the message box.

Example:

```
' Sets the debug to physical file  
PDF.SetProperty csPropIntDebugFile, "c:\inetpub\easy pdf.log"  
PDF.Debug = True
```

See also

 [csPropIntDebugTime](#)

See also:

 [Debug](#)  [Lic_Debug](#)

Constant Definition	
ID	512
Name	csPropIntDebugLevel
Type	Integer
Default	0
GetProperty	
SetProperty	
From version	2.2

Internal definition of level of debugging information:

Object design	
Number	Definition
0	Function calls
1	Low information details
2	Max information details

See also:

- [Debug](#)
- [Lic_Debug](#)
- [csPropIntDebug File](#)
- [csPropIntDebugTCP](#)
- [csPropIntDebugTime](#)

Constant Definition	
ID	511
Name	csPropIntDebugTCP
Type	String
Default	
GetProperty	
SetProperty	
From version	2.04




Sets the debugging output messages to the EasyREG tool by TCP/IP, this will enable you to debug remotely; configure the EasyReg as a listen service, check that the ports are the same as the aspEasyPDF.

Example:

```
' Debugging in localhost by port 8181
PDF.SetProperty csPropIntDebugTCP, " 127.0.0.1:8181
"
PDF.Debug = True
```

See also:




- [Debug](#)
- [Lic_Debug](#)
- [csPropIntDebug File](#)
- [csPropIntDebugTime](#)
- [csPropIntDebugLevel](#)

Constant Definition	
ID	510
Name	csPropIntDebugTime
Type	String
Default	
GetProperty	
SetProperty	
From version	 2.04

You can add the time information when debugging the library, this will help you to measure time elapsing when using different routines. For example you can see that importing graphics in JPG format takes less time that GIF graphics.

See also:

-  [Debug](#)
-  [Lic_Debug](#)
-  [csPropIntDebug File](#)
-  [csPropIntDebugTCP](#)
-  [csPropIntDebugLevel](#)

Constant Definition	
ID	516
Name	csPropIntDebugView
Type	Boolean
Default	False
GetProperty	
SetProperty	
From version	 3.10

Sets the debugging output to the Windows Kernel.



To get the all debug information from aspEasyPDF you may use a freeware tool for that purpose. We use the following freeware tool; DebugView from Sysinternals (<http://www.sysinternals.com/ntw2k/freeware/debugview.shtml>) you may download it and use it freely.

Example:

```
' Watch all kernel debug messages
PDF.SetProperty csPropIntDebugView, 1
PDF.Debug = True
```

See also:

-  [Debug](#)
-  [Lic_Debug](#)
-  [csPropIntDebug File](#)
-  [csPropIntDebugTime](#)
-  [csPropIntDebugLevel](#)

Constant Definition	
ID	504
Name	csPropIntFullScreen
Type	Number (Bool)
Default	0
GetProperty	
SetProperty	


Cool effect that acts when opening it on the PDF reader, it makes a full display on the screen of the document and disables standard menus and toolbars. With ESC key you get to the normal PDF window. Default is 0, no action to take.

Example:

'Cool effect

PDF.SetProperty csPropIntFullScreen, " 1 "

See also



 [csPropIntFitWin](#)

Constant Definition	
ID	503
Name	csPropIntFitWin
Type	Number (Bool)
Default	0
GetProperty	
SetProperty	

When opening the PDF reader will resize the document to a complete view of the first page on your screen. Default is 0

See also

 [csPropIntFullScreen](#)

Constant Definition	
ID	522
Name	csPropIntIgnoreCertificateDateInvalid
Type	Number (Bool)
Default	1
GetProperty	
SetProperty	

" Enterprise feature "



If you access SSL certificate web site you may set if you want to forbid the action if the web has a certification date expired. Default is 1, it ignores invalid date certificates

This property affects to the following functions (when http or https is used):

 [AddHTML](#)  [AddHTMLPos](#)  [AddGraphic](#)  [AddGraphicPos](#)  [AddPDF](#)  [LoadVEPFile](#)

See also

 [csPropIntIgnoreCertificateInvalid](#)

Constant Definition	
ID	521
Name	csPropIntIgnoreCertificateInvalid
Type	Number (Bool)
Default	1
GetProperty	
SetProperty	

" Enterprise feature "



If you access SSL certificate web site you may set if you want to forbid the action if the web hasn't a valid certification. Default is 1, it ignores invalid certification.

This property affects to the following functions (when http or https is used):



 [AddHTML](#)  [AddHTMLPos](#)  [AddGraphic](#)  [AddGraphicPos](#)  [AddPDF](#)  [LoadVEPFile](#)

See also

 [csPropIntIgnoreCertificateDateInvalid](#)

Constant Definition	
ID	515
Name	csPropIntHTTPTimeOut
Type	Integer
GetProperty	
SetProperty	
From version	3.0



Sets the default timeout for any HTTP request on the net. It's set for a maximum of 60 seconds, you can set it to a bigger value if you have slow Inet connections.

Constant Definition	
ID	517
Name	csPropIntHTTP Query
Type	Integer
Default	0
GetProperty	
SetProperty	
From version	3.1

Sets the default action for the HTTP request, 0 for a GET action and 1 for a POST action. Normally this should be set to 0 for normal action but if you need to use large amount of data parameter then you should use the POST action to let known the IIS to not truncate the parameters strings.

This property affects to the following functions (when http or https is used):

 [AddHTML](#)  [AddHTMLPos](#)  [AddGraphic](#)  [AddGraphicPos](#)  [AddPDF](#)  [LoadVEPFile](#)

Constant Definition	
ID	518
Name	csPropIntHTTPUser
Type	String
GetProperty	
SetProperty	

" Enterprise feature "

To allow access to any secured web site you need to set the appropriate user to access it.

The Enterprise version automatically accepts SSL certificates, can allow or disallow access to sites with invalid or expired certificates;

It can automatically retrieve proxy information from Internet Options of Control Panel, make HTTP requests with custom proxy settings or directly without any proxy server;



As it automatically supports several secure proxy authentication schemes: basic, digest, NTLM (NT Lan Manager), MSN (Microsoft Network), DPA (Distributed Password Authentication) and RPA (Remote Passphrase Authentication by CompuServe)

This property affects to the following functions (when http or https is used):

 [AddHTML](#)  [AddHTMLPos](#)  [AddGraphic](#)  [AddGraphicPos](#)  [AddPDF](#)  [LoadVEPFile](#)

See also

 [csPropIntHTTPass](#)

Constant Definition	
ID	519
Name	csPropIntHTTPUser
Type	String
GetProperty	
SetProperty	

" Enterprise feature "

To allow access to any secured web site you need to set the appropriate password to access it.

The Enterprise version automatically accepts SSL certificates, can allow or disallow access to sites with invalid or expired certificates;

It can automatically retrieve proxy information from Internet Options of Control Panel, make HTTP requests with custom proxy settings or directly without any proxy server;



As it automatically supports several secure proxy authentication schemes: basic, digest, NTLM (NT Lan Manager), MSN (Microsoft Network), DPA (Distributed Password Authentication) and RPA (Remote Passphrase Authentication by CompuServe)

This property affects to the following functions (when http or https is used):

 [AddHTML](#)  [AddHTMLPos](#)  [AddGraphic](#)  [AddGraphicPos](#)  [AddPDF](#)  [LoadVEPFile](#)

See also

 [csPropIntHTTPass](#)



Constant Definition	
ID	520
Name	csPropInt HTTPIgnoreError
Type	Number (Bool)
Default	1
GetProperty	
SetProperty	

If for some reason a web is not accessible or it display an error (returns a code different than 200 OK code) then it displays the contenst send by the server (normally the error). If you set it to false the it will not load the page and it will stop loadint the page.

From version 3.30 this set is to true so all error will be ignored and displayed it as is in the PDF.



This property affects to the following functions (when http or https is used):

 [AddHTML](#)  [AddHTMLPos](#)

Constant Definition	
ID	501
Name	csPropIntLZLevel
Type	Number
Default	1
GetProperty	
SetProperty	

The **csPropIntLZLevel** parameter determines the type of compression algorithm to be used for graphic compression, and must be one of the following:

Level compression	
Number	Definition
0	None
1	Fastest
2	Default compression
3	Maximum

Constant Definition	
ID	507
Name	csPropIntLZDocLevel
Type	Number
Default	0
GetProperty	
SetProperty	
From version	1.5



" Professional feature "

The **csPropIntLZDocLevel** parameter determines the type of compression algorithm to be used for document compression, and must be one of the following:

Level compression	
Number	Definition
0	None
1	Fastest
2	Default compression
3	Maximum

By default this is set to none, because is a main feature of the professional version, if you wish to use it you have to activate it.

Note: Remember that a compression of a big document it takes more time to generate it and less time to send it to the client. So we recommend to use it ever the compression level of 2 of any document that you generate.

Constant Definition	
ID	514
Name	csPropIntProxyPass
Type	String
GetProperty	
SetProperty	
From version	2.2



Set the password name for Proxy authentication services

This property affects to the following functions (when http or https is used):

 [AddHTML](#)  [AddHTMLPos](#)  [AddGraphic](#)  [AddGraphicPos](#)  [AddPDF](#)  [LoadVEPFile](#)

See also:

 [csPropIntProxyServer](#)  [csPropIntProxyUser](#)

Constant Definition	
ID	513
Name	csPropIntProxyUser
Type	String
GetProperty	
SetProperty	
From version	2.2

Set the user name for Proxy authentication services

This property affects to the following functions (when http or https is used):

 [AddHTML](#)  [AddHTMLPos](#)  [AddGraphic](#)  [AddGraphicPos](#)  [AddPDF](#)  [LoadVEPFile](#)

See also :

 [csPropIntProxyServer](#)  [csPropIntProxyPass](#)

Constant Definition	
ID	505
Name	csPropIntProxyServer
Type	String
Default	
GetProperty	✔
SetProperty	✔
From version	1.4

When using the professional version you may have the server inside a firewall or a proxy server, you can configure the proxy server for all the http requests. Use the notation of server, two points, and port.

Example:

```
' Proxy server
PDF.SetProperty csPropIntProxyServer, "192.168.1.1:80"
' Get it from internet!
PDF.AddGraphic " http://www.google.com/images/logo.gif "
```

Applies to:

-  [AddGraphic](#)
-  [AddGraphicPos](#)
-  [AddHTML](#)
-  [AddHTMLPos](#)
-  [csPropIntProxyUser](#)
-  [csPropIntProxyPass](#)

Constant Definition	
ID	508
Name	csPropIntRelativePath
Type	String
Default	IIS Value
GetProperty	✔
SetProperty	✔
From version	1.53

This property gets the actual relative path that is using it when you specify external files without full path notation. When running from the IIS server it will fill this property with the path of the ASP file that you are running it. If you use another language then you must fill the relative path manually to be used.




Relative paths are used for images location, see example:

Example:

```
PDF.Debug = True
' Print out the actual relative path
Response.write PDF.GetProperty(csPropIntRelativePath
)
PDF.AddHTML "<img source='images.gif'>"
PDF.AddGraphic "..\images\header.gif"
```



Applies to:

-  [AddGraphic](#)
-  [AddGraphicPos](#)
-  [AddHTML](#)
-  [AddHTMLPos](#)



Constant Definition	
ID	509
Name	csPropIntUnicodeCheck
Type	Boolean
Default	True
GetProperty	
SetProperty	
From version	 2.00

Checks for every text string that is passing by parameter to the library if it contains Unicode characters.

If you known that your code will never use Unicode, you can set this to false to avoid unnecessary checking of all strings that are being passed. This will get a better performance when using standard strings.

Constant Definition	
ID	500
Name	csPropIntVer
Type	String
GetProperty	
SetProperty	

Sets the output version of the PDF, you can override the default version for compatibility.

Constant Definition	
ID	320
Name	csDocuBackColor
Default	#FFFFFF
Type	String (Color)
GetProperty	
SetProperty	

Sets the background color for the whole document.

PDF does not have a property to tell which color to use, instant the tricky option is to set a filled box back on the document.



Color Definitions	
Color Name	sRGB Value
Black	#000000
Silver	#C0C0C0
Gray	#808080
White	#FFFFFF
Maroon	#800000
Red	#FF0000
Purple	#800080
Fuchsia	#FF00FF
Green	#008000
Lime	#00FF00
Olive	#080800
Yellow	#FFFF00
Navy	#000080
Blue	#0000FF
Teal	#008080
Aqua	#00FFFFFF

See also

 [csPageBackColor](#)

Applies to:



 [AddPage](#)

Constant Definition	
ID	211
Name	csPageAutoAdd
Type	Boolean
Default	True
GetProperty	
SetProperty	
From version	1.5

Normally when you are using the component this automatically adds new pages when need, see [AddText](#) or [AddHTML](#), sometimes you don't want that the pages are inserted automatically, with this property you can control it to enable or disable the auto adding pages.

Applies to:

 [AddText](#)  [AddHTML](#)



Constant Definition	
ID	210
Name	csPageBackColor
Type	String (Color)
Default	#FFFFFF
GetProperty	
SetProperty	

This enables you to specify the color of the actual page which you are editing, not for the whole document that is been set with the csDocuBackColor property.


Color Definitions	
Color Name	sRGB Value
Black	#000000
Silver	#C0C0C0
Gray	#808080
White	#FFFFFF
Maroon	#800000
Red	#FF0000
Purple	#800080
Fuchsia	#FF00FF
Green	#008000
Lime	#00FF00
Olive	#0808000
Yellow	#FFFF00
Navy	#000080
Blue	#0000FF
Teal	#008080
Aqua	#00FFFFFF

See also

 [csDocuBackColor](#)



Constant Definition	
ID	212
Name	csPageCount
Type	String
GetProperty	
SetProperty	
From version	1.5

Returns the number of pages that has been created on the PDF document

Note: From version 2.0 you can use the  [PageCount](#) property.

See also



 [PageCount](#)  [PageNumber](#)  [_csPageNumber](#)

Constant Definition	
ID	209
Name	csPageHeight
Type	Decimal
GetProperty	
SetProperty	


Gets the actual page height, this is read only property.
You may be interested on getting the [width page](#).

See also

 [csPageWidth](#)



Constant Definition	
ID	207
Name	csPageNumber
Type	Number
GetProperty	
SetProperty	

Gets the actual page number, this is read only property and does not have any option to control it.

Note: From version 2.0 you can use the  [PageNumber](#) property directly.

See also

 [PageNumber](#)  [PageCount](#)  [_csPageCount](#)

Constant Definition	
ID	208
Name	csPageWidth
Type	Decimal
GetProperty	
SetProperty	

Gets the actual page width, this is read only property.
You may be interested on getting the [height page](#).



When setting the page size with [PageSize](#) , you may need to retrieve the dimension later for use it to dynamically set cursors or whatever.

Example:

```
' Do my own center page
PWidth = P DF.GetProperty( csPageWidth
)
PText = "Hello World"
PLength = P DF.GetTextWidth( PText )
PX = (PWidth - PLength) / 2
PDF.AddTextPos PX, 100, "Hello world"
```

See also

 [csPageHeight](#)

Constant Definition	
ID	114
Name	csPropAddText_UpdPos
Type	Integer (Bool)
Default	False
GetProperty	
SetProperty	
From version	1.70

This constants defines the way that the cursors are updated. The csPropPosX and csPropPosY constants are the actual cursor positions when using AddText function. By default the AddTextPos which prints at text at a specified position is not being altered after using it. If you set the **csPropAddText_UpdPos** to 1 you specify that the cursors must be also being updated after and AddTextPos or AddTextWidth is being called.

Example:

```
PDF.SetProperty csPropAddText_UpdPos, 0
x1 = PDF.GetProperty csPropPosX
PDF.AddTextPos 10,10,"This is a test update cursors"
x2 = PDF.GetProperty csPropPosX
' x1 = x2, the cursors didn't update
PDF.SetProperty csPropAddText_UpdPos, 1
x1 = PDF.GetProperty csPropPosX
PDF.AddTextPos 10,10,"This is a test update cursors"
x2 = PDF.GetProperty csPropPosX
' x1 <> x2, csPropPosX points at the end of the text that has been inserted
```

Applies to:

-  [AddTextPos](#)
-  [AddTextWidth](#)

Constant Definition	
ID	113
Name	csPropAddTextWidth
Type	Decimal
Default	0
GetProperty	✓
SetProperty	✓
From version	1.6


Alters the function of AddTextWidth, you have three options:



- 0 - Text is cute if it doesn't fit on the width parameter.
- 1 - Text is cute if it doesn't fit on the width parameter and adds three points
- 2 - Text will wrap down

Example show of combinations of csPropAddTextWidth for string: " **Test string shows different options of addTextWidth**"

Test string sho
Test string s...
Test string shows different options of addTextWidth

Applies to:

 [AddTextWidth](#)

Constant Definition	
ID	110
Name	csPropCharSpacing
Type	Float
Default	0
GetProperty	
SetProperty	

Specify the space between each character. By default is 0 that means that it will not use the character spacing.

With this property you can set looks like this:

Hello World

 H e l l o W o r l d



 H e l l o W o r l d

See also:

 [csPropWordSpacing](#)  [csPropTextVertSpace](#)

Applies to:

 [AddText](#)  [AddTextPos](#)  [AddTextWidth](#)

Constant Definition	
ID	201
Name	csPropMarLeft
Type	Float
GetProperty	
SetProperty	

Sets the margin of the text from the left side to be use on the actual page and consecutive pages when creating news.

As this can act on the actual page you can use it to dynamically change it to set columns or paragraphs indents.

Example:



```
' Set margins to the left side to 10mm  
PDF.SetProperty csPropMarLeft, cnvUnitmm(10)  
PDF.AddText "Hello world"  
' Change it to 20mm for the next line  
PDF.SetProperty csPropMarLeft, cnvUnitmm(20)  
PDF.AddText "<br>Hello world"
```

See also:

 [SetMargins](#)  [csPropMarRight](#)  [csPropMarTop](#)  [csPropMarBottom](#)

Applies to:

 [AddText](#)  [AddHTML](#)  [AddLink](#)  [AddGraphic](#)

Constant Definition	
ID	203
Name	csPropMarRight
Type	Float
GetProperty	
SetProperty	

Sets the margin of the text from the right side.



As this can act on the actual page you can use it to dynamically change it to set columns or paragraphs indents.

See also:

 [SetMargins](#)  [_csPropMarLeft](#)  [_csPropMarTop](#)  [_csPropMarBottom](#)

Applies to:

 [AddText](#)  [AddHTML](#)  [AddLink](#)  [AddGraphic](#)

Constant Definition	
ID	202
Name	csPropMarTop
Type	Float
GetProperty	
SetProperty	



Sets the margin from the top of the page

See also:

 [SetMargins](#)  [_csPropMarLeft](#)  [_csPropMarRight](#)  [_csPropMarBottom](#)

Applies to:

 [AddText](#)  [AddHTML](#)  [AddLink](#)  [AddGraphic](#)

Constant Definition	
ID	204
Name	csPropMarBottom
Type	Float
GetProperty	
SetProperty	



Sets the margin from the bottom of the page



See also:




 [SetMargins](#)  [_csPropMarLeft](#)  [_csPropMarRight](#)  [_csPropMarTop](#)


Applies to:





 [AddText](#)  [AddHTML](#)  [AddLink](#)  [AddGraphic](#)

Constant Definition	
ID	205
Name	csPropPosX
Type	Float
GetProperty	
SetProperty	

Note: From version 2.0 you can use the direct properties:  [PosXCursor](#) and  [PosYCursor](#)

This property controls the cursors inside the page, this is quite useful to use it in combination with the  [AddText](#) method. You have two options to position text on the document, the  [AddTextPos](#) , that gives you the option to print everywhere with cursor position and the  [AddText](#) , that adds consecutive blocks of text on the page.

If you work with the  [AddText](#) you may need some times to control the cursors, knowing where we are putting the text and changing it to another position to take it faster to that place.

This property controls and changes each time we insert text on the page the X Cursor, Horizontal position. This affects to the  [AddText](#) ,  [AddGraphic](#) ,  [AddLink](#) and  [AddHTML](#) .

Example:



```
' Where we are?  
XPos = PDF.GetProperty(csPropPosX  
)  
' Move it 10 PDF unit  
XPos = XPos + PDF.cnvUnitmm(10)  
PDF.SetProperty csPropPosX, XPos  
PDF.AddText "Hello world"
```




See also:


 [PosXCursor](#)  [PosYCursor](#)  [csPropPosY](#)





Applies to:

 [AddText](#)  [AddHTML](#)  [AddLink](#)  [AddGraphic](#)

Constant Definition	
ID	206
Name	csPropPosY
Type	Decimal
GetProperty	
SetProperty	

This property controls the cursors inside the page, this is quite useful to use it in combination with the  [AddText](#) method. You have two options to position text on the document, the  [AddTextPos](#) , that gives you the option to print everywhere with cursor position and the  [AddText](#) , that adds consecutive blocks of text on the page.

If you work with the  [AddText](#) you may need some times to control the cursors, knowing where we are putting the text and changing it to another position to take it faster to that place.

This property controls and changes each time we insert text on the page the Y Cursor, vertical position. This affects to the  [AddText](#) ,  [AddGraphic](#) ,  [AddLink](#) and  [AddHTML](#) .

Example:

```
' Jump to 100 mm from the top  
PDF.SetProperty csPropPosY, cnvUnitmm(100)  
PDF.AddText "Hello world"
```

See also:

 [PosXCursor](#)  [PosYCursor](#)  [csPropPosX](#)

Applies to:

 [AddText](#)  [AddHTML](#)  [AddLink](#)  [AddGraphic](#)

Constant Definition	
ID	100
Name	csPropTextFont
Type	String
Default	F1
GetProperty	Yes
SetProperty	Yes

Set the default font to use when printing on the document.
This specification works for [AddTextPos](#) and [AddText](#) methods.

The Standard Type 1 Fonts

The PostScript names of 14 Type 1 fonts, known as the *standard fonts* , are as follows:

Font Names	
Name	Font Type
F1	Courier
F2	Courier-Bold
F3	Courier-Oblique
F4	Courier-BoldOblique
F5	Helvetica
F6	Helvetica-Bold
F7	Helvetica-Oblique
F8	Helvetica-BoldOblique
F9	Times-Roman
F10	Times-Bold
F11	Times-Italic
F12	Times-BoldItalic
F13	Symbol
F14	ZapfDingbats

These fonts, or their font metrics and suitable substitution fonts, are guaranteed to be available to the viewer application. If you wish to use your own True type font you can use it with the [SetTrueTypeFont](#) , this will enable you to add more fonts to your document.

Example:

```
const csPropTextFont = 100
' Write with Helvetica Font
PDF.SetProperty csPropTextFont, " F5
"
PDF.AddText " Hello world "
```

See also:

 [csPropTextSize](#)  [csPropTextColor](#)  [csPropTextUnderLine](#)

Applies to:

 [AddText](#)  [AddTextPos](#)  [AddTextWidth](#)

Constant Definition	
ID	101
Name	csPropTextSize
Type	Integer
Default	10
GetProperty	
SetProperty	

Size of the font to be used, you can use any valid size.

Example:



```
const csPropTextSize = 101
PDF.SetProperty csPropTextSize, 10
PDF.AddText " Hello "
PDF.SetProperty csPropTextSize, 24
PDF.AddText " World<br> "
```

See also:

-  [csPropTextFont](#)
-  [csPropTextColor](#)
-  [csPropTextUnderLine](#)

Applies to:

-  [AddText](#)
-  [AddTextPos](#)
-  [AddTextWidth](#)

Constant Definition	
ID	102
Name	csPropTextAlign
Type	Integer
Default	0
GetProperty	
SetProperty	

You can align the text with the 4 standard options to align text. Left side, this one is the default to be used. To the Right side, you can set the margin of the right side to move the text on the page, you can add columns effects working with both properties. Center on the page. And the last one and cosmetic the Justified.

This property as only effect with the [AddText](#) method. You can not set align properties to the AddTextPos.



Align Property	
Description	Value
Left	0
Right	1
Center	2
Justified	3

Example:

```
const csPropTextAlign = 102
PDF.SetProperty csPropTextAlign, 2
PDF.AddText " Set me on the center of the
page<br> "
PDF.SetProperty csPropTextAlign, 1
PDF.AddText " I'm at right position<br> "
PDF.SetProperty csPropTextAlign, 0
PDF.AddText " Now on the left "
```

Applies to:

-  [AddText](#)
-  [AddTextWidth](#)

Constant Definition	
ID	103
Name	csPropTextColor
Type	String (Color)
Default	#000000
GetProperty	
SetProperty	

You set a color attribute for the actual font. The value can be any hexadecimal number, specified according to the sRGB color space, or one of sixteen color names. Hexadecimal numbers must be prefixed by a "#" character. You can also use color names.

Note: To insert a RGB color use the 3 hexadecimal html notation, for CMYK use 4 hexadecimal html notation, the library will automatically switch from RGB to CYMK or vice versa.

Color Definitions	
Color Name	sRGB Value
Black	#000000
Silver	#C0C0C0
Gray	#808080
White	#FFFFFF
Maroon	#800000
Red	#FF0000
Purple	#800080
Fuchsia	#FF00FF
Green	#008000
Lime	#00FF00
Olive	#080800
Yellow	#FFFF00
Navy	#000080
Blue	#0000FF
Teal	#008080
Aqua	#00FFFFFF

Example:

```
const csPropTextColor = 103
PDF.SetProperty csPropTextColor, "#0000FF"
PDF.AddText " I'm looking blue<br> "
PDF.SetProperty csPropTextColor, "#FF0000"
PDF.AddText " Now red <br> "
PDF.SetProperty csPropTextColor, "Black"
PDF.AddText " What about black name? "
```

See also:

-  [csPropTextFont](#)
-  [csPropTextSize](#)
-  [csPropTextUnderLine](#)

Applies to:

 [AddText](#)  [AddTextPos](#)  [AddTextWidth](#)

Constant Definition	
ID	104
Name	csPropTextUnderLiner
Type	Integer (Bool)
Default	False
GetProperty	✔
SetProperty	✔

Attribute Specifications, suggests that text be rendered as underlined text.

This specification works for [AddTextPos](#) and [AddText](#) methods.

Example:

```
PDF.SetProperty csPropTextUnderLine, 1  
PDF.AddText "This is printed underlined"
```

' Set it without underline

```
PDF.SetProperty csPropTextUnderLine, 0
```

Professional Example:

```
PDF.AddHTML "<u>This is printed underlined</u>"
```

See also:








 [csPropTextFont](#)  [csPropTextSize](#)  [csPropTextColor](#)

Applies to:

 [AddText](#)  [AddTextPos](#)  [AddTextWidth](#)



Constant Definition	
ID	105
Name	csPropTextRender
Type	Integer
Default	0
GetProperty	✔
SetProperty	✔

When drawing the text to the document you can specify the render type. It supports the seven font types from the PDF. But the 4,5,6,7 modes are using filling patterns that are not supported by the aspEasyPDF, so you only have the option to use 0,1,2 and 3.

MODE	EXAMPLE	DESCRIPTION
0		Fill text.
1		Stroke text.
2		Fill, then stroke, text.
3		Neither fill nor stroke text (invisible).
4		Fill text and add to path for clipping (see above).
5		Stroke text and add to path for clipping.
6		Fill, then stroke, text and add to path for clipping.
7		Add text to path for clipping.

Applies to:



 [AddText](#)  [AddTextPos](#)  [AddTextWidth](#)

Constant Definition	
ID	106
Name	csPropTextOverGraph
Type	Boolean
Default	False
GetProperty	
SetProperty	

Allows you to set if the text is drawn in top off all graphics.

Default value is False, that means that the text and graphics are drawn on the same priority as their are being called.

Setting it to True will add all graphics to the back.

Constant Definition	
ID	107
Name	csPropText3DColor
Type	String (Color)
Default	#000000
GetProperty	
SetProperty	

Text colour to use for the shadow color when drawing the 3D text effect. This property is set in combination with the [csPropText3DPos](#) .
Default is " #000000 ", black color.



Color Definitions	
Color Name	sRGB Value
Black	#000000
Silver	#C0C0C0
Gray	#808080
White	#FFFFFF
Maroon	#800000
Red	#FF0000
Purple	#800080
Fuchsia	#FF00FF
Green	#008000
Lime	#00FF00
Olive	#080800
Yellow	#FFFF00
Navy	#000080
Blue	#0000FF
Teal	#008080
Aqua	#00FFFF

See also:

 [csPropText3DPos](#)

Applies to:

 [AddTextPos](#)

Constant Definition	
ID	108
Name	csPropText3DPos
Type	String
Default	
GetProperty	
SetProperty	

If you want to draw the text with a 3D effect then set this property with the X,Y offset you wish to set the shadow. You can also set the shadow color with [csPropText3DColor](#) .

Note: This property only works with the AddTextPos it may be implemented on the [AddText](#) mehod on future releases.

Example:



```
PDF.SetProperty csPropText3DColor, " #0000FF "
' Enable Shadow X Y back offset
PDF.SetProperty csPropText3DPos, " 0.5,0.5 "
' Write the text
PDF.AddTextPos 10,10," This is a text with 3D shadow
"
' Disable Shadow
PDF.SetProperty csPropText3DPos, " 0,0 "
```

See also:

 [csPropText3DColor](#)

Applies to:

 [AddTextPos](#)

Constant Definition	
ID	109
Name	csPropTextAngle
Type	Float
Default	0
GetProperty	
SetProperty	

If you want to rotate a text use this property to specify the the angle to rotate in radians. This property is for use only with the AddTextPos method as it doesn't control any margins and break pages. You can combine other properties that affects the addTextPos without restrictions.

From version 1.70 you can now specify angles values. To specify angles you must use a specific symbol to tell the library that the values is in angles (° or o) and not in radians.

Example:



```
' Rotate 45°  
PDF.SetProperty csPropTextAngle, " 0.5 "  
' Write the text  
PDF.AddTextPos 100,200,"This is going up..."
```

Example in angle from version 1.70:

```
' Rotate 90°  
PDF.SetProperty csPropTextAngle, " 90° "  
' If you can not use the symbol ° then use the letter o in lowercase  
' PDF.SetProperty csPropTextAngle, " 90o "  
' Write the text  
PDF.AddTextPos 100,200," This is going up... "
```

Applies to:

 [AddTextPos](#)

Constant Definition	
ID	112
Name	csPropTextEntityConv
Type	Integer (Bool)
Default	True
GetProperty	
SetProperty	
From version	1.53

When using AddText and AddTextPos you can use html entities to specify special characters to be used on the text, if you want to disable this feature you must set the constant to 0, false.

ISO Latin-1 Character Set

The following table contains the complete ISO Latin-1 character set, corresponding to the first 256 entries of the Unicode character repertoire in Internet Explorer 4.0. The table describes each character, its decimal code, and its named entity reference for HTML, and also provides a brief description.

Character	Decimal code	Named entity	Description
---	�	---	Unused
---		---	Unused
---		---	Unused
---		---	Unused
---		---	Unused
---		---	Unused
---		---	Unused
---		---	Unused
---		---	Unused
---			---	Horizontal tab

	---	Line feed
---		---	Unused
---		---	Unused
---		---	Carriage Return
---		---	Unused
---		---	Unused
---		---	Unused
---		---	Unused
---		---	Unused
---		---	Unused
---		---	Unused
---		---	Unused
---		---	Unused
---		---	Unused
---		---	Unused
---		---	Unused

---		---	Unused
---		---	Unused
---		---	Unused
---		---	Unused
---		---	Unused
---		---	Unused
	 	---	Space
!	!	---	Exclamation mark
"	"	"	Quotation mark
#	#	---	Number sign
\$	$	---	Dollar sign
%	%	---	Percent sign
&	&	&	Ampersand
'	'	---	Apostrophe
((---	Left parenthesis
))	---	Right parenthesis
*	*	---	Asterisk
+	+	---	Plus sign
,	,	---	Comma
-	-	---	Hyphen
.	.	---	Period (fullstop)
/	/	---	Solidus (slash)
0	0	---	Digit 0
1	1	---	Digit 1
2	2	---	Digit 2
3	3	---	Digit 3
4	4	---	Digit 4
5	5	---	Digit 5
6	6	---	Digit 6
7	7	---	Digit 7
8	8	---	Digit 8
9	9	---	Digit 9
:	:	---	Colon
;	;	---	Semicolon
<	<	<	Less than
=	=	---	Equals sign
>	>	>	Greater than
?	?	---	Question mark
@	@	---	Commercial at
A	A	---	Capital A
B	B	---	Capital B
C	C	---	Capital C
D	D	---	Capital D
E	E	---	Capital E
F	F	---	Capital F
G	G	---	Capital G
H	H	---	Capital H

I	I	---	Capital I
J	J	---	Capital J
K	K	---	Capital K
L	L	---	Capital L
M	M	---	Capital M
N	N	---	Capital N
O	O	---	Capital O
P	P	---	Capital P
Q	Q	---	Capital Q
R	R	---	Capital R
S	S	---	Capital S
T	T	---	Capital T
U	U	---	Capital U
V	V	---	Capital V
W	W	---	Capital W
X	X	---	Capital X
Y	Y	---	Capital Y
Z	Z	---	Capital Z
[[---	Left square bracket
\	\	---	Reverse solidus (backslash)
]]	---	Right square bracket
^	^	---	Caret
_	_	---	Horizontal bar (underscore)
`	`	---	Acute accent
a	a	---	Small a
b	b	---	Small b
c	c	---	Small c
d	d	---	Small d
e	e	---	Small e
f	f	---	Small f
g	g	---	Small g
h	h	---	Small h
i	i	---	Small i
j	j	---	Small j
k	k	---	Small k
l	l	---	Small l
m	m	---	Small m
n	n	---	Small n
o	o	---	Small o
p	p	---	Small p
q	q	---	Small q
r	r	---	Small r
s	s	---	Small s
t	t	---	Small t
u	u	---	Small u
v	v	---	Small v
w	w	---	Small w



x	x	---	Small x
y	y	---	Small y
z	z	---	Small z
{	{	---	Left curly brace
	|	---	Vertical bar
}	}	---	Right curly brace
~	~	---	Tilde
•		---	Unused
€	€	---	Unused
	 	 	Nonbreaking space
¡	¡	¡	Inverted exclamation
¢	¢	¢	Cent sign
£	£	£	Pound sterling
¤	¤	¤	General currency sign
¥	¥	¥	Yen sign
¦	¦	¦ or	Broken vertical bar
§	§	§	Section sign
¨	¨	¨ or ¨	Diæresis / Umlaut
©	©	©	Copyright
ª	ª	ª	Feminine ordinal
«	«	«	Left angle quote, guillemot left
¬	¬	¬	Not sign
-	­	­	Soft hyphen
®	®	®	Registered trademark
ˆ	¯	ˆ or &hibar;	Macron accent
°	°	°	Degree sign
±	±	±	Plus or minus
²	²	²	Superscript two
³	³	³	Superscript three
´	´	´	Acute accent
µ	µ	µ	Micro sign
¶	¶	¶	Paragraph sign
·	·	·	Middle dot
¸	¸	¸	Cedilla
¹	¹	¹	Superscript one
º	º	º	Masculine ordinal
»	»	»	Right angle quote, guillemot right
¼	¼	¼	Fraction one-fourth
½	½	½	Fraction one-half
¾	¾	¾	Fraction three-fourths
¿	¿	¿	Inverted question mark
À	À	À	Capital A, grave accent
Á	Á	Á	Capital A, acute accent
Â	Â	Â	Capital A, circumflex
Ã	Ã	Ã	Capital A, tilde
Ä	Ä	Ä	Capital A, diæresis / umlaut
Å	Å	Å	Capital A, ring

Æ	Æ	Æ	Capital AE ligature
Ç	Ç	Ç	Capital C, cedilla
È	È	È	Capital E, grave accent
É	É	É	Capital E, acute accent
Ê	Ê	Ê	Capital E, circumflex
Ë	Ë	Ë	Capital E, diæresis / umlaut
Ì	Ì	Ì	Capital I, grave accent
Í	Í	Í	Capital I, acute accent
Î	Î	Î	Capital I, circumflex
Ï	Ï	Ï	Capital I, diæresis / umlaut
Ð	Ð	Ð	Capital Eth, Icelandic
Ñ	Ñ	Ñ	Capital N, tilde
Ò	Ò	Ò	Capital O, grave accent
Ó	Ó	Ó	Capital O, acute accent
Ô	Ô	Ô	Capital O, circumflex
Õ	Õ	Õ	Capital O, tilde
Ö	Ö	Ö	Capital O, diæresis / umlaut
×	×	×	Multiply sign
Ø	Ø	Ø	Capital O, slash
Ù	Ù	Ù	Capital U, grave accent
Ú	Ú	Ú	Capital U, acute accent
Û	Û	Û	Capital U, circumflex
Ü	Ü	Ü	Capital U, diæresis / umlaut
Ý	Ý	Ý	Capital Y, acute accent
Þ	Þ	Þ	Capital Thorn, Icelandic
ß	ß	ß	Small sharp s, German sz
à	à	à	Small a, grave accent
á	á	á	Small a, acute accent
â	â	â	Small a, circumflex
ã	ã	ã	Small a, tilde
ä	ä	ä	Small a, diæresis / umlaut
å	å	å	Small a, ring
æ	æ	æ	Small ae ligature
ç	ç	ç	Small c, cedilla
è	è	è	Small e, grave accent
é	é	é	Small e, acute accent
ê	ê	ê	Small e, circumflex
ë	ë	ë	Small e, diæresis / umlaut
ì	ì	ì	Small i, grave accent
í	í	í	Small i, acute accent
î	î	î	Small i, circumflex
ï	ï	ï	Small i, diæresis / umlaut
ð	ð	ð	Small eth, Icelandic
ñ	ñ	ñ	Small n, tilde
ò	ò	ò	Small o, grave accent
ó	ó	ó	Small o, acute accent
ô	ô	ô	Small o, circumflex

õ	õ	õ	Small o, tilde
ö	ö	ö	Small o, diæresis / umlaut
÷	÷	÷	Division sign
ø	ø	ø	Small o, slash
ù	ù	ù	Small u, grave accent
ú	ú	ú	Small u, acute accent
û	û	û	Small u, circumflex
ü	ü	ü	Small u, diæresis / umlaut
ý	ý	ý	Small y, acute accent
þ	þ	þ	Small thorn, Icelandic
ÿ	ÿ	ÿ	Small y, diæresis / umlaut

Applies to:

 [AddText](#)  [AddTextPos](#)  [AddTextWidth](#)

Constant Definition	
ID	115
Name	csPropTextVertSpace
Type	Float
Default	0
GetProperty	
SetProperty	



You can set the vertical space between two lines. By default is 0

See also:

 [csPropCharSpacing](#)  [csPropWordSpacing](#)

Applies to:

 [AddText](#)  [AddTextPos](#)  [AddTextWidth](#)

Constant Definition	
ID	111
Name	csPropWordSpacing
Type	Float
Default	0
GetProperty	
SetProperty	

Specify the space between each word. By default is 0 that has no effect.

With this property you can set looks like this:

Hello World

Hello World

Hello World

See also:

 [csPropCharSpacing](#)  [csPropTextVertSpace](#)

Applies to:

 [AddText](#)  [AddTextPos](#)  [AddTextWidth](#)