PostScript document creation

Introduction

This module allows to create PostScript documents. It has many similarities with the pdf extension. Actually the API is almost identical and one can in many cases just replace the prefix of each function from pdf_ to ps_. This also works for functions which has no meaning in the PostScript document (like adding hyperlinks) but will have an effect if the document is converted to PDF.

Documents created by this extension are sometimes even superior to documents created with the pdf extension, because pslib's text rendering functions can handle kerning, hyphenation and ligatures which results in much better output of boxed text.

Installing/Configuring

Requirements

You need at least PHP 4.3.0 and pslib >= 0.1.12. The ps library (pslib) is available at » http://pslib.sourceforge.net/.

Installation

A short installation note: just type

```
$ pecl install ps
```

in your console.

Runtime Configuration

This extension has no configuration directives defined in *php.ini*.

Resource Types

This extension defines a PostScript document resource returned by ps_new().

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

The following two tables lists all constants defined by the ps extension.

Contants for line caps

Name	Meaning
ps_LINECAP_BUTT	
ps_LINECAP_ROUND	
ps_LINECAP_SQUARED	

Contants for line joins

Name	Meaning
ps_LINEJOIN_MITER	
ps_LINEJOIN_ROUND	
ps_LINEJOIN_BEVEL	

PS Functions

Contact Information

If you have comments, bugfixes, enhancements for either this extension or pslib then please drop me a mail » steinm@php.net. Any help is very welcome.

ps_add_bookmark

ps_add_bookmark -- Add bookmark to current page

Description

int **ps_add_bookmark** (resource `$psdoc`, string `$text` [, int `$parent` [, int `$open`]])

Adds a bookmark for the current page. Bookmarks usually appear in PDF-Viewers left of the page in a hierarchical tree. Clicking on a bookmark will jump to the given page.

The bookmark has no meaning if the document is printed or viewed, but it will be used if the document is converted to pdf by either Acrobat Distiller? or Ghostview.

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`text`
: The text used for displaying the bookmark.

`parent`
: A bookmark previously created by this function which is used as the parent of the new bookmark.

`open`
: If *`open`* is unequal to zero the bookmark will be shown open by the pdf viewer.

Return Values

The returned value is a reference for the bookmark. It is only used if the bookmark shall be used as a parent. The value is greater zero if the function succeeds. In case of an error zero will be returned.

See Also

- ps_add_launchlink()
- ps_add_pdflink()
- ps_add_weblink()

ps_add_launchlink

ps_add_launchlink -- Adds link which launches file

Description

bool **ps_add_launchlink** (resource `$psdoc`, float `$llx`, float `$lly`, float `$urx`, float `$ury`, string `$filename`)

Places a hyperlink at the given position pointing to a file program which is being started when clicked on. The hyperlink's source position is a rectangle with its lower left corner at (llx, lly) and its upper right corner at (urx, ury). The rectangle has by default a thin blue border.

The hyperlink will not be visible if the document is printed or viewed but it will show up if the document is converted to pdf by either Acrobat Distiller? or Ghostview.

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`llx`
: The x-coordinate of the lower left corner.

`lly`
: The y-coordinate of the lower left corner.

`urx`
: The x-coordinate of the upper right corner.

`ury`
: The y-coordinate of the upper right corner.

`filename`
: The path of the program to be started, when the link is clicked on.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_add_locallink()

- ps_add_pdflink()
- ps_add_weblink()

ps_add_locallink

ps_add_locallink -- Adds link to a page in the same document

Description

bool **ps_add_locallink** (resource `$psdoc`, float `$llx`, float `$lly`, float `$urx`, float `$ury`, int `$page`, string `$dest`)

Places a hyperlink at the given position pointing to a page in the same document. Clicking on the link will jump to the given page. The first page in a document has number 1.

The hyperlink's source position is a rectangle with its lower left corner at (*`llx`*, *`lly`*) and its upper right corner at (*`urx`*, *`ury`*). The rectangle has by default a thin blue border.

The hyperlink will not be visible if the document is printed or viewed but it will show up if the document is converted to pdf by either Acrobat Distiller? or Ghostview.

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`llx`
: The x-coordinate of the lower left corner.

`lly`
: The y-coordinate of the lower left corner.

`urx`
: The x-coordinate of the upper right corner.

`ury`
: The y-coordinate of the upper right corner.

`page`
: The number of the page displayed when clicking on the link.

`dest`
: The parameter *`dest`* determines how the document is being viewed. It can be *fitpage*, *fitwidth*, *fitheight*, or *fitbbox*.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_add_launchlink()
- ps_add_pdflink()
- ps_add_weblink()

ps_add_note

ps_add_note -- Adds note to current page

Description

bool **ps_add_note** (resource `$psdoc`, float `$llx`, float `$lly`, float `$urx`, float `$ury`, string `$contents`, string `$title`, string `$icon`, int `$open`)

Adds a note at a certain position on the page. Notes are like little rectangular sheets with text on it, which can be placed anywhere on a page. They are shown either folded or unfolded. If unfolded, the specified icon is used as a placeholder.

The note will not be visible if the document is printed or viewed but it will show up if the document is converted to pdf by either Acrobat Distiller? or Ghostview.

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`llx`
: The x-coordinate of the lower left corner.

`lly`
: The y-coordinate of the lower left corner.

`urx`
: The x-coordinate of the upper right corner.

`ury`
: The y-coordinate of the upper right corner.

`contents`
: The text of the note.

`title`
: The title of the note as displayed in the header of the note.

`icon`
: The icon shown if the note is folded. This parameter can be set to *comment*, *insert*, *note*, *paragraph*, *newparagraph*, *key*, or *help*.

`open`
: If *`open`* is unequal to zero the note will be shown unfolded after opening the document with a pdf viewer.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_add_pdflink()
- ps_add_launchlink()
- ps_add_locallink()
- ps_add_weblink()

ps_add_pdflink

ps_add_pdflink -- Adds link to a page in a second pdf document

Description

bool **ps_add_pdflink** (resource `$psdoc`, float `$llx`, float `$lly`, float `$urx`, float `$ury`, string `$filename`, int `$page`, string `$dest`)

Places a hyperlink at the given position pointing to a second pdf document. Clicking on the link will branch to the document at the given page. The first page in a document has number 1.

The hyperlink's source position is a rectangle with its lower left corner at (*`llx`*, *`lly`*) and its upper right corner at (*`urx`*, *`ury`*). The rectangle has by default a thin blue border.

The hyperlink will not be visible if the document is printed or viewed but it will show up if the document is converted to pdf by either Acrobat Distiller? or Ghostview.

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`llx`
: The x-coordinate of the lower left corner.

`lly`
: The y-coordinate of the lower left corner.

`urx`
: The x-coordinate of the upper right corner.

`ury`
: The y-coordinate of the upper right corner.

`filename`
: The name of the pdf document to be opened when clicking on this link.

`page`
: The page number of the destination pdf document

`dest`
: The parameter *`dest`* determines how the document is being viewed. It can be *fitpage*, *fitwidth*, *fitheight*, or *fitbbox*.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_add_launchlink()
- ps_add_locallink()
- ps_add_weblink()

ps_add_weblink

ps_add_weblink -- Adds link to a web location

Description

bool **ps_add_weblink** (resource `$psdoc`, float `$llx`, float `$lly`, float `$urx`, float `$ury`, string `$url`)

Places a hyperlink at the given position pointing to a web page. The hyperlink's source position is a rectangle with its lower left corner at (*`llx`*, *`lly`*) and its upper right corner at (*`urx`*, *`ury`*). The rectangle has by default a thin blue border.

The hyperlink will not be visible if the document is printed or viewed but it will show up if the document is converted to pdf by either Acrobat Distiller? or Ghostview.

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`llx`
: The x-coordinate of the lower left corner.

`lly`
: The y-coordinate of the lower left corner.

`urx`
: The x-coordinate of the upper right corner.

`ury`
: The y-coordinate of the upper right corner.

`url`
: The url of the hyperlink to be opened when clicking on this link, e.g. *http://www.php.net* .

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_add_launchlink()
- ps_add_locallink()

- ps_add_pdflink()

ps_arc

ps_arc -- Draws an arc counterclockwise

Description

bool **ps_arc** (resource `$psdoc`, float `$x`, float `$y`, float `$radius`, float `$alpha`, float `$beta`)

Draws a portion of a circle with at middle point at (*`x`*, *`y`*). The arc starts at an angle of *`alpha`* and ends at an angle of *`beta`*. It is drawn counterclockwise (use ps_arcn() to draw clockwise). The subpath added to the current path starts on the arc at angle *`alpha`* and ends on the arc at angle *`beta`*.

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`x`
: The x-coordinate of the circle's middle point.

`y`
: The y-coordinate of the circle's middle point.

`radius`
: The radius of the circle

`alpha`
: The start angle given in degrees.

`beta`
: The end angle given in degrees.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_arcn()

ps_arcn

ps_arcn -- Draws an arc clockwise

Description

bool **ps_arcn** (resource `$psdoc`, float `$x`, float `$y`, float `$radius`, float `$alpha`, float `$beta`)

Draws a portion of a circle with at middle point at (*`x`*, *`y`*). The arc starts at an angle of *`alpha`* and ends at an angle of *`beta`*. It is drawn clockwise (use ps_arc() to draw counterclockwise). The subpath added to the current path starts on the arc at angle *`beta`* and ends on the arc at angle *`alpha`*.

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`x`
: The x-coordinate of the circle's middle point.

`y`
: The y-coordinate of the circle's middle point.

`radius`
: The radius of the circle

`alpha`
: The starting angle given in degrees.

`beta`
: The end angle given in degrees.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_arc()

ps_begin_page

ps_begin_page -- Start a new page

Description

bool **ps_begin_page** (resource `$psdoc`, float `$width`, float `$height`)

Starts a new page. Although the parameters *`width`* and *`height`* imply a different page size for each page, this is not possible in PostScript. The first call of ps_begin_page() will set the page size for the whole document. Consecutive calls will have no effect, except for creating a new page. The situation is different if you intent to convert the PostScript document into PDF. This function places pdfmarks into the document which can set the size for each page indiviually. The resulting PDF document will have different page sizes.

Though PostScript does not know different page sizes, pslib places a bounding box for each page into the document. This size is evaluated by some PostScript viewers and will have precedence over the BoundingBox in the Header of the document. This can lead to unexpected results when you set a BoundingBox whose lower left corner is not (0, 0), because the bounding box of the page will always have a lower left corner (0, 0) and overwrites the global setting.

Each page is encapsulated into save/restore. This means, that most of the settings made on one page will not be retained on the next page.

If there is up to the first call of ps_begin_page() no call of ps_findfont(), then the header of the PostScript document will be output and the bounding box will be set to the size of the first page. The lower left corner of the bounding box is set to (0, 0). If ps_findfont() was called before, then the header has been output already, and the document will not have a valid bounding box. In order to prevent this, one should call ps_set_info() to set the info field *BoundingBox* and possibly *Orientation* before any ps_findfont() or ps_begin_page() calls.

Note
Up to version 0.2.6 of pslib, this function will always overwrite the BoundingBox and Orientation, if it has been set before with ps_set_info() and ps_findfont() has not been called before.

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`width`
: The width of the page in pixel, e.g. 596 for A4 format.

height
: The height of the page in pixel, e.g. 842 for A4 format.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_end_page()
- ps_findfont()
- ps_set_info()

ps_begin_pattern

ps_begin_pattern -- Start a new pattern

Description

int **ps_begin_pattern** (resource `$psdoc`, float `$width`, float `$height`, float `$xstep`, float `$ystep`, int `$painttype`)

Starts a new pattern. A pattern is like a page containing e.g. a drawing which can be used for filling areas. It is used like a color by calling ps_setcolor() and setting the color space to *pattern*.

Parameters

psdoc
: Resource identifier of the postscript file as returned by ps_new().

width
: The width of the pattern in pixel.

height
: The height of the pattern in pixel.

x-step
: The distance in pixel of placements of the pattern in horizontal direction.

y-step
: The distance in pixel of placements of the pattern in vertical direction.

painttype
: Must be 1 or 2.

Return Values

The identifier of the pattern or **FALSE** in case of an error.

Examples

Example #1 - Creating and using a pattern

```
<?php
$ps = ps_new();

if (!ps_open_file($ps, "pattern.ps")) {
```

```
 print "Cannot open PostScript file\n";
 exit;
}

ps_set_parameter($ps, "warning", "true");
ps_set_info($ps, "Creator", "pattern.php");
ps_set_info($ps, "Author", "Uwe Steinmann");
ps_set_info($ps, "Title", "Pattern example");


$pspattern = ps_begin_pattern($ps, 10.0, 10.0, 10.0, 10.0, 1);
ps_setlinewidth($ps, 0.2);
ps_setcolor($ps, "stroke", "rgb", 0.0, 0.0, 1.0, 0.0);
ps_moveto($ps, 0, 0);
ps_lineto($ps, 7, 7);
ps_stroke($ps);
ps_moveto($ps, 0, 7);
ps_lineto($ps, 7, 0);
ps_stroke($ps);
ps_end_pattern($ps);

ps_begin_page($ps, 596, 842);
ps_setcolor($ps, "both", "pattern", $pspattern, 0.0, 0.0, 0.0);
ps_rect($ps, 50, 400, 200, 200);
ps_fill($ps);
ps_end_page($ps);

ps_close($ps);
ps_delete($ps);
?>
```

See Also

- ps_end_pattern()
- ps_setcolor()
- ps_shading_pattern()

ps_begin_template

ps_begin_template -- Start a new template

Description

int **ps_begin_template** (resource `$psdoc`, float `$width`, float `$height`)

Starts a new template. A template is called a form in the postscript language. It is created similar to a pattern but used like an image. Templates are often used for drawings which are placed several times through out the document, e.g. like a company logo. All drawing functions may be used within a template. The template will not be drawn until it is placed by ps_place_image().

Parameters

psdoc
: Resource identifier of the postscript file as returned by ps_new().

width
: The width of the template in pixel.

height
: The height of the template in pixel.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #2 - Creating and using a template

```
<?php
$ps = ps_new();

if (!ps_open_file($ps, "template.ps")) {
 print "Cannot open PostScript file\n";
 exit;
}

ps_set_parameter($ps, "warning", "true");
ps_set_info($ps, "Creator", "template.php");
ps_set_info($ps, "Author", "Uwe Steinmann");
ps_set_info($ps, "Title", "Template example");
```

```
$pstemplate = ps_begin_template($ps, 30.0, 30.0);
ps_moveto($ps, 0, 0);
ps_lineto($ps, 30, 30);
ps_moveto($ps, 0, 30);
ps_lineto($ps, 30, 0);
ps_stroke($ps);
ps_end_template($ps);

ps_begin_page($ps, 596, 842);
ps_place_image($ps, $pstemplate, 20.0, 20.0, 1.0);
ps_place_image($ps, $pstemplate, 50.0, 30.0, 0.5);
ps_place_image($ps, $pstemplate, 70.0, 70.0, 0.6);
ps_place_image($ps, $pstemplate, 30.0, 50.0, 1.3);
ps_end_page($ps);

ps_close($ps);
ps_delete($ps);
?>
```

See Also

- ps_end_template()

ps_circle

ps_circle -- Draws a circle

Description

bool **ps_circle** (resource `$psdoc`, float `$x`, float `$y`, float `$radius`)

Draws a circle with its middle point at (*`x`*, *`y`*). The circle starts and ends at position (*`x`* + *`radius`*, *`y`*). If this function is called outside a path it will start a new path. If it is called within a path it will add the circle as a subpath. If the last drawing operation does not end in point (*`x`* + *`radius`*, *`y`*) then there will be a gap in the path.

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`x`
: The x-coordinate of the circle's middle point.

`y`
: The y-coordinate of the circle's middle point.

`radius`
: The radius of the circle

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_arc()
- ps_arcn()

ps_clip

ps_clip -- Clips drawing to current path

Description

bool **ps_clip** (resource `$psdoc`)

Takes the current path and uses it to define the border of a clipping area. Everything drawn outside of that area will not be visible.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_closepath()

ps_close_image

ps_close_image -- Closes image and frees memory

Description

void **ps_close_image** (resource `$psdoc`, int `$imageid`)

Closes an image and frees its resources. Once an image is closed it cannot be used anymore.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`imageid`
Resource identifier of the image as returned by ps_open_image() or ps_open_image_file().

Return Values

Returns **NULL** on success or **FALSE** on failure.

See Also

- ps_open_image()
- ps_open_image_file()

ps_close

ps_close -- Closes a PostScript document

Description

bool **ps_close** (resource `$psdoc`)

Closes the PostScript document.

This function writes the trailer of the PostScript document. It also writes the bookmark tree. ps_close() does not free any resources, which is done by ps_delete().

This function is also called by ps_delete() if it has not been called before.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_open_file()
- ps_delete()

ps_closepath_stroke

ps_closepath_stroke -- Closes and strokes path

Description

bool **ps_closepath_stroke** (resource `$psdoc`)

Connects the last point with first point of a path and draws the resulting closed line.

Parameters

psdoc
: Resource identifier of the postscript file as returned by ps_new().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_closepath()

ps_closepath

ps_closepath -- Closes path

Description

bool **ps_closepath** (resource `$psdoc`)

Connects the last point with the first point of a path. The resulting path can be used for stroking, filling, clipping, etc..

Parameters

psdoc
: Resource identifier of the postscript file as returned by ps_new().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_clip()
- ps_closepath_stroke()

ps_continue_text

ps_continue_text -- Continue text in next line

Description

bool **ps_continue_text** (resource `$psdoc`, string `$text`)

Output a text one line below the last line. The line spacing is taken from the value "leading" which must be set with ps_set_value(). The actual position of the text is determined by the values "textx" and "texty" which can be requested with ps_get_value()

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`text`
The text to output.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_show()
- ps_show_xy()
- ps_show_boxed()

ps_curveto

ps_curveto -- Draws a curve

Description

bool **ps_curveto** (resource `$psdoc`, float `$x1`, float `$y1`, float `$x2`, float `$y2`, float `$x3`, float `$y3`)

Add a section of a cubic Bézier curve described by the three given control points to the current path.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`x1`
x-coordinate of first control point.

`y1`
y-coordinate of first control point.

`x2`
x-coordinate of second control point.

`y2`
y-coordinate of second control point.

`x3`
x-coordinate of third control point.

`y3`
y-coordinate of third control point.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_lineto()

ps_delete

ps_delete -- Deletes all resources of a PostScript document

Description

bool **ps_delete** (resource `$psdoc`)

Mainly frees memory used by the document. Also closes a file, if it was not closed before with ps_close(). You should in any case close the file with ps_close() before, because ps_close() not just closes the file but also outputs a trailor containing PostScript comments like the number of pages in the document and adding the bookmark hierarchy.

Parameters

psdoc
: Resource identifier of the postscript file as returned by ps_new().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_close()

ps_end_page

ps_end_page -- End a page

Description

bool **ps_end_page** (resource `$psdoc`)

Ends a page which was started with ps_begin_page(). Ending a page will leave the current drawing context, which e.g. requires to reload fonts if they were loading within the page, and to set many other drawing parameters like the line width, or color..

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_begin_page()

ps_end_pattern

ps_end_pattern -- End a pattern

Description

bool **ps_end_pattern** (resource `$psdoc`)

Ends a pattern which was started with ps_begin_pattern(). Once a pattern has been ended, it can be used like a color to fill areas.

Parameters

psdoc
: Resource identifier of the postscript file as returned by ps_new().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_begin_pattern()

ps_end_template

ps_end_template -- End a template

Description

bool **ps_end_template** (resource `$psdoc`)

Ends a template which was started with ps_begin_template(). Once a template has been ended, it can be used like an image.

Parameters

psdoc
: Resource identifier of the postscript file as returned by ps_new().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_begin_template()

ps_fill_stroke

ps_fill_stroke -- Fills and strokes the current path

Description

bool **ps_fill_stroke** (resource `$psdoc`)

Fills and draws the path constructed with previously called drawing functions like ps_lineto().

Parameters

psdoc
: Resource identifier of the postscript file as returned by ps_new().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_fill()
- ps_stroke()

ps_fill

ps_fill -- Fills the current path

Description

bool **ps_fill** (resource `$psdoc`)

Fills the path constructed with previously called drawing functions like ps_lineto().

Parameters

psdoc
: Resource identifier of the postscript file as returned by ps_new().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_fill_stroke()
- ps_stroke()

ps_findfont

ps_findfont -- Loads a font

Description

int **ps_findfont** (resource `$psdoc`, string `$fontname`, string `$encoding` [, bool `$embed`])

Loads a font for later use. Before text is output with a loaded font it must be set with ps_setfont(). This function needs the adobe font metric file in order to calculate the space used up by the characters. A font which is loaded within a page will only be available on that page. Fonts which are to be used in the complete document have to be loaded before the first call of ps_begin_page(). Calling ps_findfont() between pages will make that font available for all following pages.

The name of the afm file must be *`fontname`.afm*. If the font shall be embedded the file *`fontname`.pfb* containing the font outline must be present as well.

Calling ps_findfont() before the first page requires to output the postscript header which includes the BoundingBox for the whole document. Usually the BoundingBox is set with the first call of ps_begin_page() which now comes after ps_findfont(). Consequently the BoundingBox has not been set and a warning will be issued when ps_findfont() is called. In order to prevent this situation, one should call ps_set_parameter() to set the BoundingBox before ps_findfont() is called.

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`fontname`
: The name of the font.

`encoding`
: ps_findfont() will try to load the file passed in the parameter *`encoding`*. Encoding files are of the same syntax as those used by *dvips(1)*. They contain a font encoding vector (which is currently not used but must be present) and a list of extra ligatures to extend the list of ligatures derived from the afm file. *`encoding`* can be **NULL** or the empty string if the default encoding (TeXBase1) shall be used. If the encoding is set to *builtin* then there will be no reencoding and the font specific encoding will be used. This is very useful with symbol fonts.

`embed`
: If set to a value >0 the font will be embedded into the document. This requires the font outline (.pfb file) to be present.

Return Values

Returns the identifier of the font or zero in case of an error. The identifier is a positive number.

See Also

- ps_begin_page()
- ps_setfont()

ps_get_buffer

ps_get_buffer -- Fetches the full buffer containig the generated PS data

Description

string **ps_get_buffer** (resource `$psdoc`)

This function is not implemented yet. It will always return an empty string. The idea for a later implementation is to write the contents of the postscript file into an internal buffer if in memory creation is requested, and retrieve the buffer content with this function. Currently, documents created in memory are send to the browser without buffering.

Warning
This function is currently not documented; only its argument list is available.

Parameters

psdoc
: Resource identifier of the postscript file as returned by ps_new().

See Also

- ps_open_file()

ps_get_parameter

ps_get_parameter -- Gets certain parameters

Description

string **ps_get_parameter** (resource `$psdoc`, string `$name` [, float `$modifier`])

Gets several parameters which were directly set by ps_set_parameter() or indirectly by one of the other functions. Parameters are by definition string values. This function cannot be used to retrieve resources which were also set by ps_set_parameter().

The parameter *`name`* can have the following values.

fontname
: The name of the currently active font or the font whose identifier is passed in parameter *`modifier`*.

fontencoding
: The encoding of the currently active font.

dottedversion
: The version of the underlying pslib library in the format <major>.<minor>.<subminor>

scope
: The current drawing scope. Can be object, document, null, page, pattern, path, template, prolog, font, glyph.

ligaturedisolvechar
: The character which dissolves a ligature. If your are using a font which contains the ligature `ff' and `|' is the char to dissolve the ligature, then `f|f' will result in two `f' instead of the ligature `ff'.

imageencoding
: The encoding used for encoding images. Can be either *hex* or *85*. hex encoding uses two bytes in the postscript file each byte in the image. 85 stand for Ascii85 encoding.

linenumbermode
: Set to *paragraph* if lines are numbered within a paragraph or *box* if they are numbered within the surrounding box.

linebreak
: Only used if text is output with ps_show_boxed(). If set to *true* a carriage return will add a line break.

parbreak
: Only used if text is output with ps_show_boxed(). If set to *true* a carriage return will start a new paragraph.

hyphenation
: Only used if text is output with ps_show_boxed(). If set to *true* the paragraph will be hyphenated if a hypen dictionary is set and exists.

hyphendict
: Filename of the dictionary used for hyphenation pattern.

Parameters

psdoc
: Resource identifier of the postscript file as returned by ps_new().

name
: Name of the parameter.

modifier
: An identifier needed if a parameter of a resource is requested, e.g. the size of an image. In such a case the resource id is passed.

Return Values

Returns the value of the parameter or **FALSE** in case of an error.

See Also

- ps_set_parameter()

ps_get_value

ps_get_value -- Gets certain values

Description

float **ps_get_value** (resource `$psdoc`, string `$name` [, float `$modifier`])

Gets several values which were set by ps_set_value(). Values are by definition float values.

The parameter *`name`* can have the following values.

fontsize
: The size of the currently active font or the font whose identifier is passed in parameter *`modifier`*.

font
: The currently active font itself.

imagewidth
: The width of the image whose id is passed in the parameter *`modifier`*.

imageheight
: The height of the image whose id is passed in the parameter *`modifier`*.

capheight
: The height of a capital M in the currently active font or the font whose identifier is passed in parameter *`modifier`*.

ascender
: The ascender of the currently active font or the font whose identifier is passed in parameter *`modifier`*.

descender
: The descender of the currently active font or the font whose identifier is passed in parameter *`modifier`*.

italicangle
: The italicangle of the currently active font or the font whose identifier is passed in parameter *`modifier`*.

underlineposition
: The underlineposition of the currently active font or the font whose identifier is passed in parameter *`modifier`*.

underlinethickness
: The underlinethickness of the currently active font or the font whose identifier is

passed in parameter `modifier`.

textx
The current x-position for text output.

texty
The current y-position for text output.

textrendering
The current mode for text rendering.

textrise
The space by which text is risen above the base line.

leading
The distance between text lines in points.

wordspacing
The space between words as a multiple of the width of a space char.

charspacing
The space between chars. If charspacing is != 0.0 ligatures will always be dissolved.

hyphenminchars
Minimum number of chars hyphenated at the end of a word.

parindent
Indention of the first n line in a paragraph.

numindentlines
Number of line in a paragraph to indent if parindent != 0.0.

parskip
Distance between paragraphs.

linenumberspace
Overall space in front of each line for the line number.

linenumbersep
Space between the line and the line number.

major
The major version number of pslib.

minor
The minor version number of pslib.

subminor, *revision*
The subminor version number of pslib.

Parameters

psdoc
: Resource identifier of the postscript file as returned by ps_new().

name
: Name of the value.

modifier
: The parameter *modifier* specifies the resource for which the value is to be retrieved. This can be the id of a font or an image.

Return Values

Returns the value of the parameter or **FALSE**.

See Also

- ps_set_value()

ps_hyphenate

ps_hyphenate -- Hyphenates a word

Description

array **ps_hyphenate** (resource `$psdoc`, string `$text`)

Hyphenates the passed word. ps_hyphenate() evaluates the value hyphenminchars (set by ps_set_value()) and the parameter hyphendict (set by ps_set_parameter()). hyphendict must be set before calling this function.

This function requires the locale category LC_CTYPE to be set properly. This is done when the extension is initialized by using the environment variables. On Unix systems read the man page of locale for more information.

Parameters

psdoc
: Resource identifier of the postscript file as returned by ps_new().

text
: *text* should not contain any non alpha characters. Possible positions for breaks are returned in an array of interger numbers. Each number is the position of the char in *text* after which a hyphenation can take place.

Return Values

An array of integers indicating the position of possible breaks in the text or **FALSE** in case of an error.

Examples

Example #3 - Hyphennate a text

```
<?php
$word = "Koordinatensystem";
$psdoc = ps_new();
ps_set_parameter($psdoc, "hyphendict", "hyph_de.dic");
$hyphens = ps_hyphenate($psdoc, $word);
for($i=0; $i<strlen($word); $i++) {
 echo $word[$i];
 if(in_array($i, $hyphens))
   echo "-";
}
ps_delete($psdoc);
```

```
?>
```

The above example will output:

```
Ko-ordi-na-ten-sys-tem
```

See Also

- ps_show_boxed()
- locale(1)

ps_include_file

ps_include_file -- Reads an external file with raw PostScript code

Description

bool **ps_include_file** (resource `$psdoc`, string `$file`)

Warning
This function is currently not documented; only its argument list is available.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`file`

Return Values

Returns **TRUE** on success or **FALSE** on failure.

ps_lineto

ps_lineto -- Draws a line

Description

bool **ps_lineto** (resource `$psdoc`, float `$x`, float `$y`)

Adds a straight line from the current point to the given coordinates to the current path. Use ps_moveto() to set the starting point of the line.

Parameters

psdoc
Resource identifier of the postscript file as returned by ps_new().

x
x-coordinate of the end point of the line.

y
y-coordinate of the end point of the line.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #4 - Drawing a rectangle

```
<?php
$ps = ps_new();
if (!ps_open_file($ps, "rectangle.ps")) {
 print "Cannot open PostScript file\n";
 exit;
}

ps_set_info($ps, "Creator", "rectangle.php");
ps_set_info($ps, "Author", "Uwe Steinmann");
ps_set_info($ps, "Title", "Lineto example");

ps_begin_page($ps, 596, 842);
ps_moveto($ps, 100, 100);
ps_lineto($ps, 100, 200);
ps_lineto($ps, 200, 200);
ps_lineto($ps, 200, 100);
ps_lineto($ps, 100, 100);
```

```
ps_stroke($ps);
ps_end_page($ps);

ps_delete($ps);
?>
```

See Also

- ps_moveto()

ps_makespotcolor

ps_makespotcolor -- Create spot color

Description

int **ps_makespotcolor** (resource `$psdoc`, string `$name` [, float `$reserved`])

Creates a spot color from the current fill color. The fill color must be defined in rgb, cmyk or gray colorspace. The spot color name can be an arbitrary name. A spot color can be set as any color with ps_setcolor(). When the document is not printed but displayed by an postscript viewer the given color in the specified color space is use.

Parameters

psdoc
: Resource identifier of the postscript file as returned by ps_new().

name
: Name of the spot color, e.g. Pantone 5565.

Return Values

The id of the new spot color or 0 in case of an error.

Examples

Example #5 - Creating and using a spot color

```
<?php
$ps = ps_new();
if (!ps_open_file($ps, "spotcolor.ps")) {
 print "Cannot open PostScript file\n";
 exit;
}

ps_set_info($ps, "Creator", "spotcolor.php");
ps_set_info($ps, "Author", "Uwe Steinmann");
ps_set_info($ps, "Title", "Spot color example");

ps_begin_page($ps, 596, 842);
ps_setcolor($ps, "fill", "cmyk", 0.37, 0.0, 0.34, 0.34);
$spotcolor = ps_makespotcolor($ps, "PANTONE 5565 C", 0);
ps_setcolor($ps, "fill", "spot", $spotcolor, 0.5, 0.0, 0.0);
ps_moveto($ps, 100, 100);
ps_lineto($ps, 100, 200);
ps_lineto($ps, 200, 200);
```

```
ps_lineto($ps, 200, 100);
ps_lineto($ps, 100, 100);
ps_fill($ps);
ps_end_page($ps);

ps_delete($ps);
?>
```

This example creates the spot color "PANTONE 5565 C" which is a darker green (olive) and fills a rectangle with 50% intensity.

See Also

- ps_setcolor()

ps_moveto

ps_moveto -- Sets current point

Description

bool **ps_moveto** (resource `$psdoc`, float `$x`, float `$y`)

Sets the current point to new coordinates. If this is the first call of ps_moveto() after a previous path has been ended then it will start a new path. If this function is called in the middle of a path it will just set the current point and start a subpath.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

x
x-coordinate of the point to move to.

y
y-coordinate of the point to move to.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_lineto()

ps_new

ps_new -- Creates a new PostScript document object

Description

resource **ps_new** (void)

Creates a new document instance. It does not create the file on disk or in memory, it just sets up everything. ps_new() is usually followed by a call of ps_open_file() to actually create the postscript document.

Return Values

Resource of PostScript document or **FALSE** on failure. The return value is passed to all other functions as the first argument.

See Also

- ps_delete()

ps_open_file

ps_open_file -- Opens a file for output

Description

bool **ps_open_file** (resource `$psdoc` [, string `$filename`])

Creates a new file on disk and writes the PostScript document into it. The file will be closed when ps_close() is called.

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`filename`
: The name of the postscript file. If *`filename`* is not passed the document will be created in memory and all output will go straight to the browser.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_close()

ps_open_image_file

ps_open_image_file -- Opens image from file

Description

int **ps_open_image_file** (resource `$psdoc`, string `$type`, string `$filename` [, string `$stringparam` [, int `$intparam`]])

Loads an image for later use.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`type`
The type of the image. Possible values are *png*, *jpeg*, or *eps*.

`filename`
The name of the file containing the image data.

`stringparam`
Not used.

`intparam`
Not used.

Return Values

Returns identifier of image or zero in case of an error. The identifier is a positive number greater than 0.

See Also

- ps_open_image()
- ps_place_image()
- ps_close_image()

ps_open_image

ps_open_image -- Reads an image for later placement

Description

int **ps_open_image** (resource `$psdoc`, string `$type`, string `$source`, string `$data`, int `$lenght`, int `$width`, int `$height`, int `$components`, int `$bpc`, string `$params`)

Reads an image which is already available in memory. The parameter *`source`* is currently not evaluated and assumed to be *memory*. The image data is a sequence of pixels starting in th upper left corner and ending in the lower right corner. Each pixel consists of *`components`* color components, and each component has *`bpc`* bits.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`type`
The type of the image. Possible values are *png*, *jpeg*, or *eps*.

`source`
Not used.

`data`
The image data.

`length`
The length of the image data.

`width`
The width of the image.

`height`
The height of the image.

`components`
The number of components for each pixel. This can be 1 (gray scale images), 3 (rgb images), or 4 (cmyk, rgba images).

`bpc`
Number of bits per component (quite often 8).

`params`

Return Values

Returns identifier of image or zero in case of an error. The identifier is a positive number greater than 0.

See Also

- ps_open_image_file()
- ps_place_image()
- ps_close_image()

ps_open_memory_image

ps_open_memory_image -- Takes an GD image and returns an image for placement in a PS document

Description

int **ps_open_memory_image** (resource `$psdoc`, int `$gd`)

Warning
This function is currently not documented; only its argument list is available.

Parameters

psdoc
: Resource identifier of the postscript file as returned by ps_new().

gd

ps_place_image

ps_place_image -- Places image on the page

Description

bool **ps_place_image** (resource `$psdoc`, int `$imageid`, float `$x`, float `$y`, float `$scale`)

Places a formerly loaded image on the page. The image can be scaled. If the image shall be rotated as well, you will have to rotate the coordinate system before with ps_rotate().

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`imageid`
The resource identifier of the image as returned by ps_open_image() or ps_open_image_file().

`x`
x-coordinate of the lower left corner of the image.

`y`
y-coordinate of the lower left corner of the image.

`scale`
The scaling factor for the image. A scale of 1.0 will result in a resolution of 72 dpi, because each pixel is equivalent to 1 point.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_open_image()
- ps_open_image_file()

ps_rect

ps_rect -- Draws a rectangle

Description

bool **ps_rect** (resource `$psdoc`, float `$x`, float `$y`, float `$width`, float `$height`)

Draws a rectangle with its lower left corner at (*x*, *y*). The rectangle starts and ends in its lower left corner. If this function is called outside a path it will start a new path. If it is called within a path it will add the rectangle as a subpath. If the last drawing operation does not end in the lower left corner then there will be a gap in the path.

Parameters

psdoc
: Resource identifier of the postscript file as returned by ps_new().

x
: x-coordinate of the lower left corner of the rectangle.

y
: y-coordinate of the lower left corner of the rectangle.

width
: The width of the image.

height
: The height of the image.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_arc()
- **ps_cirle()**
- ps_lineto()

ps_restore

ps_restore -- Restore previously save context

Description

bool **ps_restore** (resource `$psdoc`)

Restores a previously saved graphics context. Any call of ps_save() must be accompanied by a call to ps_restore(). All coordinate transformations, line style settings, color settings, etc. are being restored to the state before the call of ps_save().

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_save()

ps_rotate

ps_rotate -- Sets rotation factor

Description

bool **ps_rotate** (resource `$psdoc`, float `$rot`)

Sets the rotation of the coordinate system.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`rot`
Angle of rotation in degree.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #6 - Rotation of the coordinate system

```
<?php
function rectangle($ps) {
   ps_moveto($ps, 0, 0);
   ps_lineto($ps, 0, 50);
   ps_lineto($ps, 50, 50);
   ps_lineto($ps, 50, 0);
   ps_lineto($ps, 0, 0);
   ps_stroke($ps);
}

$ps = ps_new();
if (!ps_open_file($ps, "rotation.ps")) {
 print "Cannot open PostScript file\n";
 exit;
}

ps_set_info($ps, "Creator", "rotation.php");
ps_set_info($ps, "Author", "Uwe Steinmann");
ps_set_info($ps, "Title", "Rotation example");
ps_set_info($ps, "BoundingBox", "0 0 596 842");

$psfont = ps_findfont($ps, "Helvetica", "", 0);
```

```
ps_begin_page($ps, 596, 842);
ps_set_text_pos($ps, 100, 100);
ps_save($ps);
ps_translate($ps, 100, 100);
ps_rotate($ps, 45);
rectangle($ps);
ps_restore($ps);
ps_setfont($ps, $psfont, 8.0);
ps_show($ps, "Text without rotation");
ps_end_page($ps);

ps_delete($ps);
?>
```

The above example illustrates a very common way of rotating a graphic (in this case just a rectangle) by simply rotating the coordinate system. Since the graphic's coordinate system assumes (0,0) to be the origin, the page coordinate system is also translated to place the graphics not on the edge of the page. Pay attention to the order of ps_translate() and ps_rotate(). In the above case the rectancle is rotated around the point (100, 100) in the untranslated coordinate system. Switching the two statements has a completely different result.

In order to output the following text at the original position, all modifications of the coordinate system are encapsulated in ps_save() and ps_restore().

See Also

- ps_scale()
- ps_translate()

ps_save

ps_save -- Save current context

Description

bool **ps_save** (resource `$psdoc`)

Saves the current graphics context, containing colors, translation and rotation settings and some more. A saved context can be restored with ps_restore().

Parameters

psdoc
: Resource identifier of the postscript file as returned by ps_new().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_restore()

ps_scale

ps_scale -- Sets scaling factor

Description

bool **ps_scale** (resource `$psdoc`, float `$x`, float `$y`)

Sets horizontal and vertical scaling of the coordinate system.

Parameters

psdoc
: Resource identifier of the postscript file as returned by ps_new().

x
: Scaling factor in horizontal direction.

y
: Scaling factor in vertical direction.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_rotate()
- ps_translate()

ps_set_border_color

ps_set_border_color -- Sets color of border for annotations

Description

bool **ps_set_border_color** (resource `$psdoc`, float `$red`, float `$green`, float `$blue`)

Links added with one of the functions ps_add_weblink(), ps_add_pdflink(), etc. will be displayed with a surounded rectangle when the postscript document is converted to pdf and viewed in a pdf viewer. This rectangle is not visible in the postscript document. This function sets the color of the rectangle's border line.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`red`
The red component of the border color.

`green`
The green component of the border color.

`blue`
The blue component of the border color.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_set_border_dash()
- ps_set_border_style()

ps_set_border_dash

ps_set_border_dash -- Sets length of dashes for border of annotations

Description

bool **ps_set_border_dash** (resource `$psdoc`, float `$black`, float `$white`)

Links added with one of the functions ps_add_weblink(), ps_add_pdflink(), etc. will be displayed with a surounded rectangle when the postscript document is converted to pdf and viewed in a pdf viewer. This rectangle is not visible in the postscript document. This function sets the length of the black and white portion of a dashed border line.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`black`
The length of the dash.

`white`
The length of the gap between dashes.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_set_border_color()
- ps_set_border_style()

ps_set_border_style

ps_set_border_style -- Sets border style of annotations

Description

bool **ps_set_border_style** (resource `$psdoc`, string `$style`, float `$width`)

Links added with one of the functions ps_add_weblink(), ps_add_pdflink(), etc. will be displayed with a surounded rectangle when the postscript document is converted to pdf and viewed in a pdf viewer. This rectangle is not visible in the postscript document. This function sets the appearance and width of the border line.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`style`
`style` can be *solid* or *dashed*.

`width`
The line width of the border.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_set_border_color()
- ps_set_border_dash()

ps_set_info

ps_set_info -- Sets information fields of document

Description

bool **ps_set_info** (resource `$p`, string `$key`, string `$val`)

Sets certain information fields of the document. This fields will be shown as a comment in the header of the PostScript file. If the document is converted to pdf this fields will also be used for the document information.

The *BoundingBox* is usually set to the value given to the first page. This only works if ps_findfont() has not been called before. In such cases the BoundingBox would be left unset unless you set it explicitly with this function.

This function will have no effect anymore when the header of the postscript file has been already written. It must be called before the first page or the first call of ps_findfont().

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`key`
The name of the information field to set. The values which can be set are *Keywords*, *Subject*, *Title*, *Creator*, *Author*, *BoundingBox*, and *Orientation*. Be aware that some of them has a meaning to PostScript viewers.

`value`
The value of the information field. The field *Orientation* can be set to either *Portrait* or *Landscape*. The *BoundingBox* is a string consisting of four numbers. The first two numbers are the coordinates of the lower left corner of the page. The last two numbers are the coordinates of the upper right corner.

Note
Up to version 0.2.6 of pslib, the BoundingBox and Orientation will be overwritten by ps_begin_page(), unless ps_findfont() has been called before.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_findfont()
- ps_begin_page()

ps_set_parameter

ps_set_parameter -- Sets certain parameters

Description

bool **ps_set_parameter** (resource `$psdoc`, string `$name`, string `$value`)

Sets several parameters which are used by many functions. Parameters are by definition string values.

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`name`
: For a list of possible names see ps_get_parameter().

`value`
: The value of the parameter.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- **ps_get_parameters()**
- ps_set_value()

ps_set_text_pos

ps_set_text_pos -- Sets position for text output

Description

bool **ps_set_text_pos** (resource `$psdoc`, float `$x`, float `$y`)

Set the position for the next text output. You may alternatively set the x and y value separately by calling ps_set_value() and choosing *textx* respectively *texty* as the value name.

If you want to output text at a certain position it is more convenient to use ps_show_xy() instead of setting the text position and calling ps_show().

Parameters

psdoc
Resource identifier of the postscript file as returned by ps_new().

x
x-coordinate of the new text position.

y
y-coordinate of the new text position.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #7 - Placing text at a given position

```
<?php
$ps = ps_new();
if (!ps_open_file($ps, "text.ps")) {
 print "Cannot open PostScript file\n";
 exit;
}

ps_set_info($ps, "Creator", "rectangle.php");
ps_set_info($ps, "Author", "Uwe Steinmann");
ps_set_info($ps, "Title", "Text placement example");

ps_begin_page($ps, 596, 842);
```

```
$psfont = ps_findfont($ps, "Helvetica", "", 0);
ps_setfont($ps, $psfont, 8.0);
ps_show_xy($ps, "Some text at (100, 100)", 100, 100);

ps_set_value($ps, "textx", 100);
ps_set_value($ps, "texty", 120);
ps_show($ps, "Some text at (100, 120)");
ps_end_page($ps);

ps_delete($ps);
?>
```

See Also

- ps_set_value()
- ps_show()

ps_set_value

ps_set_value -- Sets certain values

Description

bool **ps_set_value** (resource `$psdoc`, string `$name`, float `$value`)

Sets several values which are used by many functions. Parameters are by definition float values.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`name`
The *`name`* can be one of the following:
textrendering
The way how text is shown.

textx
The x coordinate for text output.

texty
The y coordinate for text output.

wordspacing
The distance between words relative to the width of a space.

leading
The distance between lines in pixels.

`value`
The value of the parameter.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_get_value()

- ps_set_parameter()

ps_setcolor

ps_setcolor -- Sets current color

Description

bool **ps_setcolor** (resource `$psdoc`, string `$type`, string `$colorspace`, float `$c1`, float `$c2`, float `$c3`, float `$c4`)

Sets the color for drawing, filling, or both.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`type`
The parameter *`type`* can be *both*, *fill*, or *fillstroke*.

`colorspace`
The colorspace should be one of *gray*, *rgb*, *cmyk*, *spot*, *pattern*. Depending on the colorspace either only the first, the first three or all parameters will be used.

`c1`
Depending on the colorspace this is either the red component (rgb), the cyan component (cmyk), the gray value (gray), the identifier of the spot color or the identifier of the pattern.

`c2`
Depending on the colorspace this is either the green component (rgb), the magenta component (cmyk).

`c3`
Depending on the colorspace this is either the blue component (rgb), the yellow component (cmyk).

`c4`
This must only be set in cmyk colorspace and specifies the black component.

Bugs

The second parameter is currently not always evaluated. The color is sometimes set for filling and drawing just as if *fillstroke* were passed.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

ps_setdash

ps_setdash -- Sets appearance of a dashed line

Description

bool **ps_setdash** (resource `$psdoc`, float `$on`, float `$off`)

Sets the length of the black and white portions of a dashed line.

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`on`
: The length of the dash.

`off`
: The length of the gap between dashes.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_setpolydash()

ps_setflat

ps_setflat -- Sets flatness

Description

bool **ps_setflat** (resource `$psdoc`, float `$value`)

Warning
This function is currently not documented; only its argument list is available.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`value`
The *`value`* must be between 0.2 and 1.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

ps_setfont

ps_setfont -- Sets font to use for following output

Description

bool **ps_setfont** (resource `$psdoc`, int `$fontid`, float `$size`)

Sets a font, which has to be loaded before with ps_findfont(). Outputting text without setting a font results in an error.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`fontid`
The font identifier as returned by ps_findfont().

`size`
The size of the font.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_findfont()
- ps_set_text_pos() for an example.

ps_setgray

ps_setgray -- Sets gray value

Description

bool **ps_setgray** (resource `$psdoc`, float `$gray`)

Sets the gray value for all following drawing operations.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`gray`
The value must be between 0 (white) and 1 (black).

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_setcolor()

ps_setlinecap

ps_setlinecap -- Sets appearance of line ends

Description

bool **ps_setlinecap** (resource `$psdoc`, int `$type`)

Sets how line ends look like.

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`type`
: The type of line ends. Possible values are *PS_LINECAP_BUTT*, *PS_LINECAP_ROUND*, or *PS_LINECAP_SQUARED*.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_setlinejoin()
- ps_setlinewidth()
- ps_setmiterlimit()

ps_setlinejoin

ps_setlinejoin -- Sets how contected lines are joined

Description

bool **ps_setlinejoin** (resource `$psdoc`, int `$type`)

Sets how lines are joined.

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`type`
: The way lines are joined. Possible values are *PS_LINEJOIN_MITER*, *PS_LINEJOIN_ROUND*, or *PS_LINEJOIN_BEVEL*.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_setlinecap()
- ps_setlinewidth()
- ps_setmiterlimit()

ps_setlinewidth

ps_setlinewidth -- Sets width of a line

Description

bool **ps_setlinewidth** (resource `$psdoc`, float `$width`)

Sets the line width for all following drawing operations.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`width`
The width of lines in points.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_setlinecap()
- ps_setlinejoin()
- ps_setmiterlimit()

ps_setmiterlimit

ps_setmiterlimit -- Sets the miter limit

Description

bool **ps_setmiterlimit** (resource `$psdoc`, float `$value`)

If two lines join in a small angle and the line join is set to *PS_LINEJOIN_MITER*, then the resulting spike will be very long. The miter limit is the maximum ratio of the miter length (the length of the spike) and the line width.

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`value`
: The maximum ratio between the miter length and the line width. Larger values (> 10) will result in very long spikes when two lines meet in a small angle. Keep the default unless you know what you are doing.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_setlinecap()
- ps_setlinejoin()
- ps_setlinewidth()

ps_setoverprintmode

ps_setoverprintmode -- Sets overprint mode

Description

bool **ps_setoverprintmode** (resource `$psdoc`, int `$mode`)

Warning
This function is currently not documented; only its argument list is available.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`mode`

Return Values

Returns **TRUE** on success or **FALSE** on failure.

ps_setpolydash

ps_setpolydash -- Sets appearance of a dashed line

Description

bool **ps_setpolydash** (resource `$psdoc`, float `$arr`)

Sets the length of the black and white portions of a dashed line. ps_setpolydash() is used to set more complicated dash patterns.

Parameters

psdoc
Resource identifier of the postscript file as returned by ps_new().

arr
arr is a list of length elements alternately for the black and white portion.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #8 - Drawing a dashed line

```
<?php
$ps = ps_new();
if (!ps_open_file($ps, "polydash.ps")) {
  print "Cannot open PostScript file\n";
    exit;
}

ps_set_info($ps, "Creator", "polydash.php");
ps_set_info($ps, "Author", "Uwe Steinmann");
ps_set_info($ps, "Title", "Poly dash example");

ps_begin_page($ps, 596, 842);
ps_setpolydash($ps, array(10, 5, 2, 5));
ps_moveto($ps, 100, 100);
ps_lineto($ps, 200, 200);
ps_stroke($ps);
ps_end_page($ps);

ps_delete($ps);
?>
```

This example draws a line with a 10 and 2 points long line, and gaps of 5 points inbetween.

See Also

- ps_setdash()

ps_shading_pattern

ps_shading_pattern -- Creates a pattern based on a shading

Description

int **ps_shading_pattern** (resource `$psdoc`, int `$shadingid`, string `$optlist`)

Creates a pattern based on a shading, which has to be created before with ps_shading(). Shading patterns can be used like regular patterns.

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`shadingid`
: The identifier of a shading previously created with ps_shading().

`optlist`
: This argument is not currently used.

Return Values

The identifier of the pattern or **FALSE** in case of an error.

See Also

- ps_shading()
- ps_shfill()

ps_shading

ps_shading -- Creates a shading for later use

Description

int **ps_shading** (resource `$psdoc`, string `$type`, float `$x0`, float `$y0`, float `$x1`, float `$y1`, float `$c1`, float `$c2`, float `$c3`, float `$c4`, string `$optlist`)

Creates a shading, which can be used by ps_shfill() or ps_shading_pattern().

The color of the shading can be in any color space except for *pattern*.

Parameters

psdoc
: Resource identifier of the postscript file as returned by ps_new().

type
: The type of shading can be either *radial* or *axial*. Each shading starts with the current fill color and ends with the given color values passed in the parameters *c1* to *c4* (see ps_setcolor() for their meaning).

x0, x1, y0, y1
: The coordinates *x0*, *y0*, *x1*, *y1* are the start and end point of the shading. If the type of shading is *radial* the two points are the middle points of a starting and ending circle.

c1, c2, c3, c4
: See ps_setcolor() for their meaning.

optlist
: If the shading is of type *radial* the *optlist* must also contain the parameters *r0* and *r1* with the radius of the start and end circle.

Return Values

Returns the identifier of the pattern or **FALSE** in case of an error.

See Also

- ps_shading_pattern()
- ps_shfill()

ps_shfill

ps_shfill -- Fills an area with a shading

Description

bool **ps_shfill** (resource `$psdoc`, int `$shadingid`)

Fills an area with a shading, which has to be created before with ps_shading(). This is an alternative way to creating a pattern from a shading ps_shading_pattern() and using the pattern as the filling color.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`shadingid`
The identifier of a shading previously created with ps_shading().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_shading()
- ps_shading_pattern()

ps_show_boxed

ps_show_boxed -- Output text in a box

Description

int **ps_show_boxed** (resource `$psdoc`, string `$text`, float `$left`, float `$bottom`, float `$width`, float `$height`, string `$hmode` [, string `$feature`])

Outputs a text in a given box. The lower left corner of the box is at (*`left`*, *`bottom`*). Line breaks will be inserted where needed. Multiple spaces are treated as one. Tabulators are treated as spaces.

The text will be hyphenated if the parameter *`hyphenation`* is set to *true* and the parameter *`hyphendict`* contains a valid filename for a hyphenation file. The line spacing is taken from the value *leading*. Paragraphs can be separated by an empty line just like in TeX. If the value *parindent* is set to value > 0.0 then the first n lines will be indented. The number n of lines is set by the parameter *numindentlines*. In order to prevent indenting of the first m paragraphs set the value *parindentskip* to a positive number.

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`text`
: The text to be output into the given box.

`left`
: x-coordinate of the lower left corner of the box.

`bottom`
: y-coordinate of the lower left corner of the box.

`width`
: Width of the box.

`height`
: Height of the box.

`hmode`
: The parameter *`hmode`* can be "justify", "fulljustify", "right", "left", or "center". The difference of "justify" and "fulljustify" just affects the last line of the box. In fulljustify mode the last line will be left and right justified unless this is also the last line of paragraph. In justify mode it will always be left justified.

`feature`

Used parameters

The output of ps_show_boxed() can be configured with several parameters and values which must be set with either ps_set_parameter() or ps_set_value(). Beside the parameters and values which affect text output, the following parameters and values are evaluated.

leading (value)
: Distance between baselines of two consecutive lines.

linebreak (parameter)
: Set to "true" if you want a carriage return to start a new line instead of treating it as a space. Defaults to "false".

parbreak (parameter)
: Set to "true" if you want a carriage return on a single line to start a new paragraph instead of treating it as a space. Defaults to "true".

hyphenation (parameter)
: Set to "true" in order to turn hyphenation on. This requires a dictionary to be set with the parameter "hyphendict". Defaults to "false".

hyphendict (parameter)
: Filename of the dictionary used for hyphenation pattern (see below).

hyphenminchar (value)
: The number of chars which must at least be left over before or after the hyphen. This implies that only words of at least two times this value will be hyphenated. The default value is three. Setting a value of zero will result in the default value.

parindent (value)
: Set the amount of space in pixel for indenting the first m lines of a paragraph. m can be set with the value "numindentlines".

parskip (value)
: Set the amount of extra space in pixel between paragraphs. Defaults to 0 which will result in a normal line distance.

numindentlines (value)
: Number of lines from the start of the paragraph which will be indented. Defaults to 1.

parindentskip (value)
: Number of paragraphs in the box whose first lines will not be indented. This defaults to 0. This is useful for paragraphs right after a section heading or text being continued in a second box. In both case one would set this to 1.

linenumbermode (parameter)
: Set how lines are to be numbered. Possible values are "box" for numbering lines in the whole box or "paragraph" to number lines within each paragraph.

linenumberspace (value)
: The space for the column left of the numbered line containing the line number. The line

number will be right justified into this column. Defaults to 20.

linenumbersep (value)
The space between the column with line numbers and the line itself. Defaults to 5.

Hyphenation

Text is hyphenated if the parameter *hyphenation* is set to true and a valid hyphenation dictionary is set. pslib does not ship its own hyphenation dictionary but uses one from openoffice, scribus or koffice. You can find their dictionaries for different languages in one of the following directories if the software is installed:

- */usr/share/apps/koffice/hyphdicts/*
- */usr/lib/scribus/dicts/*
- */usr/lib/openoffice/share/dict/ooo/*

Currently scribus appears to have the most complete hyphenation dictionaries.

Return Values

Number of characters that could not be written.

See Also

- ps_continue_text()

ps_show_xy2

ps_show_xy2 -- Output text at position

Description

bool **ps_show_xy2** (resource `$psdoc`, string `$text`, int `$len`, float `$xcoor`, float `$ycoor`)

Warning
This function is currently not documented; only its argument list is available.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

ps_show_xy

ps_show_xy -- Output text at given position

Description

bool **ps_show_xy** (resource `$psdoc`, string `$text`, float `$x`, float `$y`)

Output a text at the given text position.

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`text`
: The text to be output.

`x`
: x-coordinate of the lower left corner of the box surrounding the text.

`y`
: y-coordinate of the lower left corner of the box surrounding the text.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_continue_text()
- ps_show()

ps_show2

ps_show2 -- Output a text at current position

Description

bool **ps_show2** (resource `$psdoc`, string `$text`, int `$len`)

Warning
This function is currently not documented; only its argument list is available.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`text`

`len`

Return Values

Returns **TRUE** on success or **FALSE** on failure.

ps_show

ps_show -- Output text

Description

bool **ps_show** (resource `$psdoc`, string `$text`)

Output a text at the current text position. The text position can be set by storing the x and y coordinates into the values *textx* and *texty* with the function ps_set_value(). The function will issue an error if a font was not set before with ps_setfont().

ps_show() evaluates the following parameters and values as set by ps_set_parameter() and ps_set_value().

charspacing (value)
: Distance between two consecutive glyphs. If this value is unequal to zero then all ligatures will be resolved. Values less than zero are allowed.

kerning (parameter)
: Setting this parameter to "false" will turn off kerning. Kerning is turned on by default.

ligatures (parameter)
: Setting this parameter to "false" will turn off the use of ligatures. Ligatures are turned on by default.

underline (parameter)
: Setting this parameter to "true" will turn on underlining. Underlining is turned off by default.

overline (parameter)
: Setting this parameter to "true" will turn on overlining. Overlining is turned off by default.

strikeout (parameter)
: Setting this parameter to "true" will turn on striking out. Striking out is turned off by default.

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`text`
: The text to be output.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_continue_text()
- ps_show_xy()
- ps_setfont()

ps_string_geometry

ps_string_geometry -- Gets geometry of a string

Description

array **ps_string_geometry** (resource `$psdoc`, string `$text` [, int `$fontid` [, float `$size`]])

This function is similar to ps_stringwidth() but returns an array of dimensions containing the width, ascender, and descender of the text.

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`text`
: The text for which the geometry is to be calculated.

`fontid`
: The identifier of the font to be used. If not font is specified the current font will be used.

`size`
: The size of the font. If no size is specified the current size is used.

Return Values

An array of the dimensions of a string. The element 'width' contains the width of the string as returned by ps_stringwidth(). The element 'descender' contains the maximum descender and 'ascender' the maximum ascender of the string.

See Also

- ps_continue_text()
- ps_stringwidth()

ps_stringwidth

ps_stringwidth -- Gets width of a string

Description

float **ps_stringwidth** (resource `$psdoc`, string `$text` [, int `$fontid` [, float `$size`]])

Calculates the width of a string in points if it was output in the given font and font size. This function needs an Adobe font metrics file to calculate the precise width. If kerning is turned on, it will be taken into account.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`text`
The text for which the width is to be calculated.

`fontid`
The identifier of the font to be used. If not font is specified the current font will be used.

`size`
The size of the font. If no size is specified the current size is used.

Return Values

Width of a string in points.

See Also

- ps_string_geometry()

ps_stroke

ps_stroke -- Draws the current path

Description

bool **ps_stroke** (resource `$psdoc`)

Draws the path constructed with previously called drawing functions like ps_lineto().

Parameters

psdoc
: Resource identifier of the postscript file as returned by ps_new().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_closepath_stroke()
- ps_fill()
- ps_fill_stroke()

ps_symbol_name

ps_symbol_name -- Gets name of a glyph

Description

string **ps_symbol_name** (resource `$psdoc`, int `$ord` [, int `$fontid`])

This function needs an Adobe font metrics file to know which glyphs are available.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`ord`
The parameter *`ord`* is the position of the glyph in the font encoding vector.

`fontid`
The identifier of the font to be used. If not font is specified the current font will be used.

Return Values

The name of a glyph in the given font.

See Also

- ps_symbol()
- ps_symbol_width()

ps_symbol_width

ps_symbol_width -- Gets width of a glyph

Description

float **ps_symbol_width** (resource `$psdoc`, int `$ord` [, int `$fontid` [, float `$size`]])

Calculates the width of a glyph in points if it was output in the given font and font size. This function needs an Adobe font metrics file to calculate the precise width.

Parameters

`psdoc`
Resource identifier of the postscript file as returned by ps_new().

`ord`
The position of the glyph in the font encoding vector.

`fontid`
The identifier of the font to be used. If not font is specified the current font will be used.

`size`
The size of the font. If no size is specified the current size is used.

Return Values

The width of a glyph in points.

See Also

- ps_symbol()
- ps_symbol_name()

ps_symbol

ps_symbol -- Output a glyph

Description

bool **ps_symbol** (resource `$psdoc`, int `$ord`)

Output the glyph at position *`ord`* in the font encoding vector of the current font. The font encoding for a font can be set when loading the font with ps_findfont().

Parameters

`psdoc`
: Resource identifier of the postscript file as returned by ps_new().

`ord`
: The position of the glyph in the font encoding vector.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- ps_symbol_name()
- ps_symbol_width()

ps_translate

ps_translate -- Sets translation

Description

bool **ps_translate** (resource `$psdoc`, float `$x`, float `$y`)

Sets a new initial point of the coordinate system.

Parameters

psdoc
Resource identifier of the postscript file as returned by ps_new().

x
x-coordinate of the origin of the translated coordinate system.

y
y-coordinate of the origin of the translated coordinate system.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #9 - Translation of the coordinate system

```
<?php
function rectangle($ps) {
   ps_moveto($ps, 0, 0);
   ps_lineto($ps, 0, 50);
   ps_lineto($ps, 50, 50);
   ps_lineto($ps, 50, 0);
   ps_lineto($ps, 0, 0);
   ps_stroke($ps);
}

$ps = ps_new();
if (!ps_open_file($ps, "translate.ps")) {
 print "Cannot open PostScript file\n";
 exit;
}

ps_set_info($ps, "Creator", "translate.php");
ps_set_info($ps, "Author", "Uwe Steinmann");
```

```
ps_set_info($ps, "Title", "Translated example");
ps_set_info($ps, "BoundingBox", "0 0 596 842");

$psfont = ps_findfont($ps, "Helvetica", "", 0);

ps_begin_page($ps, 596, 842);
ps_set_text_pos($ps, 100, 100);
ps_translate($ps, 500, 750);
rectangle($ps);
ps_translate($ps, -500, -750);
ps_setfont($ps, $psfont, 8.0);
ps_show($ps, "Text at initial position");
ps_end_page($ps);

ps_begin_page($ps, 596, 842);
ps_set_text_pos($ps, 100, 100);
ps_save($ps);
ps_translate($ps, 500, 750);
rectangle($ps);
ps_restore($ps);
ps_setfont($ps, $psfont, 8.0);
ps_show($ps, "Text at initial position");
ps_end_page($ps);

ps_delete($ps);
?>
```

The above example demonstrates two possible ways to place a graphic (in this case just a rectangle) at any position on the page, while the graphic itself uses its own coordinate system. The trick is to change the origin of the current coordinate system before drawing the rectangle. The translation has to undone after the graphic has been drawn.

On the second page a slightly different and more elegant approach is applied. Instead of undoing the translation with a second call of ps_translate() the graphics context is saved before modifying the coordinate system and restored after drawing the rectangle.

See Also

- ps_scale()
- ps_rotate()