



PHP Pdf Class

Native PDF document creation with PHP

released under the terms of the MIT license

[Contributors](#)

Version 0.12.66
php 8.2.1



FORK ON GITHUB.COM

R&OS

<https://github.com/rospdf/>

Table of Contents

Introduction	1
Changelog	1
Use	2
Extensions	2
EZPDF Class Extension	3
Cezpdf	3
ezSetMargins	3
ezSetCmMargins	4
ezNewPage	4
ezColumnsStart	4
ezColumnsStop	4
ezInsertMode	4
ezSetY	4
ezSetDy	4
ezStartPageNumbers	5
ezWhatPageNumber	6
ezGetCurrentPageNumber	6
ezStopPageNumbers	7
ezTable	7
ezText	8
ezImage	9
ezOutput	9
ezStream	9
Inline codes	10
Base Class Functions	11
addText	11
setColor	11
setStrokeColor	11
setLineStyle	11
line	12
curve	12
ellipse	12
partEllipse	13
polygon	13
rectangle	14
filledRectangle	14
newPage	14
getFirstPageld	14
stream	14
getFontHeight	14
getFontDescender	14
getTextWidth	15
saveState	15
restoreState	15
openObject	15

reopenObject	15
closeObject	15
addObject	15
stopObject	15
addInfo	15
setPreferences	16
addImage	16
addJpegFromFile	16
addPngFromFile	16
output	16
openHere	16
selectFont	16
setFontFamily	17
setEncryption	18
addLink	18
addInternalLink	18
addDestination	18
transaction	19
Misc	19
Callback functions	19
Units	20

Introduction

This class is designed to provide a **non-module**, non-commercial alternative to dynamically

This document describes the possible useful calls to the class, the readme.php file (which will

Note that this document was generated using the demo script 'readme.php' which came with

Changelog

0.12.65

- Fix PHP 8.1 conversion from float deprecation #168

0.12.64

- Compatibility with PHP 8.1 #164

0.12.63

- Fix notice with PHP 7.4 #146

0.12.62

- Fixed repeating background image when using ezColumnsStart

0.12.61

- Fixed issue #139
- Fixed full justification during line breaks using \$pdf->ezText()

0.12.60

- Compatibility for php 7.4 using PR #136
- Slightly improved *.afm font loading for compatibility reason

0.12.58

- Recovered \$test option in addText() method #128
- Fixed broken PDF 1.3 standard when using callbacks
- Proper calculate full justification on last line

Please refer to <https://github.com/rospdf/pdf-php/releases> for all previous changes

Use

It is free for use for any purpose (public domain), though we would prefer it if you retain the
Note that there is no guarantee or warranty, implied or otherwise with this class, or the

Extensions

In order to create simple documents with ease a class extension called 'ezPdf' has been
The functions of ezpdf are described in the next section, though as it is a class extension, all

EZPDF Class Extension

(note that the creation of this document in readme.php was converted to ezpdf with the
It is anticipated that in practise only the simplest of documents will be able to be created

Cezpdf ([paper='a4'],[orientation='portrait'],[type='none'],[options=[]])

This is the constructor function, and allows the user to set up a basic page without having to
Valid values for paper are listed below, a two or four member array can also be passed, a
Starting ezpdf with the code below will create an a4 portrait document.

\$paper The valid values for the paper (thanks to the work of Nicola Asuni) are:
'4A0', '2A0', 'A0', 'A1', 'A2', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8', 'A9', 'A10', 'B0', 'B1', 'B2', 'B3', 'B4',

\$orientation 'portrait' and 'landscape' can be used

\$type following types are possible: 'none' | 'color' | 'image'

\$options if type is different then 'none' \$options can be set as follows

if \$type is set to 'color'

\$options['r'] = red-component of backgroundcolour (0 <= r <= 1)

\$options['g'] = green-component of backgroundcolour (0 <= g <= 1)

\$options['b'] = blue-component of backgroundcolour (0 <= b <= 1)

if \$type is set to 'image' then the \$options array has other attributes

\$options['img'] = location of image file; URI's are allowed if allow_url_open is enabled

\$options['width'] = width of background image; default is width of page

\$options['height'] = height of background image; default is height of page

\$options['xpos'] = horizontal position of background image; default is 0

\$options['ypos'] = vertical position of background image; default is 0

\$options['repeat'] = repeat image horizontally (1), repeat image vertically (2) or full in
highly recommend to set this->hashed to true when using repeat function

This document shows you the 'color' type with following:

```
$pdf = new Cezpdf('a4','portrait','color',[0.8,0.8,0.8]);
```

If you want to get started in an easy manner, then here is the 'hello world' program:

```
<?php
include ('class.ezpdf.php');
$pdf = new Cezpdf();
$pdf->selectFont('Helvetica');
$pdf->ezText('Hello World!',50);
$pdf->ezStream();
?>
```

ezSetMargins (top,bottom,left,right)

Sets the margins for the document, this command is optional and they will all be set to 30 by

ezSetCmMargins (top,bottom,left,right)

Sets the margins for the document using centimeters

ezNewPage ()

Starts a new page. This is subtly different to the newPage command in the base class as it

ezColumnsStart ([options])

This will start the text flowing into columns, *options* is an array which contains the control

The options are:

'gap' => the gap between the columns

'num' => the number of columns.

Both options (and the array itself) are optional, if missed out then the defaults are gap=10,

Example calls could be (you would use only one of these):

```
$pdf->ezColumnsStart();  
$pdf->ezColumnsStart(['num'=>3]);  
$pdf->ezColumnsStart(['num'=>3, 'gap'=>2]);  
$pdf->ezColumnsStart(['gap'=>20]);
```

ezColumnStop is used to stop multi-column mode.

ezColumnsStop

This stops multi-column mode, it will leave the writing point at whatever level it was at, it is

ezInsertMode ([status=1,\$pageNum=1,\$pos='before'])

This command can be used to stop and start page insert mode, while this mode is on then

All subsequent pages added with ezNewPage are then inserted within the document

Insertion page is ended by calling this command with 'status'=0, and further pages are added

ezSetY (y)

Positions the ezpdf writing pointer to a particular height on the page, don't forget that pdf

ezSetDy (dy [,mod])

Changes the vertical position of the writing point by a set amount, so to move the pointer 10

```
ezSetDy(-10)
```

If this movement makes the writing location below the bottom margin, then a new page will

The optional parameter 'mod' can be set to the value 'makeSpace', which means that if a

```
ezSetDy(-100, 'makeSpace')
```

guarantees that there will be 100 units of space above the final writing point.

ezStartPageNumbers (x,y,size,[pos],[pattern],[num]) = setNum

Add page numbers on the pages from here, place then on the 'pos' side of the coordinates

Use the given 'pattern' for display, where {PAGENUM} and {TOTALPAGENUM} are replaced

If \$num is set, then make the first page this number, the number of total pages will be

the following code produces a seven page document, numbered from the second page

```
$pdf = new Cezpdf();
$pdf->selectFont('Helvetica');
$pdf->ezNewPage();
$pdf->ezStartPageNumbers(300,500,20,'',' ',1);
$pdf->ezNewPage();
$pdf->ezNewPage();
$pdf->line(300,400,300,600); // line drawn to check 'pos' is working
$pdf->ezNewPage();
$pdf->ezNewPage();
$pdf->ezNewPage();
$pdf->ezStopPageNumbers();
$pdf->ezStream();
```

This function was modified in version 009 to return a page numbering set number (setNum in

Here is a more complex example:

```
$pdf->selectFont('Helvetica');
$pdf->ezNewPage();
$i=$pdf->ezStartPageNumbers(300,500,20,'',' ',1);
$pdf->ezNewPage();
$pdf->ezNewPage();
$pdf->ezStopPageNumbers(1,1,$i);
$pdf->ezNewPage();
$i=$pdf->ezStartPageNumbers(300,500,20,'',' ',1);
$pdf->ezNewPage();
$pdf->ezNewPage();
$pdf->ezStopPageNumbers(1,1,$i);
$pdf->ezNewPage();
$i=$pdf->ezStartPageNumbers(300,500,20,'',' ',1);
$pdf->ezNewPage();
$pdf->ezNewPage();
$pdf->setColor(1,0,0);
$pdf->ezNewPage();
$pdf->ezStopPageNumbers(1,1,$i);
$pdf->ezNewPage();
$i=$pdf->ezStartPageNumbers(300,500,20,'',' ',1);
$pdf->ezNewPage();
$pdf->ezNewPage();
$pdf->ezNewPage();
$j=$pdf->ezStartPageNumbers(300,400,20,'',' ',1);
$k=$pdf->ezStartPageNumbers(300,300,20,'',' ',1);
```



```

$pdf->ezNewPage();
$pdf->ezNewPage();
$pdf->ezNewPage();
$pdf->ezStopPageNumbers(1,1,$i);
$pdf->ezNewPage();
$pdf->ezNewPage();
$pdf->ezStopPageNumbers(1,1,$j);
$pdf->ezStopPageNumbers(0,1,$k);
$pdf->ezNewPage();
$pdf->ezNewPage();

```

This will create a document with 23 pages, the numbering shown on each of the pages is:

page	contents
1	blank
2	1 of 3
3	2 of 3
4	3 of 3
5	1 of 3
6	2 of 3
7	3 of 3
8	1 of 3
9	2 of 3
10	3 of 3
11	4 of 4
12	1 of 8
13	2 of 8
14	3 of 8
15	4 of 8
16	5 of 8, 1 of 6, 1 of 8
17	6 of 8, 2 of 6, 2 of 8
18	7 of 8, 3 of 6, 3 of 8
19	8 of 8, 4 of 6, 4 of 8
20	5 of 6, 5 of 8
21	6 of 6, 6 of 8
22	blank
23	blank

ezWhatPageNumber (pageNum,[setNum]) = num

Returns the number of a page within the specified page numbering system.

'pageNum' => the absolute number of the page within the document (this is based on the 'setNum' => the page numbering set, returned from the *ezStartPageNumbers* command.

ezGetCurrentPageNumber

return the page number of the current page

ezStopPageNumbers ([stopTotal],[next],[setNum])

In version 009 this function was enhanced to include a number of extra parameters:

'stopTotal' => 0 or 1 (default 0), stops the totaling for the page numbering set. So for example

'next' => 0 or 1, stops on the next page, not this one.

'setNum' => (defaults to 0) define which set number is to be stopped, this is the number

ezTable (array data,[array cols],[title],[array options]) = y

\$data The easy way to throw a table of information onto the page, can be used with just the

The table will start writing from the current writing point, and will proceed until the all the data

The return value from the function is the y-position of the writing pointer after the table has

since 0.12.27 \$data can contain the follow keys to colorize cells

```
'[columnName]Fill' => [r,g,b] // used to fill a cell background color  
'[columnName]Color' => [r,g,b] // used to color the cell text
```

The other options are described here below

\$cols (optional) is an associative array, the keys are the names of the columns from \$data to

\$title (optional) is the title to be put on the top of the table

\$options is an associative array which can contain:

'showHeadings' => 0 or 1 (enable or disable head line)

'shaded'=> 0,1,2, default is 1 (alternate lines are shaded)

0->no shading, 2->both sets are shaded

'shadeCol' => (r,g,b) array, defining the colour of the shading, default is (0.8,0.8,0.8)

'shadeCol2' => (r,g,b) array, defining the colour of the shading of the second set, default is

'fontSize' => 10

'textCol' => (r,g,b) array, text colour

'titleFontSize' => 12

'rowGap' => 2 , the space between the text and the row lines on each row

'colGap' => 5 , the space between the text and the column lines in each column

'lineCol' => (r,g,b) array, defining the colour of the lines, default, black.

'xPos' => 'left','right','center','centre',or coordinate, reference coordinate in the x-direction

'xOrientation' => 'left','right','center','centre', position of the table w.r.t 'xPos'. This entry is to

'width' => , the exact width of the table, the cell widths will be adjusted to give the table this

'maxWidth' => , the maximum width of the table, the cell widths will only be adjusted if the

'cols' => [=> array('justification'=>'left', 'width'=>100, 'link'=>, 'bgcolor'=> (r,g,b)), =>....) allow

'innerLineThickness' => , the thickness of the inner lines, defaults to 1

'outerLineThickness' => , the thickness of the outer lines, defaults to 1

'protectRows' => , the number of rows to keep with the heading, if there are less than this on

'nextPageY'=> true or false (eg. 0 or 1) Sets the same Y position of the table for all future

'evenColumns' => 0, 1, 3, Set set all columns to the same widths, version 0.12.44

'evenColumnsMin' => , the minimum width a column should have

0.12.9:

'shadeHeadingCol'=>(r,g,b) array, defining the background color of headings, default is empty

0.12.11:

'gridlines'=> EZ_GRIDLINE_* default is EZ_GRIDLINE_DEFAULT, overrides 'showLines' to
'alignHeadings' => 'left','right','center'

0.12.27:

'evenColumns' => 0,1 make all column width the same size

Below is an example output of ezTable containing the following \$data

```
$data = [
  ['num'=>1,'name'=>'gandalf','type'=>'wizard' ]
  ,['num'=>2,'name'=>'bilbo','type'=>'hobbit','url'=>'https://github.com/rosp
  ,['num'=>3,'name'=>'frodo','type'=>'hobbit' ]
  ,['num'=>4,'name'=>'saruman','type'=>'bad
  ,['num'=>5,'name'=>'sauron','type'=>'really bad dude', 'typeFill' =>
];

$pdf->ezTable($data,$cols,'', [
  'gridlines'=> EZ_GRIDLINE_DEFAULT,
  'shadeHeadingCol'=> [0.6,0.6,0.5],
  'alignHeadings'=>'center',
  'width'=>400,
  'cols'=> [
    'name'=> ['bgcolor'=> [0.9,0.9,0.7] ],
    'type'=> ['bgcolor'=> [0.6,0.4,0.2] ]
  ]
];
```

Number	Name	Type
1	gandalf	wizard
2	bilbo	hobbit
3	frodo	hobbit
4	saruman	bad dude
5	sauron	really bad dude

Please refer to examples/tables.php for more demos

ezText (text,[size],[array options]) = y

This is designed for putting blocks of text onto the page. It will add a string of text to the

The return value from the function (y) is the vertical position on the page of the writing

possible options are:

'left'=> number, gap to leave from the left margin

'right'=> number, gap to leave from the right margin

'aleft'=> number, absolute left position (overrides 'left')

'aright'=> number, absolute right position (overrides 'right')

'justification' => 'left','right','center','centre','full'

only set one of the next two items (leading overrides spacing)

'leading' => number, defines the total height taken by the line, independent of the font height.
'spacing' => a real number, though usually set to one of 1, 1.5, 2 (line spacing as used in

This function now supports the use of underlining markers (<u> </u>), these are

ezImage (image,[padding],[width],[resize],[justification], [angle],[array border])

This function makes it much easier to simply place an image (either jpeg or png) within the

This function can be used by simply supplying the name of an image file as the first

The arguments are:

\$image is a string containing the filename and path of the jpeg or png image you want to

\$padding (optional) is the number of page units that will pad the image on all sides. The

\$width (optional) is the width of the image on the page. The default is to use the actual

\$resize (optional) can be one of 'full', 'width', or 'none'. The default value is 'full'.

The value 'none' means that an image will not be sized up to fit the width of a column and will

The value 'width' behaves the same as 'none' with the exception that images will be resized

The value 'full' behaves the same as 'width' with the exception that images will be resized

\$justification (optional) determines the horizontal position of the image within the current

\$angle (optional) allows you to rotate the image

\$border (optional) is an array which specifies the details of the border around the image.

\$border['width'] is the width of the border. The default is 1.

\$border['cap'] is the cap type as specified in `setLineStyle`. The default is 'round'.

\$border['join'] is the join type as specified in `setLineStyle`. The default is 'round'.

\$border['color'] is an associative array for specifying the line color of the border.

The values are as specified in `setStrokeColor` and should be assigned to:

\$border['color']['red'], **\$border['color']['green']** and **\$border['color']['blue']** respectively.

ezOutput ([debug])

Very similar to the output function from the base class, but performs any closing tasks that

If you are using `ezpdf`, then you should use this function, rather than the one from the base

ezStream ([options])

Very similar to the stream function from the base class (all the same options, see later in this

If you are using ezpdf, then you should use this function, rather than the one from the base

Inline codes

There are a few callback functions (see callback functions) which are contained within the

Underline

Though underlining is supported in the ezPdf class by using the <u> directive, if you use the base class functions to add text (such as addtext) then this wont work, instead

(Note that what the ezPdf class does internally is convert the <u> and </u>

So as an example, this code adds some text with two pieces of underlining, one done each

```
$pdf->ezText('The <u>quick brown</u> fox, <c:uline>is
```

The quick brown fox, is sick of jumping the lazy dog

Links to URLs

If you are adding links to a document, it is quite tricky to figure out where to put the rectangle

The *alink* callback allows for simple insertion of links, the format is:

```
<c:alink:your_url_here>text to be clickable</c:alink>
```

So as an example:

```
$pdf->ezText('<c:alink:https://github.com/rospdf/pdf-php/ pdf
```

[R&OS pdf class](#)

Links within the document

There is a directive similar to *alink*, but designed for linking within the document, this is the

It is similar to alink except that instead of providing a URL the label of a pre-created

```
<c:ilink:destination_label>text to be clickable</c:ilink>
```

```
// place required to be marked
$pdf->addDestination('xxxxyyzzz','Fit');
// add lots of stuff, new pages etc, then...
$pdf->ezText('<c:ilink:xxxxyyzzz>R&OS pdf class</c:ilink>');
```

Click here to go to the 5th item on the table of contents.

Note that the code for the example and the actual one shown are not identical for technical

Base Class Functions

addText (x,y,size,text[,width=0][,justification='left'][,angle=0][,wordspace=0][,test=0)

Add the text at a particular location on the page, noting that the origin on the axes in a pdf 'adjust', gives the value of units to be added to the width of each space within the text. This is The text stream can now (version 006) contain directives to make the text bold and/or italic.

```
<b>bold text</b>
<i>italic text</i>
<b><i>bold italic text</i></b>
```

Note that there can be no spaces within the directives, and that they must be in lower case.

Especially **bold** and *italic* requires the style of the font being used

If you wish to print an actual '<', most of the time it would cause no problem, except in the

```
$pdf->addText(150,$y,10,"the quick brown fox <b>jumps</b>
```

the quick brown fox **jumps** over the lazy dog!

setColor (r,g,b,[force=0])

Set the fill colour to the r,g,b triplet, each in the range 0->1.

If force is set, then the entry will be forced into the pdf file, otherwise it will only be put in if it

setStrokeColor (r,g,b,[force=0])

Set the stroke color, see the notes for the fill color.

setLineStyle ([width],[cap],[join],[dash],[phase])

This sets the line drawing style.

width, is the thickness of the line in user units

cap is the type of cap to put on the line, values can be 'butt','round','square' where the join can be 'miter', 'round', 'bevel'

dash is an array which sets the dash pattern, is a series of length values, which are the for example: (2) represents 2 on, 2 off, 2 on , 2 off ...

(2,1) is 2 on, 1 off, 2 on, 1 off.. etc

phase is a modifier on the dash pattern which is used to shift the point at which the pattern

```
$pdf->setLineStyle(1);
```

```
$pdf->setLineStyle(5);
```

[illegible]

Note that the code shown with each of these lines is just the line style command, the drawing

```
# line (x1,y1,x2,y2)
```

Draw a line from (x1,y1) to (x2,y2), set the line width using setLineStyle.

```
# curve (x0,y0,x1,y1,x2,y2,x3,y3)
```

Draw a Bezier curve. The end points are $(x_0, y_0) \rightarrow (x_3, y_3)$, and the control points are the other

Bezier curves are neat, but probably not for the novice. The **ellipse** command uses a series

The distinctive feature of these curves is that the curve is guaranteed to lie within the 4 sided

`$pdf->curve(200,$y+40,250,$y+5,350,$y,400,$y+45);`

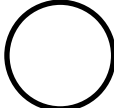
Note that the Bezier curve is a tangent to the line between the control points at either end of

```
# ellipse (x0,y0,r1,[r2=0],[angle=0],[nSeg=8])
```

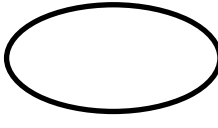
Draw an ellipse, centred at (x_0, y_0) , with radii (r_1, r_2) , oriented at 'angle' (anti-clockwise), and

If r_2 is left out, or set to zero, then it is assumed that a circle is being drawn.

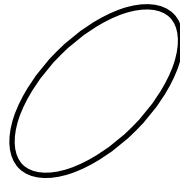
```
$pdf->ellipse(300,$y+25,20);
```



```
$pdf->ellipse(300,$y+25,40,20);
```

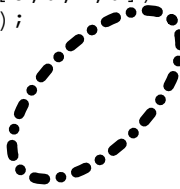


```
$pdf->ellipse(300,$y+25,40,20,45);
```



Of course the previous line style features also apply to these lines

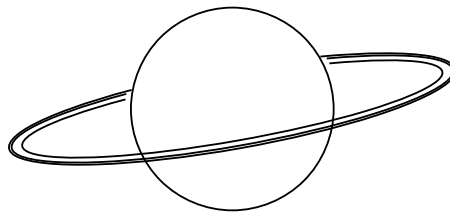
```
$pdf->setLineStyle(4,'round','', [0,6,4,6]);  
$pdf->ellipse(300,$y+25,40,20,45);
```



partEllipse (x,y,a1,a2,r1 [,r2] [,angle] [,nSeg])

Draw a part ellipse (or circle), draw an ellipse centered on (x,y) from angle 'a1' to angle 'a2'

```
$pdf->ellipse(300,$y+25,38);  
$pdf->partEllipse(300,$y+25,119,421,80,12,10);  
$pdf->partEllipse(300,$y+25,117,423,84,14,10);  
$pdf->partEllipse(300,$y+25,115,425,85,15,10);
```



polygon (p,np,[f=0])

Draw a polygon, where there are np points, and p is an array containing
If f=1 then fill the area.

```
$pdata = [200,10,400,20,300,50,150,40];  
$pdf->polygon($pdata,4);
```



```
$pdf->polygon($pdata,4,1);
```



```
$pdf->setColor(0.9,0.9,0.9);  
$pdf->polygon($pdata,4,1);
```




rectangle (x1,y1,width,height)

Build a rectangle with no background, only borders

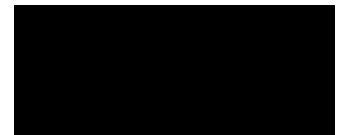
```
$pdf->rectangle(400, $pdf->y, 120,50);
```



filledRectangle (x1,y1,width,height)

Build a rectangle with filled color.

```
$pdf->filledRectangle(400, $pdf->y, 120, 50);
```



newPage ([insert,id,pos]) = id

Starts a new page and returns the id of the page contents, this can be safely ignored, but

The command is usually used without any of the options to simply add a new page to the end

getFirstPageId () = id

A related command is this which returns the id of the first page, this page is created during

stream ([array options])

Used for output, this will set the required headers and output the pdf code.

The options array can be used to set a number of things about the output:

'Content-Disposition'=>'filename' sets the filename, though not too sure how well this will
'Accept-Ranges'=>1 or 0 - if this is not set to 1, then this header is not included, off by
'compress'=> 1 or 0 - apply content stream compression, this is on (1) by default.

getFontHeight (size) = height

Returns the height of the current font, in the given size. This is the distance from the bottom

getFontDescender (size) = height

Returns a number which is the distance that the descender goes beneath the Baseline, for a

getTextWidth (size,text) = width

Returns the width of the given text string at the given size.

saveState ()

Save the graphic state.

restoreState ()

Restore a saved graphics state.

openObject () = id

Start an independent object. This will return an object handle, and all further writes to a page

reopenObject (id)

Makes the point of current content insertion the numbered object, this 'id' must have been
This will allow the user to add information to previous pages, as long as they have stored the

closeObject ()

Close the currently open object. Further writes will now go to the current page.

addObject (id,[options='add'])

Add the object specified by id to the current page (default). If a string is supplied in options,
'add' - add to the current page only.

'all' - add to every page from the current one on.

'odd' - add to all odd numbered pages from now on.

'even' - add to all even numbered pages from now on.

'next' - add to just the next page.

'nextodd' - add to all odd numbered pages from the next one.

'nexteven' - add to all even numbered pages from the next one.

stopObject (id)

If the object (id) has been appearing on pages up to now, then stop it, this page will be the

addInfo (label,value)

Add document information, the valid values for label are:

Title, Author, Subject, Keywords, Creator, Producer, CreationDate, ModDate, Trapped

modified in version 003 so that 'label' can also be an array of key->value pairs, in which case

setPreferences (label,value)

Set some document preferences, the valid values for label are:

HideToolbar, HideMenuBar, HideWindowUI, FitWindow, CenterWindow,

modified in version 003 so that 'label' can also be an array of key->value pairs, in which case

addImage (img,x,y,w,[h],[quality=75],[angle=0])

Add an image to the document, this feature needs some development. But as it stands, img

The image will be placed with its lower left corner at (x,y), and w and h refer to page units,

addJpegFromFile (imgFileName,x,y,w,[h],[angle=0])

Add a JPEG image to the document, this function does not require the GD functionality, so be more reliable and better quality. The syntax of the command is similar to the above

x,y are the position of the lower left corner of the image, and w,h are the width and height.

addPngFromFile (imgFileName,x,y,w,[h],[angle=0])

Similar to *addJpegFromFile*, but for PNG images.

output ([debug=0]) = a

As an alternative to streaming the output directly to the browser, this function simply returns

If the 'debug' parameter is set to 1, then the only effect at the moment is that the

openHere (style[,a][,b][,c])

Make the document open at a specific page when it starts. The page will be the one which is

The style option will define how it will look when open, valid values are:

(where the extra parameters will be used to supply any values specified for the style chosen)

'XYZ' left, top, zoom *open at a particular coordinate on the page, with a given zoom factor*

'Fit' *fit the page to the view*

'FitH' top *fit horizontally, and at the vertical position given*

'FitV' left *fit vertically, and at the horizontal position given*

'FitB' *fit the bounding box of the page into the viewer*

'FitBH' top *fit the width of the bounding box to the viewer and set at the vertical position given*

'FitBV' left *fit bounding box vertically and set given horizontal position*

selectFont (fontName [,encoding = ""] [, set = 1] [,subsetFont = false])

This selects a font to be used from this point in the document. Errors can occur if some of the

In all versions prior to 0.12 it was required to "convert" ttf fonts into either *.AFM or *.UFM for Since version >= 0.12.0 R&OS pdf class natively reads the information from the TTF font by It also supports font subsetting (>= 0.11.8) by using TTFsubset.php. Thanks to Thanos

The encoding directive is still experimental and no guarantee its working at all in version >= Note that the encoding directive will be effective **only the first time** that a font is selected,

```
// use a Times-Roman font with MacExpertEncoding
$pdf->selectFont('Times-Roman','MacExpertEncoding');
// this next line should be equivalent
$pdf->selectFont('Times-Roman',['encoding'=>'MacExpertEncoding']);

// setup the helvetica font for use with german characters
$diff=array(196=>'Adieresis',228=>'adieresis',
            214=>'Odieresis',246=>'odieresis',
            220=>'Udieresis',252=>'udieresis',
            223=>'germandbls');
$pdf->selectFont('Helvetica',
               ,array('encoding'=>'WinAnsiEncoding'
                   , 'differences'=>$diff));
```

setFontFamily (family,options)

This function defines the relationship between the various fonts that are in the system, so

It maintains a set of arrays which give the alternatives for the base fonts. The defaults that

'Helvetica'

```
'b'=>'Helvetica-Bold'
'i'=>'Helvetica-Oblique'
'bi'=>'Helvetica-BoldOblique'
'ib'=>'Helvetica-BoldOblique'
```

'Courier'

```
'b'=>'Courier-Bold'
'i'=>'Courier-Oblique'
'bi'=>'Courier-BoldOblique'
'ib'=>'Courier-BoldOblique'
```

'Times-Roman'

```
'b'=>'Times-Bold'
'i'=>'Times-Italic'
'bi'=>'Times-BoldItalic'
'ib'=>'Times-BoldItalic'
```

(where 'b' indicate bold, and 'i' indicates italic)

Which means that (for example) if you have selected the 'Courier' font, and then you use the

Note that at the moment it is not possible to have any more fonts, so there is little use in

Note also that it is possible to have a different font when some text is bolded, then italicized,

If you bold a font twice, then under the current system it would look for a 'bb' entry in the font

```
$tmp = array(
    'b'=>'Courier-Bold'
    , 'i'=>'Courier-Oblique'
    , 'bi'=>'Courier-BoldOblique'
    , 'ib'=>'Courier-BoldOblique'
    , 'bb'=>'Times-Roman'
);
$pdf->setFontFamily('Courier',$tmp);
```

setEncryption ([userPass=""],[ownerPass=""],[pc=array], [mode=1])

Calling this function sets up the document to be encrypted, this is the only way to mark the

Since version 0.11.2 a new parameter "\$mode" has been added to this function to allow RC4

Using the call without options, defaults to preventing the user from cut & paste or printing.

```
$pdf->setEncryption();
```

Setting either off the passwords will mean that the user will have to enter a password to open

The pc array can be used to **allow** specific actions. The following example, sets an owner

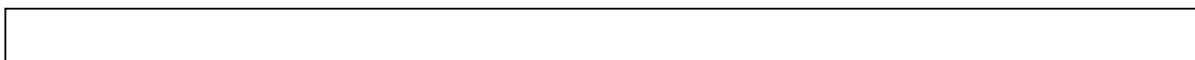
```
$pdf->setEncryption('trees','frogs',['copy','print']);
```

addLink (url,x0,y0,x1,y1)

Creates a clickable rectangular area within the document, which takes the user to the URL

See the information in the *inline codes* chapter of the ezPdf section to see an easy way of

```
$pdf->addLink("https://github.com/rospdf/pdf-php/",50,100,500,120);
$pdf->rectangle(50,100,450,20);
```



addInternalLink (label,x0,y0,x1,y1)

Creates an internal link in the document, 'label' is the name of a target point, and the other

Note that the destination markers are created using the *addDestination* function described

See the information in the *inline codes* chapter of the ezPdf section to see an easy way of

addDestination (label,style[,a][,b][,c])

This marks a point in the document as a potential destination for an internal link, the 'label'

transaction (action)

One of the major problems with this class has been the inability to format things so that they

Transaction support (terminology borrowed from databases) allows you to mark a point in the

'action' can be set to one of 'start','commit','rewind','abort', which are fairly obvious in their

'start' - mark a checkpoint, a position to return to upon 'abort'

'commit' - you are happy with this one, it releases the most recent 'start' though does not

'rewind' - you are not happy, but are planning to start again from the last checkpoint, this is

'abort' - you are not happy with you the document is looking, this will return you to the state at

Notes:

- 'start' commands can be nested, and 'abort' or 'commit' will always affect the most recent
- though often it seems that you do not need to 'commit' if you are happy with your changes,
- these commands, though they are part of the base class, save all the setting of the object in

Misc

Callback functions

Code has been included within class.pdf.php which allows the user to place a marker within

The marker is either of the forms:

```
<c:func:paramater>aaa bbb ccc</c:func>
<C:func:paramater>
```

The first form (with the small 'c') is for when the situation requires both an opening and a

When this piece of text is placed, the class function 'func' will be called, and it must have

```
x => x-position
y => y-position
status =>
f => function name
height => font height
decender => font decender
```

Where the status can have one of the values 'start','end','sol','eol'. The start and end values

Note that the function needs to be a function of the pdf class, so if you wish to use a custom

NOTE For various technical reasons if there are any spaces within the marker, then these

NOTE These functions should be used with care, it is possible to set up an infinite loop. If the

As an example, here is the code which extends the class for this document

```
include 'class.ezpdf.php';
```

```

class Creport extends Cezpdf {

    // make a location to store the information that will be collected during
    var $reportContents = [];

    // it is necessary to manually call the constructor of the extended class
    function Creport($p,$o){
        $this->Cezpdf($p,$o);
    }

    /*
    The function 'rf' records in an array the page number level number and
    <C:rf:1top%20heading>, which would be for a level 1 heading called 'top
    After the bulk of the document has been constructed, the array
    */
    function rf($info){
        $tmp = $info['p'];
        $lvl = $tmp[0];
        $lbl = rawurldecode(substr($tmp,1));
        $num=$this->ezWhatPageNumber($this->ezGetCurrentPageNumber());
        $this->reportContents[] = [$lbl,$num,$lvl ];
    }

    /*
    The dots function is called by a marker of the form:
    <C:dots:213>
    Which would represent a heading being show for which the original document
    */
    function dots($info){
        // draw a dotted line over to the right and put on a page number
        $tmp = $info['p'];
        $lvl = $tmp[0];
        $lbl = substr($tmp,1);
        $xpos = 520;

        switch($lvl){
            case '1':
                $size=16;
                $thick=1;
                break;
            case '2':
                $size=12;
                $thick=0.5;
                break;
        }
        $this->saveState();
        $this->setLineStyle($thick,'round','', [0,10]);
        $this->line($xpos,$info['y'],$info['x']+5,$info['y']);
        $this->restoreState();
        $this->addText($xpos+5,$info['y'],$size,$lbl);
    }
}

```

Both of these functions use only the single point call to a callback function, for an example of

Units

The units used for positioning within this class are the default pdf user units, in which each and some additional user contributed notes are (thanks Andrew):

1/72" is 0.3528mm or 1 point

1 point was historically 0.0138 inches, a little under 1/72"

10mm is 28.35 points

A4 is 210 x 297 mm or 595.28 x 841.89 points