

PDFlib, PDFlib+PDI, PPS

A library for generating PDF on the fly
PDFlib 10.0.0

API Reference

For use with C, C++, Java, .NET, Objective-C, Perl, PHP, Python, RPG, Ruby



Copyright © 1997–2022 PDFlib GmbH and Thomas Merz. All rights reserved.
PDFlib users are granted permission to reproduce printed or digital copies of this manual for internal use.

PDFlib GmbH
Franziska-Bilek-Weg 9, 80339 München, Germany
www.pdflib.com
phone +49 • 89 • 452 33 84-0

sales@pdflib.com
support@pdflib.com (please include your license number)

This publication and the information herein is furnished as is, is subject to change without notice, and should not be construed as a commitment by PDFlib GmbH. PDFlib GmbH assumes no responsibility or liability for any errors or inaccuracies, makes no warranty of any kind (express, implied or statutory) with respect to this publication, and expressly disclaims any and all warranties of merchantability, fitness for particular purposes and noninfringement of third party rights.

PDFlib and the PDFlib logo are registered trademarks of PDFlib GmbH. PDFlib licensees are granted the right to use the PDFlib name and logo in their product documentation. However, this is not required.

PDFlib contains PANTONE® spot color data © Pantone LLC, 2019. PANTONE® colors displayed in the software application or in the user documentation may not match PANTONE-identified standards. Consult current PANTONE Color Publications for accurate color. PANTONE® and other Pantone, Inc. trademarks are the property of Pantone, Inc. © Pantone, LLC, 2019. Pantone, Inc. is the copyright owner of color data and/or software which are licensed to PDFlib GmbH to distribute for use only in combination with PDFlib Software. PANTONE Color Data and/or Software shall not be copied onto another disk or into memory unless as part of the execution of PDFlib Software.

PDFlib contains HKS® spot color data licensed from HKS Warenzeichenverband e.V.

*PDFlib contains the following third-party components:
Adobe CMap resources, Copyright © 1990-2019 Adobe
AES, Arcfour and SHA algorithms, Copyright © 1995-1998 Eric Young
Brotli decompression code, Copyright © 2009, 2010, 2013-2016 by the Brotli Authors
Curl multiprotocol file transfer library, Copyright © 1996-2021 Daniel Stenberg
Expat XML parser, Copyright © 2001-2019 Expat maintainers
ICCLib, Copyright © 1997-2021 Graeme W. Gill
ICU International Components for Unicode, Copyright © 1991-2020 Unicode, Inc.
libjpeg, Copyright © 1991-2020, Thomas G. Lane, Guido Vollbeding
libpng, Copyright © 2006-2017 Glenn Randers-Pehrson, Copyright © 2018-2019 Cosmin Truta
Libtiff image library, Copyright © 1988-1997 Sam Leffler, Copyright © 1991-1997 Silicon Graphics, Inc.
MD5 message digest, Copyright © 1991-2, RSA Data Security, Inc.
NotoSans fonts, Copyright © 2012 Google Inc. All Rights Reserved.
OpenJPEG library, Copyright © 2002-2014, Université catholique de Louvain (UCL), Belgium
OpenSSL cryptographic library, Copyright © 1998-2018 The OpenSSL Project
sRGB v2 ICC profile, Copyright © International Color Consortium 2016
WOFF2 font decompression, Copyright © 2013-2017 by the WOFF2 Authors
Zlib compression library, Copyright © 1995-2017 Jean-loup Gailly and Mark Adler*

*The PDFlib Block Plugin contains the following additional third-party component:
wxWidgets Cross-Platform GUI Library, Copyright © 2018 © 1998 Julian Smart, © 2018 wxWidgets*



Contents

1 Programming Concepts 7

- 1.1 Option Lists 7
 - 1.1.1 Syntax 7
 - 1.1.2 Simple Data Types 10
 - 1.1.3 Fontsize and Action Data Types 12
 - 1.1.4 Color Data Type 13
 - 1.1.5 Geometric Data Types 15
- 1.2 Method Scopes 17
- 1.3 Logging 18
- 1.4 Exception Handling 19
- 1.5 Unicode Conversion 21

2 General Methods 25

- 2.1 Global Options 25
- 2.2 Creating and Deleting PDFlib Objects 32
- 2.3 PDFlib Virtual File System (PVF) 34
- 2.4 Downloading Data from the Network 37
- 2.5 PDF Object Creation API (POCA) 39

3 Document and Page Methods 43

- 3.1 Document Methods 43
- 3.2 Fetching PDF Documents from Memory 54
- 3.3 Page Methods 55
- 3.4 Layers 61

4 Font and Text Methods 65

- 4.1 Font Handling 65
- 4.2 Text Filter and Appearance Options 76
- 4.3 Simple Text Output 81
- 4.4 User-defined Type 3 Fonts 85
- 4.5 User-defined 8-Bit Encodings 88

5 Text and Table Formatting 89

- 5.1 Single-Line Text with Textlines 89
- 5.2 Multi-Line Text with Textflows 96
- 5.3 Table Formatting 113

6	Object Fitting and Matchboxes	123
6.1	Object Fitting	123
6.2	Matchboxes	131
6.2.1	Defining Matchboxes	131
6.2.2	Using Matchboxes	131
6.2.3	Querying Matchboxes	133
7	Graphics Methods	135
7.1	Graphics Appearance Options	135
7.2	Graphics State	139
7.3	Coordinate System Transformations	143
7.4	Path Construction	146
7.5	Painting and Clipping	150
7.6	Path Objects	152
8	Color Methods	159
8.1	Setting Color	159
8.2	ICC Profiles	161
8.3	Spot Colors	163
8.4	DeviceN Colors	164
8.5	Shadings and Shading Patterns	166
8.6	Tiling Patterns	169
9	Image, SVG and Template Methods	171
9.1	Images	171
9.2	SVG Graphics	179
9.3	Templates (Form XObjects)	186
9.4	Common XObject Options	190
10	PDF Import (PDI) and pCOS Methods	193
10.1	PDF Document Import Methods	193
10.2	PDF Page Import Methods	198
10.3	Other PDI Processing	205
10.4	pCOS Methods	207
11	Block Filling Methods (PPS)	211
11.1	Options for all Block Types	211
11.1.1	Rectangle Options	211
11.1.2	Tagging Option	212
11.2	Textline and Textflow Blocks	213

11.3	Image Blocks	215
11.4	PDF Blocks	216
11.5	Graphics Blocks	217
12	Interactive Features	219
12.1	Bookmarks	219
12.2	Annotations	221
12.3	Form Fields	229
12.4	Actions	237
12.5	Named Destinations	242
13	Multimedia, 3D and Geospatial Features	245
13.1	Multimedia Assets and Attachments	245
13.1.1	Loading Assets	245
13.1.2	Options for Rendition Assets	247
13.1.3	Options for other Asset Types	249
13.2	Suboptions for RichMedia Annotations	251
13.3	3D Artwork	254
13.3.1	3D Views for RichMedia and 3D Annotations	254
13.3.2	Loading Contents for 3D Annotations	258
13.3.3	Options for 3D Annotations	259
13.4	Geospatial Features	260
14	Tagged PDF and Marked Content	263
14.1	Tagging Methods	263
14.2	Abbreviated Tagging	271
14.3	Marked Content	272
15	Document Interchange	275
15.1	Document Information Fields	275
15.2	XMP Metadata	277
15.3	PDF Packages and Portfolios	278
15.4	Document Part Hierarchy	282
A	List of API Methods	285
B	List of all Options and Keywords	287
C	Revision History	303
	Index	305

1 Programming Concepts

API methods in this chapter:

- ▶ `PDF_get_errnum()`
- ▶ `PDF_get_errmsg()`
- ▶ `PDF_get_apiname()`
- ▶ `PDF_get_opaque()`
- ▶ `PDF_convert_to_unicode()`

1.1 Option Lists

Option lists are a powerful yet easy method for controlling API method calls. Instead of requiring a multitude of method parameters, many API methods support option lists, or *optlists* for short. These are strings which can contain an arbitrary number of options. Option lists support various data types and composite data like lists. In most language bindings optlists can easily be constructed by concatenating the required keywords and values.

Bindings .NET language binding: C# programmers should keep in mind that the `AppendFormat()` `StringBuilder` method uses the `{` and `}` braces to represent format items which will be replaced by the string representation of arguments. On the other hand, the `Append()` method does not impose any special meaning on the brace characters. Since the option list syntax makes use of the brace characters, care must be taken in selecting the `AppendFormat()` or `Append()` method appropriately.

1.1.1 Syntax

Formal option list syntax definition. Option lists must be constructed according to following rules:

- ▶ All elements (keys and values) in an option list must be separated by one or more of the following separator characters: space, tab, carriage return, newline, equal sign '='.
- ▶ An outermost pair of enclosing braces is not part of the element. The sequence `{ }` designates an empty element.
- ▶ Separators within the outermost pair of braces no longer split elements, but are part of the element. Therefore, an element which contains separators must be enclosed with braces.
- ▶ If an element contains brace characters these must be protected with a preceding backslash character.
- ▶ If an element contains a sequence of one or more backslash characters in front of a brace, each backslash in the sequence must be protected with another backslash character.
- ▶ Option lists must not contain binary zero values.

An option may have a list value according to its documentation in this PDFlib API Reference. List values contain one or more elements (which may themselves be lists). They are separated according to the rules above, with the only difference that the equal sign is no longer treated as a separator.

Note Option names (i.e. the key) never contain hyphen characters. Keep this in mind since the tables with option descriptions may sometimes contain long option names which are hyphenated. The hyphen must be omitted when supplying the option in an option list.

Simple option lists. In many cases option lists will contain one or more key/value pairs. Keys and values, as well as multiple key/value pairs must be separated by one or more whitespace characters (space, tab, carriage return, newline). Alternatively, keys can be separated from values by an equal sign '=':

```
key=value
key = value
key value
key1 = value1 key2 = value2
```

To increase readability we recommend to use equal signs between key and value and whitespace between adjacent key/value pairs.

Since option lists will be evaluated from left to right an option can be supplied multiply within the same list. In this case the last occurrence will overwrite earlier ones. In the following example the first option assignment will be overridden by the second, and *key* will have the value *value2* after processing the option list:

```
key=value1 key=value2
```

List values. Lists contain one or more separated values, which may be simple values or list values in turn. Lists are bracketed with { and } braces, and the values in the list must be separated by whitespace characters. Examples:

```
dasharray={11 22 33}           (list containing three numbers)
position={ center bottom }     (list containing two keywords)
```

A list may also contain nested lists. In this case the lists must also be separated from each other by whitespace. While a separator must be inserted between adjacent } and { characters, it can be omitted between braces of the same kind:

```
polylinelist={{10 20 30 40} {50 60 70 80}} (list containing two lists)
```

If the list contains exactly one list the braces for the nested list must not be omitted:

```
polylinelist={{10 20 30 40}} (list containing one nested list)
```

Nested option lists and list values. Some options accept the type *option list* or *list of option lists*. Options of type *option list* contain one or more subordinate options. Options of type *list of option lists* contain one or more nested option lists. When dealing with nested option lists it is important to specify the proper number of enclosing braces. Several examples are listed below.

The value of the option *metadata* is an option list which itself contains a single option *filename*:

```
metadata={filename=info.xmp}
```

The value of the option *fill* is a list of option lists containing a single option list:

```
fill={{ area=table fillcolor={rgb 1 0 0} }}
```

The value of the option *fill* is a list of option lists containing two option lists:

```
fill={{ area=rowodd fillcolor={rgb 0 1 0} } { area=roweven fillcolor={rgb 1 0 0} }}
```

List containing one option list with a value that includes spaces:

```
attachments={{filename={foo bar.xml} }}
```

List containing three strings:

```
itemnamelist = { {Isaac Newton} {James Clark Maxwell} {Albert Einstein} }
```

List containing two keywords:

```
position={left bottom}
```

List containing different types (float and keyword):

```
position={10 bottom}
```

List containing one rectangle:

```
boxes={{10 20 30 40}}
```

List containing two polylines with percentages:

```
polygons = {{10 20 40 60 90 120}} {12 87 34 98 34% 67% 34% 7%}}
```

Common traps and pitfalls. This paragraph lists some common errors regarding option list syntax.

Braces are not separators; the following is wrong:

```
key1 {value1}key2 {value2}                WRONG!
```

This will trigger the error message *Unknown option 'value2'*. Similarly, the following are wrong since the separators are missing:

```
key{value}                                WRONG!  
key={{value1}{value2}}                  WRONG!
```

Braces must be balanced; the following is wrong (see below for unquoted string syntax):

```
key={open brace { }                      WRONG!
```

This will trigger the error message *Braces aren't balanced in option list 'key={open brace {}'*. A single brace as part of a string must be preceded by an additional backslash character:

```
key={closing brace \} and open brace \{ }  CORRECT!
```

A backslash at the end of a string value must be preceded by another backslash if it is followed by a closing brace character:

```
key={\value\}                            WRONG!  
key={\value\\}                            CORRECT!
```

1.1.2 Simple Data Types

String. Strings are plain ASCII strings (or EBCDIC strings on EBCDIC platforms) which are generally used for keywords. Strings containing whitespace or '=' characters must be bracketed with { and }:

```
password={ secret string }           (string value contains three blanks)
contents={length=3mm}                (string value containing one equal sign)
```

The characters { and } must be preceded by an additional \ character if they are supposed to be part of the string:

```
password={weird\}string}            (string value contains a right brace)
```

A backslash in front of the closing brace of an element must be preceded by a backslash character:

```
filename={C:\path\name\\}          (string ends with a single backslash)
```

An empty string can be constructed with a pair of braces:

```
{}
```

Content strings, hypertext strings and name strings: these can hold Unicode content in various formats. Single bytes can be expressed by an escape sequence if the option *escapesequence* is set. For details on these string types and encoding choices for string options see the *PDFlib Tutorial*.

C and Perl, PHP, Ruby with *stringformat=legacy*: if an option list starts with a [EBCDIC-]UTF-8 BOM, each content, hypertext or name string of the option list is interpreted as a [EBCDIC-]UTF-8 string.

Unquoted string values. In the following situations the actual characters in an option value may conflict with optlist syntax characters:

- ▶ Passwords or file names may contain unbalanced braces, backslashes and other special characters
- ▶ Supplying JavaScript code in options is problematic due to the use of { and } braces

In order to provide a simple mechanism for supplying arbitrary text or binary data which does not interfere with option list syntax elements, unquoted option values can be supplied along with a length specifier in the following syntax variants:

```
key[n]=value
key[n]={value}
```

The decimal number *n* contains the number of code units required to represent *value*, e.g. bytes in language bindings with *stringformat=utf8* or UTF-16 code units in Java or .NET.

The braces around the string value are optional, but strongly recommended. They are required for strings starting with a space or other separator character. Braces, separators and backslashes within the string value are taken literally without any special interpretation.

Example for specifying a 7-character password containing space and brace characters. The whole string is surrounded by braces which are not part of the option value:

```
password[7]={ ab}c d}
```

If an option value in a nested option list is provided with a length count, the enclosing option list must also supply a length count, e.g.

```
fitannotation[34]={contents[19]={this is a brace '}'}}
```

Unichar. A Unichar is a single Unicode value where several syntax variants are supported: decimal values ≥ 10 (e.g. 173), hexadecimal values prefixed with *x*, *X*, *ox*, *oX*, or *U+* (*xAD*, *oxAD*, *U+ooAD*), numerical references, character references, and glyph name references but without the *'&'* and *';* decoration (*shy*, *#xAD*, *#173*). Alternatively, literal characters can be supplied. Examples:

replacementchar=?	(literal)
replacementchar=63	(decimal)
replacementchar=x3F	(hexadecimal)
replacementchar=0x3F	(hexadecimal)
replacementchar=U+003F	(Unicode notation)
replacementchar=euro	(HTML character reference)
replacementchar=.question	(standard glyph name reference)
replacementchar=.marina	(font-specific glyph name reference)

Single characters which happen to be a number are treated literally, not as decimal Unicode values:

```
replacementchar=3 (U+0033 THREE, not U+0003!)
```

Unichars must be in the hexadecimal range *0-0x10FFFF* (decimal *0-1114111*). However, some options are restricted to the range *0-0xFFFF* (*0-65535*). This is noted in the respective option description.

Unicode range. A Unicode range identifies a contiguous range of Unicode characters via start and end characters of the range. The start and end values of a Unicode range must be separated by a minus sign *'-'* without any spaces, e.g.

```
forcechars={U+03AC-U+03CE}
```

Boolean. Booleans have the values *true* or *false*; if the value of a Boolean option is omitted, the value *true* is assumed. As a shorthand notation *noname* can be used instead of *name=false*:

embedding	(equivalent to embedding=true)
noembedding	(equivalent to embedding=false)

Keyword. An option of type keyword can hold one of a predefined list of fixed keywords. Example:

```
blendmode=overlay
```

For some options the value hold either a number or a keyword.

Number. Option list support several numerical types.

Integer types can hold decimal and hexadecimal integers. Positive integers starting with *x*, *X*, *ox*, or *oX* specify hexadecimal values:

```
-12345
0
0xFF
```

Floats can hold decimal floating point or integer numbers; period and comma can be used as decimal separators for floating point values. Exponential notation is also supported. The following values are all equivalent:

```
size = -123.45
size = -123,45
size = -1.2345E2
size = -1.2345e+2
```

Percentages are numbers with a % character directly after the numerical value. Some options allow negative percentages:

```
leading=120%
topoffset=-20.5%
```

Handle. Handles identify various types of objects, e.g. fonts, images, ICC profiles or actions. Technically these are integer values which have been returned earlier by an API method. For example, an image handle is returned by `PDF_load_image()`. Handles must always be treated as opaque types; they must never be modified or created by the application directly (as opposed to using a handle returned by an API method). Handles must always be valid for the respective type of object. For example, an option which expects an image handle must not be supplied with a graphics handle, although both handles are integer types.

1.1.3 Fontsize and Action Data Types

Fontsize. A fontsize can be defined in several ways which allow the size of text to be specified in absolute values, relative to some external entity, or relative to some font property. In general the fontsize must be different from 0 unless the option description mentions otherwise.

In the most common case a fontsize contains a single float value which specifies refers to units in the user coordinate system:

```
fontsize=12
```

The second variant contains a percentage, where the basis of the percentage depends on the context (e.g. the width of the fitbox for `PDF_fit_textline()`):

```
fontsize=8%
```

In the third variant, the fontsize is specified as an option list which must contain a keyword and a number. The keyword describes the desired font metric according to Table 1.1, and the number contains the desired size. PDFlib will calculate the proper fontsize so that the selected text metric matches the supplied value:

```
fontsize={capheight 5}
```

Action list. An action list specifies one or more actions. Each entry in the list consists of an event keyword (trigger) and a list of action handles which must have been created

Table 1.1 Suboptions for options of type fontsize

option	description
<i>ascender</i>	The number is interpreted as ascender height.
<i>bodyheight</i>	The number is interpreted as minimum distance between baselines, i.e. descenders and ascenders of adjacent lines may exactly touch if this value is used as leading. This is the default behavior if no keyword is provided.
<i>capheight</i>	The number is interpreted as capital letter height.
<i>xheight</i>	The number is interpreted as lowercase letter height.

with `PDF_create_action()`. Actions will be performed in the listed order. The set of allowed events (e.g. `docopen`) and the type of actions (e.g. JavaScript) are documented separately for the respective options.

List containing a single trigger with three actions:

```
action={ activate={ 0 1 2 } }
```

List containing three triggers with one action for each:

```
action={ keystroke=0 format=1 validate=2 }
```

1.1.4 Color Data Type

Overview of color spaces. You can specify the colors for filling and stroking paths and text characters. Colors can be specified in several color spaces (see PDFlib Tutorial for a full discussion of color spaces and values). Each item in the following list starts with the corresponding color space keyword for color options):

- ▶ *gray*: Gray values between 0=black and 1=white;
- ▶ *rgb*: RGB triples, i.e. three values between 0 and 1 specifying the percentage of red, green, and blue; (0, 0, 0)=black, (1, 1, 1)=white. The commonly used RGB color values in the range 0–255 must be divided by 255 in order to scale them to the range 0–1 as required by PDFlib.

As an alternative to numerical RGB values you can specify RGB colors via their HTML names or hexadecimal values.

- ▶ *cmyk*: Four CMYK values between 0 = no color and 1 = full color, representing cyan, magenta, yellow, and black values; (0, 0, 0, 0)=white, (0, 0, 0, 1)=black. Note that this is different from the RGB specification.
- ▶ *iccbased* (not for `PDF_setcolor()`) and *iccbasedgray/rgb/cmyk*: ICC-based colors are based on an ICC profile.
- ▶ *spotname*: name of a predefined spot color and a tint value in the range 0=no color to 1=maximum intensity.

Alternatively, the name of a custom spot color, a tint value, and an alternate representation in one of the other color spaces above.

- ▶ *spot*: handle for a predefined or custom spot color created with `PDF_makespotcolor()` and a tint value.
- ▶ *devicen*: handle for a DeviceN color space created with `PDF_create_devicen()` and N tint values for the named colorants. Tint values are in the range 0=no color to 1=maximum intensity.

- ▶ *lab* expects device-independent colors in the CIE L*a*b* color space. Colors are specified by a luminance value in the range 0-100 and two color values *a* and *b* in the range -128 to 127. The *a* component ranges from green to red/magenta (negative values indicate green, positive values indicate magenta), and the *b* component ranges from blue to yellow (negative values indicate blue, positive values indicate yellow).
- ▶ *pattern*: shading pattern identified by a pattern handle. A shading pattern provides a gradual transition between two or more colors and can be created with `PDF_shading_pattern()` based on a shading handle created with `PDF_shading()`.
- ▶ *pattern*: tiling pattern identified by a pattern handle. A tiling pattern contains arbitrary text, vector graphics, or images which are repeatedly tiled across the area to be filled. Tiling patterns can be created with `PDF_begin_pattern_ext()`.

The default color for stroke and fill operations is black. The color space for this default color is selected automatically to match PDF/X and PDF/A color requirements.

Color options. Color options can be defined in three different forms: using an RGB color name, hexadecimal RGB values, or a flexible option list for colors in any color space.

In the first form all valid color names from SVG 1.1 can be supplied directly to specify an RGB color or an sRGB color if the sRGB ICC profile has been selected, e.g.

```
strokecolor=pink
```

Color names are case-insensitive. A list of valid RGB color names can be found at the following location:

www.w3.org/TR/SVG11/types.html#ColorKeywords

In the second form a hash '#' character followed by any three pairs of hexadecimal digits *00-FF* can be supplied to specify an RGB color value, e.g.

```
strokecolor=#FFC0CB
```

Cookbook In the third form an color option list specified a color space and color value. A color option list contains a color space keyword and a list with a variable number of float values depending on the particular color space. Table 1.2 contains descriptions of color space keywords with examples. As detailed in the respective method descriptions, a particular option list may support only a subset of the color space keywords.

Table 1.2 Keywords for the color data type in option lists

keyword	additional values	example
<i>gray</i>	single float value for the grayscale color space	{ gray 0.5 }
<i>rgb</i>	three float values for the RGB color space	{ rgb 1 0 0 }
(no keyword)	HTML color name or hexadecimal values for an RGB color	pink #FFC0CB
<i>cmyk</i>	four float values for the CMYK color space	{ cmyk 0 1 0 0 }
<i>lab</i>	three float values for the Lab color space	{ lab 100 50 30 }
<i>spot</i>	spot color handle created with <code>PDF_makespotcolor()</code> , followed by a float specifying the tint value	{ spot <handle> 0.8 }

Table 1.2 Keywords for the color data type in option lists

keyword	additional values	example
spotname	(up to 63 bytes; fewer Unicode characters depending on format and encoding) spot color name and a float specifying the tint value in the range 0..1	{ spotname {PANTONE 281 U} 0.5 }
spotname	Similar to the simple form of spotname above, but a color value can be added to specify the alternate color for a custom spot color (i.e. a spot color name which is not known internally to PDFlib). If multiple options define the same custom spot color name all definitions must be consistent (i.e. define the same alternate color).	{ spotname {PDFlib Blue} 0.5 { lab 100 50 30 } }
devicen	DeviceN color space handle created with <code>PDF_create_devicen()</code> followed by N float values for the tint values of the colorants in the range 0..1.	{ devicen <handle> 0.8 0.9 } { devicen <handle> 0 0 0.1 0.2 }
iccbased	ICC profile handle or keyword <code>srgb</code> , plus 1, 3 or 4 color values depending on the type of ICC profile (gray, RGB, or CMYK). The <code>srgb</code> keyword must not be used in document scope.	{ iccbased <handle> 0.5 } { iccbased <handle> 0 0 0.75 } { iccbased srgb 0 0 0.75 } { iccbased <handle> 0 0 0.3 1 }
iccbasedgray	single float value referring to an ICC profile selected with the option <code>iccprofilegray</code>	{ iccbasedgray 0.5 }
iccbasedrgb	three float values value referring to an ICC profile selected with the option <code>iccprofilergb</code>	{ iccbasedrgb 1 0 0 }
iccbasedcmyk	four float values value referring to an ICC profile selected with the option <code>iccprofilecmyk</code>	{ iccbasedcmyk 0 1 0 0 }
pattern	shading pattern handle created with <code>PDF_shading_pattern()</code>	{ pattern <handle> }
pattern	tiling pattern handle created with <code>PDF_begin_pattern_ext()</code>	{ pattern <handle> }
none	specifies the absence of color	none

1.1.5 Geometric Data Types

Line. A line is a list of four float values specifying the x and y coordinates of the start and end point of a line segment. The coordinate system for interpreting the coordinates (default or user coordinate system) varies depending on the option, and is documented separately:

```
line = {10 40 130 90}
```

Polyline. A polyline is a list containing one or more points where each point is described by a pair of float values. Each pair in the list specifies the x and y coordinates of a point; these points will be connected by line segments. The coordinate system for interpreting the coordinates (default or user coordinate system) varies depending on the option, and is documented separately:

```
polyline = {10 20 30 40 50 60}
```

The following option lists are equivalent:

```
polyline = {10 20 30r 40r 50r 60r}  
polyline = {10 20 40 60 90 120}
```

Quadrilaterals are a special type of polylines: these are rectangles which may be rotated and for which exactly four points must be specified.

Another special type are polygons: these are polylines which will automatically be closed by a line segment.

Rectangle. A rectangle is a list of four float values specifying the x and y coordinates of the lower left and upper right corners of a rectangle. The coordinate system for interpreting the coordinates (default or user coordinate system) varies depending on the option, and is documented separately. Some options accept percentages, where the basis for the percentage depends on the context (e.g. the fitbox of a Textflow). Examples:

```
cropbox={ 0 0 500 600 }  
box={40% 30% 50% 70%}
```

Relative coordinates. Some options support relative coordinates. Relative coordinates can be supplied by adding the suffix r immediately after a number. The option list description explains the basis to which relative coordinates relate (e.g. a Block rectangle). Within a coordinate list a relative coordinate relates to the previous x or y coordinate. Relative coordinates at the beginning of a list relate to the origin, i.e. they are absolute coordinates.

The following options are equivalent:

```
box={12 34 56r 78r}  
box={12 34 68 112}
```

Circle. A circle is specified as a list of four float values where the first pair specifies the x and y coordinates of the center, and the second pair specifies the x and y coordinates of an arbitrary point on the circle. The coordinate system for interpreting the coordinates (default or user coordinate system) varies depending on the option, and is documented separately:

```
circle={200 325 200 200}
```

Curve list. A curve list consists of two or more connected third-order Bézier curve segments. A Bézier curve is specified by four control points. The first control point is the starting point and the fourth point is the end point of the curve. The second and third point control the shape of the curve. In a curve list the last point of a segment serves as the first point for the next segment:

```
curve={200 700 240 600 80 580 400 660 400 660 440 620}
```

The last control point will become the new current point after drawing the curves.

1.2 Method Scopes

PDFlib applications must obey certain structural rules which are easy to understand. For example, you obviously begin a document before ending it. PDFlib enforces correct ordering of method calls with a strict scoping system. The scope definitions can be found in Table 1.3. All API method descriptions specify the allowed scope for each method. Calling a method outside of the allowed scopes results in an exception. You can query the current scope with the *scope* keyword of `PDF_get_option()`.

Table 1.3 Method scope definitions

scope name	definition
path	started by one of <code>PDF_moveto()</code> , <code>PDF_circle()</code> , <code>PDF_arc()</code> , <code>PDF_arcn()</code> , <code>PDF_rect()</code> , <code>PDF_ellipse()</code> or <code>PDF_elliptical_arc()</code> ; terminated by any of the methods in Section 7.5, »Painting and Clipping«, page 150
page	between <code>PDF_begin_page_ext()</code> and <code>PDF_end_page_ext()</code> , but outside of path scope
template	between <code>PDF_begin_template_ext()</code> and <code>PDF_end_template_ext()</code> , but outside of path scope
pattern	between <code>PDF_begin_pattern_ext()</code> and <code>PDF_end_pattern()</code> , but outside of path scope
font	between <code>PDF_begin_font()</code> and <code>PDF_end_font()</code> , but outside of glyph scope
glyph	between <code>PDF_begin_glyph_ext()</code> and <code>PDF_end_glyph()</code> , but outside of path scope
document	between <code>PDF_begin_document()</code> and <code>PDF_end_document()</code> , but outside of page, path, glyph, template, pattern, and font scope
object	during the lifetime of the PDFlib object, but outside of document scope; in the C and RPG language bindings between <code>PDF_new()</code> and <code>PDF_delete()</code> , but outside of document scope

1.3 Logging

The logging feature can be used to trace API calls. The contents of the log file may be useful for debugging purposes, or may be requested by PDFlib GmbH support. Logging options can be supplied in the following ways:

- ▶ As an option list for the global *logging* option of *PDF_set_option()*, e.g.:

```
p.set_option("logging={filename=trace.log remove}");
```
- ▶ In an environment variable called *PDFLIBLOGGING*. This will activate the logging output starting with the very first call to one of the API methods.

Table 1.4 Suboptions for the logging option

option	description
(empty list)	Enable log output
disable	(Boolean) Disable logging output
enable	(Boolean) Enable logging output
filename	(String) Name of the log file; stdout and stderr are recognized as special names. On CICS this option is ignored, and logging output is always written to stderr. Output is appended to existing contents. Default: pdflog on z/OS PDFlib.log on macOS and IBM System i \\PDFlib.log on Windows /tmp/PDFlib.log on all other systems The log file name can alternatively be supplied in the environment variable PDFLIBLOGFILE.
flush	(Boolean) If true, the log file will be closed after each output, and reopened for the next output to make sure that the output will actually be flushed. This may be useful when chasing program crashes where the log file is truncated, but significantly slows down processing. If false, the log file will be opened only once. Default: false
includepid includetid includeoid	(Boolean; not on MVS) Include the process id, thread id, or object id in the log file name. This should be enabled if multiple processes use the same log file. Default: false
remove	(Boolean) If true, an existing log file is deleted before writing new output. Default: false
removeon-success	(Boolean) Remove the generated log file in <i>PDF_delete()</i> unless an exception occurred. This may be useful for analyzing occasional problems or problems which occur only sporadically. It is recommended to combine this option with includepid/includetid/includeoid as appropriate.

Table 1.4 Suboptions for the logging option

option	description
classes	(Option list) List containing options of type integer, where each option describes a logging class and the corresponding value describes the granularity level. Level 0 disables a logging class, positive numbers enable a class. Increasing levels provide more and more detailed output. The following options are provided (default: {api=1 warning=1}):
api	Log all API calls with their method parameters and results. If api=2 a timestamp will be created in front of all API trace lines, and deprecated methods and options will be marked.
filesearch	Log all attempts related to locating files via SearchPath or PVF.
resource	Log all attempts at locating resources via Windows registry, UPR definitions as well as the results of the resource search.
tagging	Structure element (tagging) operations
user	User-specified logging output supplied with the userlog option.
warning	Log all PDFlib warnings, i.e. error conditions which can be ignored or fixed internally. If warning=2 messages from methods which do not throw any exception, but hook up the message text for retrieval via <code>PDF_get_errmsg()</code> , and the reason for all failed attempts at opening a file (searching for a file in searchpath) will also be logged.

1.4 Exception Handling

Table 1.5 details the relevant option for this section. This option is supported by many methods as indicated in the corresponding option list descriptions. It can also be supplied as global option to `PDF_set_option()` (see Section 2.1, »Global Options«, page 25).

Table 1.5 Exception-related option for `PDF_set_option()`

option	description
errorpolicy	(Keyword) Controls the error behavior of API methods which return a value. The global option errorpolicy can be overridden by the errorpolicy option of many methods, and serves as default for this option. Supported keywords (default: return):
return	If a processing error occurs the method will return. Methods which can return an error code, e.g. <code>PDF_load_image()</code> , return -1 (in PHP: 0). Methods which return result strings, e.g. <code>PDF_fit_table()</code> , return the string_error. Application developers must check the return value against -1 (in PHP: 0) or <code>_error</code> to detect error situations. In case of an error a detailed description can be queried with <code>PDF_get_errmsg()</code> . This setting is recommended for new applications. Even with errorpolicy=return an exception is thrown if a syntax error in an option list is found or the PDF output cannot be written.
exception	If an error occurs the method will throw an exception. The exception must be caught in client code. The partial PDF output generated so far is unusable and must be discarded (this can be automated with the <code>removefragments</code> document option).

C++ Java C# **int get_errnum()**

Perl PHP **int get_errnum()**

C **int PDF_get_errnum(PDF *p)**

Get the number of the last thrown exception or the reason of a failed method call.

Returns The error code of the most recent error condition.

Scope Between an exception thrown by PDFlib and the death of the PDFlib object. Alternatively, this method may be called after a method returned a -1 (in PHP: o) error code, but before calling any other method except those listed in this section.

Bindings In C++, Java, Objective-C, .NET and PHP this method is also available as `get_errnum()` in the `PDFlibException` object.

C++ Java C# **String** `get_errmsg()`

Perl PHP **string** `get_errmsg()`

C **const char ***`PDF_get_errmsg(PDF *p)`

Get the text of the last thrown exception or the reason of a failed method call.

Returns Text containing the description of the most recent error condition.

Scope Between an exception thrown by PDFlib and the death of the PDFlib object. Alternatively, this method may be called after a method returned a -1 (in PHP: o) error code, but before calling any other method except those listed in this section.

Bindings In C++, Java, Objective-C, .NET and PHP this method is also available as `get_errmsg()` in the `PDFlibException` object.

C++ Java C# **String** `get_apiname()`

Perl PHP **string** `get_apiname()`

C **const char ***`PDF_get_apiname(PDF *p)`

Get the name of the API method which threw the last exception or failed.

Returns The name of the API method which threw an exception, or the name of the most recently called method which failed with an error code.

Scope Between an exception thrown by PDFlib and the death of the PDFlib object. Alternatively, this method may be called after a method returned a -1 (in PHP: o) error code, but before calling any other method except those listed in this section.

Bindings In C++, Java, Objective-C, .NET and PHP this method is also available as `get_apiname()` in the `PDFlibException` object.

C++ **void ***`get_opaque()`

C **void ***`PDF_get_opaque(PDF *p)`

Fetch the opaque application pointer stored in PDFlib.

Returns The opaque application pointer stored in PDFlib which has been supplied in the call to `PDF_new2()`.

Details PDFlib never touches the opaque pointer, but supplies it unchanged to the client. This may be used in multi-threaded applications for storing private thread-specific data within the PDFlib object. It is especially useful for thread-specific exception handling.

Scope any

Bindings Only available in the C and C++ bindings.

1.5 Unicode Conversion

C++ `string convert_to_unicode(string inputformat, string input, string optlist)`

Java C# `string convert_to_unicode(string inputformat, byte[] input, string optlist)`

Perl PHP `string convert_to_unicode(string inputformat, string input, string optlist)`

C `const char *PDF_convert_to_unicode(PDF *p,
const char *inputformat, const char *input, int inputlen, int *outputlen, const char *optlist)`

Convert a string in an arbitrary encoding to a Unicode string in various formats.

inputformat Unicode text format or encoding name specifying interpretation of the input string:

- ▶ Unicode text formats: *utf8*, *ebcdicutf8* (on EBCDIC platforms), *utf16*, *utf16le*, *utf16be*, *utf32*
- ▶ Only if the *font* option is specified: *builtin*, *glyphid*
- ▶ All configured 8-bit encodings and encodings available on the host system
- ▶ The keyword *auto* specifies the following behavior: if the input string contains a UTF-8 or UTF-16 BOM it is used to determine the appropriate format, otherwise the current system codepage is assumed.

input String to be converted to Unicode.

inputlen (C and RPG language bindings only) Length of the input string in bytes. If *inputlen=0* a null-terminated string must be provided.

outputlen (C and RPG language bindings only) C-style pointer to a memory location where the length of the returned string (in bytes) will be stored.

optlist An option list specifying options for input interpretation and Unicode conversion:

- ▶ Text filter options according to Table 4.5: *charref*, *escapesequence*
- ▶ Unicode conversion options according to Table 1.6:
bom, *errorpolicy*, *font*, *inflate*, *outputformat*

Returns A Unicode string created from the input string according to the specified parameters and options. If the input string does not conform to the specified input format (e.g. invalid UTF-8 string) an empty output string is returned if *errorpolicy=return*, and an exception will be thrown if *errorpolicy=exception*.

Details This method may be useful for general Unicode string conversion. It is provided for the benefit of users who work in environments without suitable Unicode converters.

Scope any

Bindings C language binding: the returned strings are stored in a ring buffer with up to 10 entries. If more than 10 strings are converted the buffers are reused, which means that clients must copy the strings if they want to access more than 10 strings in parallel. For example, up to 10 calls to this method can be used as parameters for a *printf()* statement since the return strings are guaranteed to be independent if no more than 10 strings are used at the same time.

C++ language binding: The parameters *inputformat* and *optlist* have the configured string type, while *input* and the returned data have type *std::string*.

Table 1.6 Options for *PDF_convert_to_unicode()*

option	description
bom	<p>(Keyword; ignored for <i>outputformat=utf32</i>; in .NET, Java, Objective-C and Python only none is allowed) Policy for adding a byte order mark (BOM) to the output string. Supported keywords (default: none) :</p> <p>add Add a BOM.</p> <p>keep Add a BOM if the input string has a BOM.</p> <p>none Don't add a BOM.</p> <p>optimize Add a BOM except if <i>outputformat=utf8</i> or <i>ebcdicutf8</i> and the output string contains only characters in the range < U+007F.</p>
errorpolicy	<p>(Keyword) Behavior in case of conversion errors (default: the value of the <i>errorpolicy</i> global option, see Table 1.5):</p> <p>return The replacement character will be used if a character reference cannot be resolved or a code or glyph ID doesn't exist in the specified font. An empty string will be returned in case of conversion errors.</p> <p>exception An exception will be thrown in case of conversion errors.</p>
font	<p>(Font handle; required for <i>inputformat=builtin</i> and <i>glyphid</i>) Apply font-specific conversion according to the specified font.</p>
inflate	<p>(Boolean; only for <i>inputformat=utf8</i>) If true, an invalid UTF-8 input string will not trigger an exception, but rather an inflated byte string in the specified output format will be generated. The inflated string contains Unicode characters which correspond to the ASCII interpretation of the bytes in the input string. This may be useful for debugging. Default: false</p>
output-format	<p>(Keyword) Unicode text format of the generated string: <i>utf8</i>, <i>ebcdicutf8</i> (on EBCDIC platforms), <i>utf16</i>, <i>utf16le</i>, <i>utf16be</i>, <i>utf32</i>. An empty string is equivalent to <i>utf16</i>. Default: <i>utf16</i></p> <p>The output format is forced to <i>utf16</i> in Unicode-capable language bindings, and to <i>utf8</i> if <i>stringformat=utf8</i>.</p> <p>IBM System i and IBM Z: if <i>stringformat=ebcdicutf8</i> the output format is forced to <i>ebcdicutf8</i>.</p> <p>RPG language binding: only the following output formats are allowed: <i>ebcdicutf8</i>, <i>utf8</i>, <i>utf16</i>, <i>utf32</i>.</p>



2 General Methods

API methods in this chapter:

- ▶ `PDF_set_option()`
- ▶ `PDF_get_option()`
- ▶ `PDF_get_string()`
- ▶ `PDF_new()`
- ▶ `PDF_delete()`
- ▶ `PDF_create_pvf()`
- ▶ `PDF_delete_pvf()`
- ▶ `PDF_info_pvf()`
- ▶ `PDF_download()`
- ▶ `PDF_poca_new()`
- ▶ `PDF_poca_delete()`
- ▶ `PDF_poca_insert()`
- ▶ `PDF_poca_remove()`

2.1 Global Options

PDFlib offers various global options for controlling the library and the appearance of the PDF output. These options retain their settings across the life span of the PDFlib object, or until they are explicitly changed by the client.

C++ Java C# `void set_option(String optlist)`

Perl PHP `set_option(string optlist)`

C `void PDF_set_option(PDF *p, const char *optlist)`

Set one or more global options.

optlist An option list specifying global options according to Table 2.1. The following options can be used:

- ▶ Options for resource handling and resource categories according to Table 2.1: *CMap, Encoding, enumeratefonts, FontnameAlias, FontOutline, HostFont, ICCProfile, resourcefile, saveresources, searchpath*
- ▶ Options for file handling and licensing according to Table 2.1: *avoiddemostamp, filenamehandling, license, licensefile*
- ▶ Text filter options according to Table 2.1: *charref, escapesequence, glyphcheck*
- ▶ Option for interactive elements according to Table 2.1: *usercoordinates*
- ▶ Other options according to Table 2.1: *asciifile, autospace, compress, kerning, logging, network, shutdownstrategy, usehostfonts, userlog*
- ▶ Option for error handling: *errorpolicy* (see Table 1.5)
- ▶ Options for color handling according to Table 2.1 and Table 8.2: *iccprofilecmyk, iccprofilegray, iccprofilergb*
- ▶ In the C language binding and in Perl, PHP, Ruby with *stringformat=legacy* the following encoding-related options according to Table 2.2 can also be used: *hypertextencoding, hypertextformat, stringformat, usehypertextencoding, textformat*

Details Except for resource category options new values override previously set option values.

The following options provide default values for the same-named text options (see Table 4.5 and Table 4.6):

charref, *escapesequence*, *glyphcheck*, *Kerning*, *textformat*

At the same time these options change the options of the same name in the current text state. It is recommended to set options for content strings only in `PDF_set_text_option()` to avoid unwanted side effects.

Scope any, but restricted scopes apply to some options

Table 2.1 Global options for `PDF_set_option()`

option	description
asciifile	(Boolean; only on IBM System i and IBM Z). Expect text files in ASCII encoding. Default: true on IBM System i; false on IBM Z
autospace	If true and the current font contains a glyph for U+0020, PDFlib will automatically add a space character after each text output. This may be useful for generating Tagged PDF. Note that adding spaces changes the current text position. Default: false
avoiddemo-stamp	(Boolean) If true, an exception is thrown when no valid license key was found; if false, a demo stamp is created on all pages. This option must be set before the first call to <code>PDF_begin_document()</code> . Default: false
charref	(Boolean) If true, enable substitution of numeric and character entity references and glyph name references for all content, name and hypertext strings. In order to avoid character reference substitution in places where it is not desired (e.g. file names) it is recommended to set this option for content strings only in <code>PDF_set_text_option()</code> ; see PDFlib Tutorial for details. Default: false
CMap	(List of pairs of name strings; deprecated) List of key/value pairs for a resource definition separated by whitespace or equal signs '='.
compress	(Integer) Compression level from 0=no compression, 1=best speed, etc. to 9=best compression. This option does not affect image data handled in passthrough mode. Default: 6. Scope: any except object
Encoding	(List of pairs of name strings) List of key/value pairs for a resource definition separated by whitespace or equal signs '=' (see PDFlib Tutorial for details). Multiple calls add new entries to the internal list.
enumerate-fonts	(Boolean) If true, PDFlib searches for font files in all directories which can be accessed via the SearchPath resource. Host fonts are not processed, but can be included by adding the system font directory to the search path, e.g. C:/Windows/Fonts. Font enumeration may take significant time for a large number of fonts. The resulting resource list can be saved to a file with the <code>saveresources</code> option. The recommended strategy is to create and save the resource list only if the set of fonts has changed, but not for each document or PDFlib object. For each font PDFlib determines the font-family, font-weight and font-style names and synthesizes an API font name according to the following scheme: <code><font-family>[, <font-weight>][, font-style]</code> PDFlib creates a <code>FontOutline</code> resource of the form <code><fontname>=<pathname></code> which connects the artificial font name with the full path name of the font. In addition to the API font name PDFlib creates a <code>FontnameAlias</code> resource with the PostScript name of the font if it is different from the artificial name: <code><PostScript fontname>=<artificial fontname></code> As a result, the font can be loaded by the artificial font name or the PostScript name. Default: false
escape-sequence	(Boolean) If true, enable substitution of escape sequences in all content, name and hypertext strings. In order to avoid escape sequence substitutions in places where it is not desired (e.g. file names) it is recommended to set this option for content strings only in <code>PDF_set_text_option()</code> ; Default: false

Table 2.1 Global options for `PDF_set_option()`

option	description
filename-handling	(Keyword) Indicates the encoding of file names. File names supplied as method parameters without UTF-8 BOM in non-Unicode aware language bindings are interpreted according to this option. Default: unicode on Windows and macOS, auto on IBM System i, otherwise honorLang: ascii 7-bit ASCII baseibcdic Basic EBCDIC according to code page 1047, but only Unicode values <= U+007E baseibcdic_37 Basic EBCDIC according to code page 0037, but only Unicode values <= U+007E honorlang (Not on IBM System i) The environment variables LC_ALL, LC_CTYPE and LANG are interpreted. The codeset specified in LANG is applied to file names if it is available. unicode Unicode encoding in (EBCDIC-) UTF-8 format all valid encoding names Any encoding except glyphid and builtin
Fontname- Alias	(List of pairs of name strings) List of key/value pairs for a resource definition separated by whitespace or equal signs '=' (see PDFlib Tutorial for details). Multiple calls add new entries to the internal list.
FontOutline	(List of pairs of name strings) List of key/value pairs for a resource definition separated by whitespace or equal signs '=' (see PDFlib Tutorial for details). Multiple calls add new entries to the internal list.
glyphcheck	(Keyword) See Table 4.5 for a description. It is recommended to set this option for content strings only in <code>PDF_set_text_option()</code> ; see PDFlib Tutorial for details. Default: replace
HostFont	(List of pairs of name strings) List of key/value pairs for a resource definition separated by whitespace or equal signs '=' (see PDFlib Tutorial for details). Multiple calls add new entries to the internal list.
ICCProfile	(List of pairs of name strings) List of key/value pairs for a resource definition separated by whitespace or equal signs '=' (see PDFlib Tutorial for details). Multiple calls add new entries to the internal list.
iccprofilecmk iccprofilegray iccprofilergb	(ICC profile handle) ICC profile which specifies a CMYK, Gray, or RGB color space for use with the icc-basedcmk/gray/rgb color options. Default: no ICC color space
kerning	(Boolean) If true, enable kerning for fonts which have been opened with the readkerning option; disable kerning otherwise. Default: true
license	(String) License key for PDFlib, PDFlib+PDI, or PPS (see PDFlib Tutorial for details). The key can be set before the first call to <code>PDF_begin_document()</code> . Use the avoiddemostamp option to make sure that missing license keys will not accidentally result in a demo stamp.
licensefile	(Name string) Name of a file containing the license key (see PDFlib Tutorial for details). The license file can only be set once before the first call to <code>PDF_begin_document()</code> .
logging	(Option list) Logging options according to Table 1.4
network	(Network option list) Option list according to Table 2.9 for controlling network access in API methods which implicitly download data (explicit downloading with <code>PDF_download()</code> is not affected). All network settings remain in effect until the next call to <code>PDF_set_option()</code> . Each call sets all network options from scratch (including defaults for options which are not specified), i.e. multiple calls don't accumulate options. The url network suboption is not allowed since the target network address is taken from context, e.g. referenced font or image resources in SVG graphics.
resourcefile	(Name string) Relative or absolute file name of the PDFlib UPR resource file. The resource file is loaded immediately before the first access to any resource. Existing resources will be kept; their values are overridden by new ones if they are set again.
saveresources	(Option list) Save the current resource list to a file. The following option is supported: filename The name of the resource file to which the resource list will be saved. Default: pdflib.upr

Table 2.1 Global options for `PDF_set_option()`

option	description
searchpath	(List of name strings) One or more relative or absolute path name(s) of directories containing files to be read. The search path can be set multiply; the entries are accumulated and used in least-recently-set order (see PDFlib Tutorial for details). It is recommended to use double braces for all entries to avoid problems with directory names containing space characters. An empty string list (i.e. <code>{}</code>) deletes all search path entries including the default entries. On Windows the search path can also be set via a registry entry. Default: platform-specific, see PDFlib Tutorial
shutdown-strategy	(Integer) Strategy for releasing global resources which are allocated once for all PDFlib objects. Each global resource is initialized on demand when it is first needed. This option must be set to the same value for all PDFlib objects in a process; otherwise the behavior is undefined (default: 0): <ul style="list-style-type: none"> 0 A reference counter keeps track of how many PDFlib objects use the global resources. When the last PDFlib object is deleted the resources are released. 1 The resources are kept until the end of the process. This may slightly improve performance, but requires more memory after the last PDFlib object is deleted.
user-coordinates	(Boolean) If false, coordinates for interactive elements (e.g. annotations) are expected in the default coordinate system; otherwise the current user coordinate system will be used. Default: false
userlog	String which will be copied to the log file
usehostfonts	(Boolean) If true, host fonts are included in the font search. Default: true

Table 2.2 Options for `PDF_set_option()` which apply only to C as well as to Perl, PHP, Ruby with `stringformat=legacy`

option	description
hypertext-encoding	(String) Encoding for hypertext strings. An empty string is equivalent to unicode. Default: auto
hypertext-format	(Keyword) Format for hypertext strings as method parameters. Supported keywords are bytes, utf8, ebcdicutf8, utf16, utf16le, utf16be, and auto. Default: auto
stringformat	(Keyword) Format of all strings at the API, i.e. name strings, content strings, hypertext strings and option lists. Supported keywords (default: legacy for C, but utf8 for Perl, PHP and Ruby): <ul style="list-style-type: none"> ebcdicutf8 (Only on IBM System i and IBM Z) All strings and option lists are expected in EBCDIC-UTF-8 format with or without BOM. legacy Name strings, content strings, hypertext strings and option lists are treated according to the textformat, hypertextformat and hypertextencoding options. utf8 (Not on IBM System i and IBM Z) All strings and option lists are expected in UTF-8 format with or without BOM. The options textformat, hypertextformat and hypertextencoding are not allowed. The Textflow option fixedtextformat is forced to true. In the C language binding name strings as method parameters are still interpreted as UTF-16 strings if the length parameter is supplied with a value larger than 0.
usehypertext-encoding	(Boolean) If true, the encoding specified in the hypertextencoding option will also be used for name strings. If false, the encoding for name strings without UTF-8 BOM is host. Default: false
textformat	(Keyword) Format used to interpret content strings. Supported keywords: bytes, utf8, ebcdicutf8 (only on IBM System i and IBM Z), utf16, utf16le, utf16be, and auto. Default: auto

C++ Java C# **double** *get_option*(String keyword, String optlist)

Perl PHP **float** *get_option*(string keyword, string optlist)

C **double** *PDF_get_option*(PDF *p, const char *keyword, const char *optlist)

Retrieve some option or other value.

keyword Keyword specifying the option to retrieve. The keywords below are supported; see description of *PDF_set_option()*, *PDF_set_text_option()* and *PDF_set_graphics_option()* regarding their meaning. Keywords for which no corresponding option exists are described in Table 2.3:

- ▶ Keywords for the string index of the *n*-th entry of the specified resource, where *n* corresponds to the *resourcenumber* option:
Encoding, FontnameAlias, FontOutline, HostFont, ICCProfile, searchpath
- ▶ Keywords for Boolean option values return 1 for *true* or 0 for *false*:
asciifile, autospace, avoiddmostamp, charref, decorationabove, escapesequence, fakebold, kerning, overline, pdi, spotcolorlookup, strikeout, tagged, underline, usercoordinates, usehostfonts
- ▶ Keywords for integer and float option values:
charspacing, compress, ctm_a, ctm_b, ctm_c, ctm_d, ctm_e, ctm_f, currentx, currenty, icccomponents, flatness, font, fontsize, horizscaling, iccprofilecmyk, iccprofilegray, iccprofilergb, italicangle, leading, linecap, linejoin, linewidth, major, minor, miterlimit, pageheight, pagewidth, revision, scope, strokewidth, textrendering, textrise, textx, texty, underline-position, underlinewidth, wordspacing
- ▶ Keywords returning a string index for an option value or -1 if the string value is not available:
cliprule, errorpolicy, filenamehandling, fillrule, glyphcheck, resourcefile, scope
- ▶ Keywords for querying the current structure element (only in Tagged PDF mode):
activeitemid, activeitemindex, activeitemisinline, activeitemkidcount, activeitemname, activeitemstandardname
- ▶ In the C language binding and in Perl, PHP, Ruby with *stringformat=legacy* the following encoding-related keywords can also be used:
hypertextencoding, hypertextformat, stringformat, usehypertextencoding, textformat

optlist Option list specifying an option according to Table 2.4.

Returns The value of some option as requested by *keyword*. If no value for the requested keyword is available, the method returns -1. If the requested keyword produces text, a string index is returned, and the corresponding string must be retrieved with *PDF_get_string()*.

Scope any, but restricted scopes apply to some keywords

Table 2.3 Additional keywords for *PDF_get_option()*

keyword	description
activeitemid	(Integer) Item id of the currently active structure item. This may be used with <i>PDF_activate_item()</i> or the parent suboption of <i>PDF_begin_item()</i> and the tag option. -1 is returned if no root element has been created yet. Scope: document, page
activeitem-index	(Integer) Zero-based index of the currently active structure item within its parent. This may be used with the index tag option. If the current item is a pseudo element or the root element or no root element has been created yet -1 is returned. Scope: document, page

Table 2.3 Additional keywords for `PDF_get_option()`

keyword	description
<code>activeitem-isinline</code>	(Integer) 1 if the currently active structure item is a direct element, 0 otherwise. Scope: document, page
<code>activeitem-kidcount</code>	(Integer) Number of child elements of the currently active structure element created up to this point (not counting pseudo elements). -1 is returned if no root element has been created yet. Scope: document, page
<code>activeitem-name</code>	String index for the type name of the currently active structure element or pseudo element, or -1 if no root element has been created yet. Scope: document, page
<code>activeitem-standard-name</code>	String index for the standard structure type name to which the currently active item is role mapped, or -1 if no root element has been created yet or the current item is a custom element for which no role mapping is available. If no role map is active the original type name is returned. Scope: document, page
<code>ctm_a</code> <code>ctm_b</code> <code>ctm_c</code> <code>ctm_d</code> <code>ctm_e</code> <code>ctm_f</code>	(Float) The components of the current transformation matrix (CTM) for vector graphics. Scope: page, pattern, template, glyph, path
<code>currentx</code> <code>currenty</code>	(Float) The x or y coordinate (in units of the current coordinate system), respectively, of the current point. Scope: page, pattern, template, glyph, path
<code>iccprofile</code>	(Integer) Number of color components in the ICC profile referenced by the handle provided in the <code>iccprofile</code> option
<code>major</code> <code>minor</code> <code>revision</code>	(Integer) Major, minor, or revision number of PDFlib, respectively. Scope: any, null ¹
<code>pageheight</code> <code>pagewidth</code>	(Float) Page size of the current page (dimensions of the MediaBox). Scope: any except object, but only reasonable in page scope
<code>pdi</code>	(Integer) Returns 1 if PDI has been included when building the underlying library. This is true for all combined PDFlib, PDFlib+PDI, and PPS binaries distributed by PDFlib GmbH, regardless of the license key. Otherwise it returns 0. Scope: any, null ¹
<code>scope</code>	(Integer) String index for the name of the current scope (see Table 1.3)
<code>textx</code> <code>texty</code>	(Float) The x or y coordinate of the current text position. Scope: page, pattern, template, glyph

1. C language binding: may be called with a PDF * argument of NULL or 0

Table 2.4 Options for `PDF_get_option()`

option	description
<code>textstate</code>	(Boolean) If true, the values of the following options are retrieved from the current text state, otherwise from the global options (default: false): <code>charref</code> , <code>escapesequence</code> , <code>glyphcheck</code> , <code>kerneling</code> , <code>textformat</code>
<code>iccprofile</code>	(ICC profile handle) ICC profile for use with the <code>iccprofile</code> keyword
<code>resource-number</code>	(Integer) Number of the resource to be retrieved; resources are numbered starting with 1. Default: 1

C++ Java C# **String** `get_string(int idx, String optlist)`

Perl PHP **string** `get_string(int idx, string optlist)`

C **const char ***`PDF_get_string(PDF *p, int idx, const char *optlist)`

Retrieve a string value.

idx String index returned by one of the `PDF_get_option()` or `PDF_info_*`() methods, or -1 if an option is supplied.

optlist An option list specifying options according to Table 2.5.

Returns The value of some string as requested by `idx` and `optlist`.

Scope any

Bindings C: The returned string is valid until the next call to any API method.

Table 2.5 Option for `PDF_get_string()`

option	description
version	(Boolean) Full PDFlib version string in the format <major>.<minor>.<revision>, possibly suffixed with additional qualifiers such as beta, rc, etc. Scope: any, null ¹

1. C language binding: may be called with a PDF * argument of NULL or 0

2.2 Creating and Deleting PDFlib Objects

C **PDF *PDF_new(void)**

Create a new PDFlib object.

Returns A handle to a PDFlib object which is to be used in subsequent PDFlib calls. If this method doesn't succeed due to unavailable memory it returns NULL.

Scope *null*; this method starts *object* scope, and must always be paired with a matching *PDF_delete()* call.

Bindings C: In order to load the PDFlib DLL dynamically at runtime use *PDF_new_dl()*. *PDF_new_dl()* returns a pointer to a *PDFlib_api* structure filled with pointers to all PDFlib API methods. If the DLL cannot be loaded, or a mismatch of major or minor version number is detected, NULL will be returned.

Other language bindings: this method is not available since it is hidden in the PDFlib constructor.

C **PDF *PDF_new2(void (*errorhandler)(PDF *p, const char *msg),
void* (*allocproc)(PDF *p, size_t size, const char *caller),
void* (*reallocproc)(PDF *p, void *mem, size_t size, const char *caller),
void (*freeproc)(PDF *p, void *mem),
void *opaque)**

Create a new PDFlib object with client-supplied error handling and memory allocation routines.

errorhandler Pointer to a user-supplied error-handling function. The error handler is ignored in *PDF_TRY/PDF_CATCH* sections.

allocproc Pointer to a user-supplied memory allocation function.

reallocproc Pointer to a user-supplied memory reallocation function.

freeproc Pointer to a user-supplied free function.

opaque Pointer to some user data which may be retrieved later with *PDF_get_opaque()*.

Returns A handle to a PDFlib object which is to be used in subsequent PDFlib calls. If this function doesn't succeed due to unavailable memory it will return NULL in C or throw an exception in C++.

Details This function creates a new PDFlib object with client-supplied error handling and memory allocation routines. Unlike *PDF_new()*, the caller may optionally supply own procedures for error handling and memory allocation. The function pointers for the error handler, the memory procedures, or both may be NULL. PDFlib will use default routines in these cases. Either all three memory routines must be provided, or none.

Scope *null*; this function starts *object* scope, and must always be paired with a matching *PDF_delete()* call.

Bindings C++: this function is indirectly available via the PDF constructor. Not all function arguments must be given since default values of NULL are supplied. All supplied methods must be »C« style functions, not C++ methods.

C *void PDF_delete(PDF *p)*

Delete a PDFlib object and free all internal resources.

Details This function deletes a PDFlib object and frees all document-related PDFlib-internal resources. This function must only be called once for a given PDFlib object. *PDF_delete()* should also be called for cleanup when an exception occurred. *PDF_delete()* itself is guaranteed to not throw any exception. If more than one PDF document will be generated it is not necessary to call *PDF_delete()* after each document, but only when the complete sequence of PDF documents is done.

Scope *any*; no more API method calls with the same PDFlib object are allowed with the PDF object after this call.

Bindings C: If the PDFlib DLL has been loaded dynamically at runtime with *PDF_new_dl()*, use *PDF_delete_dl()* to delete the PDFlib object.

C++: this method is indirectly available via the PDF destructor.

Java: this method is automatically called by the wrapper code. However, it can explicitly be called from client code in order to overcome shortcomings in Java's finalizer system.

Objective-C: this method is called when the PDFlib object's *release* method is called.

Perl and PHP: this method is automatically called when the PDFlib object goes out of scope.

2.3 PDFlib Virtual File System (PVF)

C++ `void create_pvf(string filename, const void *data, size_t size, string optlist)`

Java C# `void create_pvf(String filename, byte[] data, String optlist)`

Perl PHP `create_pvf(string filename, string data, string optlist)`

C `void PDF_create_pvf(PDF *p,
const char *filename, int len, const void *data, size_t size, const char *optlist)`

Create a named virtual read-only file from data provided in memory.

filename (Name string) The name of the virtual file. This is an arbitrary string which can later be used to refer to the virtual file in other PDFlib calls. The name of the virtual file will be subject to the *SearchPath* mechanism if it uses only slash '/' characters as directory or file name separators.

len (C language binding only) Length of *filename* (in bytes). If *len=0* a null-terminated string must be provided.

data A reference to the data for the virtual file. In C and C++ this is a pointer to a memory location. In Java this is a byte array. In Perl, Python, and PHP this is a string.

size (C, C++ and RPG only) The length in bytes of the memory area containing the data.

optlist An option list according to Table 2.6. The following option can be used: *copy*

Details The virtual file name can be supplied to any API method which uses input files. Use the *createpvf* option of `PDF_begin_document()` to create a PVF file which contains the generated PDF output. Some of these methods may set a lock on the virtual file until the data is no longer needed. Virtual files will be kept in memory until they are deleted explicitly with `PDF_delete_pvf()`, or automatically in `PDF_delete()`.

Each PDFlib object will maintain its own set of PVF files. Virtual files cannot be shared among different PDFlib objects, but they can be used for creating multiple documents with the same PDFlib object. Multiple threads working with separate PDFlib objects do not need to synchronize PVF use. If *filename* refers to an existing virtual file an exception will be thrown. This method does not check whether *filename* is already in use for a regular disk file.

Unless the *copy* option has been supplied, the caller must not modify or free (delete) the supplied data before a corresponding successful call to `PDF_delete_pvf()`. Not obeying to this rule will most likely result in a crash.

Scope any

Table 2.6 Option for `PDF_create_pvf()`

option	description
<code>copy</code>	(Boolean) If true, PDFlib, creates an internal copy of the supplied data. In this case the caller may dispose of the supplied data immediately after this call. Default: false for C and C++, but true for all other language bindings

C++ Java C# **int delete_pvf(String filename)**

Perl PHP **int delete_pvf(string filename)**

C **int PDF_delete_pvf(PDF *p, const char *filename, int len)**

Delete a named virtual file and free its data structures.

filename (Name string; will be interpreted according to the global *filenamehandling* option (see Table 2.1) The name of the virtual file as supplied to *PDF_create_pvf()*.

len (C language bindings only) Length of *filename* (in bytes). If *len=0* a null-terminated string must be provided.

Returns -1 (in PHP: 0) if the virtual file exists but is locked, and 1 otherwise. If *filename* does not refer to a valid virtual file 1 is returned.

Details If the file isn't locked, PDFlib deletes the data structures associated with *filename*. After successfully calling this method *filename* may be reused. All virtual files are automatically deleted in *PDF_delete()*.

The detailed semantics depend on whether or not the *copy* option has been supplied to the corresponding call to *PDF_create_pvf()*: If the *copy* option has been supplied, both the administrative data structures for the file and the actual file contents (data) are freed; otherwise, the contents are not freed since the client is supposed to do so.

Scope any

C++ Java C# **double info_pvf(String filename, String keyword)**

Perl PHP **float info_pvf(string filename, string keyword)**

C **double PDF_info_pvf(PDF *p, const char *filename, int len, const char *keyword)**

Query properties of a virtual file or the PDFlib Virtual File system (PVF).

filename (Name string) The name of the virtual file. The filename may be empty if *keyword=filecount*.

len (C language bindings only) Length of *filename* (in bytes). If *len=0* a null-terminated string must be provided.

keyword Keyword according to Table 2.6.

Table 2.7 Keywords for *PDF_info_pvf()*

keyword	description
filecount	Total number of files in the PDFlib Virtual File system maintained for the current PDFlib object. The filename parameter is ignored.
exists	1 if the file exists in the PDFlib Virtual File system (and has not been deleted), otherwise 0
size	(Only for existing virtual files) Size of the specified virtual file in bytes.
iscopy	(Only for existing virtual files) 1 if the copy option was supplied when the specified virtual file was created, otherwise 0
lockcount	(Only for existing virtual files) Number of locks for the specified virtual file set internally by PDFlib API methods. The file can only be deleted if the lock count is 0.

Returns The value of some file parameter as requested by *keyword*.

Details This method returns various properties of a virtual file or the PDFlib Virtual File system (PVF). The property is selected by *keyword*.

Scope any

2.4 Downloading Data from the Network

C++ Java C# *int download(string filename, string optlist)*

Perl PHP *int download(string filename, string optlist)*

C *int PDF_download(PDF *p, const char *filename, int len, const char *optlist)*

Download data from a network resource and store it in a disk-based or virtual file.

Note This method is not available on IBM System i.

filename (Name string) Name of a disk-based or virtual target file where the downloaded data is stored.

len (C language binding only) Length of *filename* (in bytes). If *len=0* a null-terminated string must be provided.

optlist An option list according to Table 2.8. The following options can be used: *createpvf*, *source*

Returns -1 (in PHP: 0) in case of an error, and 1 otherwise

Details This method downloads data from a network resource and stores it in a disk-based or virtual file (depending on the *createpvf* option). Caching functionality is available as follows: if *createpvf=true* and the same *filename* and URL as in an earlier call are supplied again, the existing data from the earlier call is used and the current call succeeds without any network activity. If caching is not desired (e.g. for dynamically generated resources which deliver different data for multiple accesses to the same URL), the PVF file must either be deleted with *PDF_delete_pvf()* prior to the current call or a different *filename* must be supplied. Caching is not available for disk-based files.

Details of network operation can be controlled with options. The following protocols are supported: *file*, *http*, *https*, and *ftp*.

Scope any.

Table 2.8 Options for *PDF_download()*

option	description
createpvf	(Boolean) If <i>true</i> , a PVF file containing the downloaded data is created. If a file with the specified name already exists in PVF and has been filled with data from the same URL in a previous call to <i>PDF_download()</i> the current call succeeds without downloading the data again (i.e. caching). Since PVF file contents cannot be accessed directly this option is only useful for supplying the downloaded data as input file to <i>PDFlib</i> . If this option is <i>false</i> a disk-based file is created and filled with the downloaded data. Default: <i>false</i>
source	(Network option list; required) Options according to Table 2.9 for controlling details of network operation.

Table 2.9 Suboptions for network option lists (options source in `PDF_download()` and network in `PDF_set_option()`)

option	description
disable	(Boolean; only for <code>PDF_set_option()</code>) If true, all network operations are suppressed. Default: false
httpauthen- tication	(Keyword; only for http) Authentication method(s) to try. A server may not support a particular authentication method (or any at all). Setting the authentication type explicitly may be preferable over the default (even if it results in the same method being selected) due to performance advantages. Supported keywords (default: any) : any Select the most secure authentication method supported by the server. anysafe Like any, but exclude basic authentication. basic Basic authentication with user name and password. This method is not recommended since user name and password are sent over the network in plain text. digest Digest authentication with hashed user name and password according to RFC 2617. This is more secure than basic authentication. ntlm NTLM authentication as used in Microsoft products
password	(String) Password for basic and digest authentication
proxy	(Option list) Suboptions for configuring network access through a proxy server: httpauthentication (Keyword; only for http proxy) See main network option of the same name. noproxy (String) Comma-separated list of host names that do not require a proxy. A numerical IPv6 address must be specified without enclosing brackets. password (String) See main network option of the same name. sslcertdir (String; only for https proxy) See main network option of the same name. sslcertfile (String; only for https proxy) See main network option of the same name. sslverifyhost (String; only for https proxy) See main network option of the same name. sslverifypeer (String; only for https proxy) See main network option of the same name. url (String; required) Host name or numerical IP address of the proxy server. The URL may include user name and password. A numerical IPv6 address must be enclosed in brackets [. . .]. A port number may be appended with a colon ':' at the end. If no port number is specified the default port number 1080 is used. If no protocol is specified http is used. username (String) See main network option of the same name. A proxy server can also be configured with the common environment variables <code>http_proxy</code> , <code>https_proxy</code> , <code>no_proxy</code> , <code>all_proxy</code> . Options have priority over environment variables.
sslcertdir	(String; only for https) Name of a directory which contains trusted CA certificates in PEM encoding which may be required for establishing an SSL connection. OpenSSL naming conventions (OpenSSL command <code>c_rehash</code>) apply to certificate file names.
sslcertfile	(String; only for https) Name of a file which contains one or more trusted CA certificates in PEM encoding which may be required for establishing an SSL connection.
sslverifyhost	(Boolean; only for https) If true, the Subject Alternate Name field in the server certificate must match the host name in the URL in order to establish a connection. Default: true
sslverifypeer	(Boolean; only for https) If true, the server certificate must be verifiable against the set of trusted certificates supplied with the <code>sslcertdir</code> or <code>sslcertfile</code> options. If false, a server certificate which cannot be verified because no known trusted root is available for it is accepted. Default: false
timeout	(Integer) Timeout for accessing the resource in milliseconds. The value 0 means that no timeout is effective. Default: 5000
username	(String) User name for basic and digest authentication
url	(String; only for <code>PDF_download()</code> and required in this case) Fully qualified URL of the network resource which must include a protocol identifier, e.g. <code>http://</code> . Characters can be specified in URL encoding, e.g. <code>%20</code> . The URL may include user name and password in standard syntax.

2.5 PDF Object Creation API (POCA)

Object types and frozen objects. The PDF object creation API (POCA) is a low-level interface for creating PDF objects. POCA supports the following object types:

- ▶ simple object types: boolean, integer, name, float, string;
- ▶ container object types: array, dictionary;
- ▶ specific types for PDFlib Blocks: percentage, color.

POCA handles representing PDF objects can be used as follows:

- ▶ with the *dpm* option of `PDF_begin/end_dpart()` to create document part metadata;
- ▶ with the *blocks* option of `PDF_begin/end_page_ext()` to create PDFlib Blocks for use with PPS;
- ▶ with the *richmediaargs* option of `PDF_create_action()` to specify arguments for the JavaScript associated with a rich media annotation.

Supplying a PDF container object to any of the options listed above freezes the container object itself as well as all objects referenced from the container directly or indirectly, i.e. the full object tree created by the container is frozen. Frozen objects can be used again with the options above, but they can no longer be modified with `PDF_poca_insert()` or `PDF_poca_remove()`.

C++ Java C# `int poca_new(String optlist)`

Perl PHP `int poca_new(string optlist)`

C `int PDF_poca_new(PDF *p, const char *optlist)`

Create a new PDF container object of type dictionary or array and insert objects.

optlist An option list for creating and populating a container.

- ▶ Options for creating a container according to Table 2.10: *containertype, usage*
- ▶ Options for inserting objects in the container according to Table 2.12: *direct, hypertextencoding, index, key, type, value, values*

Returns A POCA container handle which can be used until it is deleted with `PDF_poca_delete()`.

Details This method creates an empty PDF container object of the specified container type. The container can immediately be populated in the same call or later calls to `PDF_poca_insert()`.

PDF/VT A POCA container handle for an object of type dictionary with *usage=dpm* can be supplied as Document Part Metadata (DPM) with the *dpm* option of `PDF_begin/end_dpart()`.

Scope any

Table 2.10 Options for `PDF_poca_new()`

option	description
container-type	(Keyword; required) Type of the container: dict or array. Unspecified array slots and array slots which have been removed without inserting a new object contain the keyword null in the PDF output.

Table 2.10 Options for `PDF_poca_new()`

option	description
usage	(Keyword; required) Context in which the new container will be used. This option enables some checks to make sure that the container is suited for the intended use:
blocks	(Only relevant for <code>containertype=dict</code> ; only in the PPS product) The Block dictionary (the container which will be supplied to the <code>blocks</code> option of <code>PDF_begin/end_page_ext()</code>) must contain one or more PDFlib Block definitions. The option <code>usage=blocks</code> must also be supplied to all container objects which will directly or indirectly be inserted into the new dictionary.
dpm	(Only relevant for <code>containertype=dict</code>) All keys in the new dictionary or any dictionary contained in it must consist of ASCII characters, must conform to the rules of an XML NMTOKEN. This ensures that the dictionary can be used as Document Part Metadata (DPM) dictionary. The option <code>usage=dpm</code> must also be supplied to all container objects which will directly or indirectly be inserted into the new dictionary.
richmediaargs	(Only for <code>containertype=array</code>) The array can contain objects of type string, integer, float, or Boolean.

C++ Java C# **void** `poca_delete(int container, String optlist)`

Perl PHP `poca_delete(int container, string optlist)`

C **void** `PDF_poca_delete(PDF *p, int container, const char *optlist)`

Delete a PDF container object.

container A valid POCA container handle retrieved with `PDF_poca_new()`.

optlist An option list according to Table 2.11. The following option can be used:
recursive

Details The container is deleted and can no longer be used. If the container is referenced from another dictionary or array all dictionary references to the deleted container are removed, and all array references to the deleted container are replaced with the *null* object. POCA container objects are not automatically deleted in `PDF_end_document()`.

Scope any; must always be paired with a matching `PDF_poca_new()` call.

Table 2.11 Options for `PDF_poca_delete()`

option	description
recursive	(Boolean) If <code>true</code> , the container object itself and all objects referenced from it are deleted recursively. This may be useful as a shortcut for deleting a full object tree. Default: <code>false</code>

C++ Java C# **void poca_insert(int container, String optlist)**

Perl PHP **poca_insert(int container, string optlist)**

C **void PDF_poca_insert(PDF *p, int container, const char *optlist)**

Insert a simple or container object in a PDF container object.

container A valid POCA container handle retrieved with `PDF_poca_new()`. Frozen containers (see »Object types and frozen objects«, page 39) are not allowed since they can no longer be modified.

optlist An option list according to Table 2.12. The following options can be used: *direct*, *hypertextencoding*, *index*, *key*, *type*, *value*, *values*

Details This method inserts an object in a container. The order in which objects are inserted in a container is not significant. Inserted containers may be populated after insertion; it is not required that inserted containers be complete at the time of insertion.

Inserting an object into a container must not create a loop of direct objects within the object hierarchy. For example, a directly inserted dictionary cannot include a direct reference to its container. In order to create cyclic references use *direct=false* to create indirect objects which can reference arbitrary other objects.

Scope any

Table 2.12 Options for `PDF_poca_new()`, `PDF_poca_insert()` and `PDF_poca_remove()`

option	description
direct ¹	(Boolean; only for type=array and dict; ignored for other types) If true, the object will be inserted directly in the container; if false, an indirect PDF object will be created and a reference to the indirect PDF object will be inserted in the container. Indirect objects are useful to save space in the generated PDF if an object is used more than once. Default: true
hypertext-encoding	(Keyword) Encoding for the key, value, and values options. An empty string is equivalent to unicode. Default: value of the global hypertextencoding option
index	(Integer; only for containers with type=array; required for <code>PDF_poca_remove()</code>) The zero-based index at which the value(s) will be inserted or deleted in the array. The value -1 can be used to insert the element as the new last item. The array will grow as necessary to include an element with the specified index. If the array already contains a value at the specified index it will be replaced with the new value. Default for <code>PDF_poca_new()</code> and <code>PDF_poca_insert()</code> : -1
key	(Hypertext string; only for containers with type=dict and required in this case) The key under which the value is inserted in the dictionary container. The key must not include the leading '/' slash character. The key must conform to the conditions specified in the dictionary's usage option. If the dictionary already contains an entry with the same key it is replaced with the new value.
type ¹	(Keyword; required) Type of the inserted object: array, boolean, dict, integer, name, float, string, percentage, color The following types are not allowed if the container has been created with usage=dpm: name (use type=string instead) The following types are only allowed if the container has been created with usage=blocks: color, percentage

Table 2.12 Options for `PDF_poca_new()`, `PDF_poca_insert()` and `PDF_poca_remove()`

option	description
value ¹	<p>(Data type according to the <code>type</code> option; exactly one of the options <code>value</code> and <code>values</code> must be provided) The value of the inserted object, subject to the container type and the <code>type</code> option:</p> <p>For array and dictionary containers:</p> <p>If <code>type=boolean</code> the value must have option <code>type</code> string, and must contain one of the strings <code>true</code> or <code>false</code>.</p> <p>If <code>type=string</code> or <code>name</code> the value must have option <code>type</code> <code>Hypertext</code> string, and must contain the target directly. Values for <code>type=name</code> are limited to 127 bytes in UTF-8 representation, and must not include the leading <code>'/'</code> slash character.</p> <p>If <code>type=integer</code> the value must have option <code>type</code> integer and must contain the target directly.</p> <p>If <code>type=float</code> the value must have option <code>type</code> <code>float</code> or <code>integer</code> and must contain the target directly.</p> <p>If <code>type=array</code> or <code>dict</code> the value must have option <code>type</code> POCA container handle (i.e. created with <code>PDF_poca_new()</code>) and must specify the inserted container. The inserted object must have been created with the same usage option as the container.</p> <p>For <code>type=percentage</code> the value must have option <code>type</code> number. It will be interpreted as a percentage value and must include the percent sign (e.g. 50%). It will be written as Block data type percentage</p> <p>For <code>type=color</code> the value must have option <code>type</code> <code>color</code> (see Table 1.2, page 14). It will be written as Block data type <code>color</code>. The following color space keywords are not allowed: <code>iccbased</code>, <code>iccbasedgray</code>, <code>iccbasedrgb</code>, <code>iccbasedcmyk</code>, <code>pattern</code>, <code>devicen</code></p> <p>In order to pass arbitrary strings with this option the option list syntax described in »Unquoted string values«, page 10, may be useful.</p>
values ¹	<p>(List of one or more values according to the <code>type</code> option; only for containers with <code>type=array</code>; exactly one of the options <code>value</code> and <code>values</code> must be provided) One or more values of the same type which is inserted in the array at the position specified by the <code>index</code> option. See option <code>value</code> regarding the conditions for specific types. If the specified list contains only a single element, the effect is equivalent to the <code>value</code> option. If the list contains more than one element, all elements in the list are inserted in the array sequentially, possibly overriding existing elements. The array will grow as necessary to include all elements in the specified list.</p>

1. Only for `PDF_poca_new()` and `PDF_poca_insert()`

C++ Java C# **`void poca_remove(int container, String optlist)`**

Perl PHP **`poca_remove(int container, string optlist)`**

C **`void PDF_poca_remove(PDF *p, int container, const char *optlist)`**

Remove a simple or container object from a PDF container object.

container A valid POCA dictionary or array handle retrieved with `PDF_poca_new()`. Frozen containers (see »Object types and frozen objects«, page 39) are not allowed since they can no longer be modified.

optlist The following options of `PDF_poca_insert()` in Table 2.12 can be used: `hypertextencoding`, `index`, `key`

Details This method removes an object from a container of type array or dictionary. Nothing happens if the specified object doesn't exist in the container.

Scope any

3 Document and Page Methods

API methods in this chapter:

- ▶ `PDF_begin_document()`
- ▶ `PDF_begin_document_callback()`
- ▶ `PDF_end_document()`
- ▶ `PDF_get_buffer()`
- ▶ `PDF_begin_page_ext()`
- ▶ `PDF_end_page_ext()`
- ▶ `PDF_suspend_page()`
- ▶ `PDF_resume_page()`
- ▶ `PDF_define_layer()`
- ▶ `PDF_set_layer_dependency()`
- ▶ `PDF_begin_layer()`
- ▶ `PDF_end_layer()`

3.1 Document Methods

C++ Java C# `int begin_document(String filename, String optlist)`

Perl PHP `int begin_document(string filename, string optlist)`

C `int PDF_begin_document(PDF *p, const char *filename, int len, const char *optlist)`

C++ `void begin_document_callback(size_t (*writeproc) (PDF *p, void *data, size_t size), string optlist)`

C `void PDF_begin_document_callback(PDF *p, size_t (*writeproc) (PDF *p, void *data, size_t size), const char *optlist)`

Create a new PDF document subject to various options.

filename (Name string; will be interpreted according to the global *filenamehandling* option, see Table 2.1) Absolute or relative name of the PDF output file to be generated. If *filename* is empty, the PDF document will be generated in memory instead of on file, and the generated PDF data must be fetched by the client with the `PDF_get_buffer()` method. On Windows it is OK to use UNC paths or mapped network drives.

len (C language binding only) Length of *filename* (in bytes). If *len=0* a null-terminated string must be provided.

writeproc (Only for the C and C++ language bindings) C callback function which will be called by PDFlib in order to submit (portions of) the generated PDF data. The supplied *writeproc* must be a C-style function, not a C++ method.

optlist An option list specifying document options:

- ▶ General options: *errorpolicy* (see Table 1.5) and *hypertextencoding* (see Table 2.1)
- ▶ Document options according to Table 3.1. Some of these options can also be specified in `PDF_end_document()`; in this case they have precedence over identical options specified in `PDF_begin_document()`:

attachments, autoxmp, destination, emitdocinfo, groups, labels, linearize, metadata, object-streams, openmode, optimize, pagelayout, portfolio, search², uri, viewerpreferences

- ▶ Options for PDF compatibility and standards according to Table 3.2:
compatibility, limitcheck, nodenamelist, pdfa, pdfua, pdfvt, pdfx, recordlevel, uses-transparency
- ▶ Options for Tagged PDF according to Table 3.3:
checktags, lang, pdftagset¹, rolemap, structureassociatedfiles, structuretype¹, tag¹, tagged, tagsets¹
- ▶ Security options according to Table 3.4:
attachmentpassword, masterpassword, permissions, userpassword
- ▶ Output processing options according to Table 3.5:
createoutput, createpvf, filemode, flush, inmemory, recordsize, removefragments, tempdirname, tempfilenames

Returns -1 (in PHP: 0) on error, and 1 otherwise. If *filename* is empty this method will always succeed, and never return the error value.

Details This method creates a new PDF file using the supplied *filename*. PDFlib will attempt to open a file with the given name, and close the file when the PDF document is finished. *PDF_begin_document_callback()* opens a new PDF document, but instead of writing to a disk file it calls a client-supplied callback method to deliver the PDF output data. The function supplied as *writeproc* must return the number of bytes written. If the return value doesn't match the *size* argument supplied by PDFlib, an exception will be thrown. The frequency of *writeproc* calls is configurable with the *flush* option. The callback function must not call any API method or macro with the same context pointer except *PDF_get_opaque()*.

PDF/VT The following option is not allowed: *groups*. Setting *usestransparency=false* for documents which do not contain any transparent objects may result in faster ripping due to Form XObject encapsulation without transparency groups.

Scope *object*; this method starts *document* scope if the file could successfully be opened, and must always be paired with a matching *PDF_end_document()* call.

Bindings ASP: the *MapPath* facility should be used to construct full path names to be passed to this method.

C, C++, Java, JScript: take care of properly escaping the backslash path separator. For example, the following denotes a file on a network drive: `\\\\malik\\rp\\foo.pdf`. *PDF_begin_document_callback()* is only available in C and C++.

C++ Java C# **void end_document(String optlist)**

Perl PHP **end_document(string optlist)**

C **void PDF_end_document(PDF *p, const char *optlist)**

Close the generated PDF document and apply various options.

optlist An option list specifying document processing options:

- ▶ General option: *hypertextencoding* (see Table 2.1)
- ▶ Document options according to Table 3.1. Options specified in *PDF_end_document()* have precedence over identical options specified in *PDF_begin_document()*. The following options can be used:

action, associatedfiles, attachments, autoxmp, destination, destname, labels, metadata, openmode, pagelayout, portfolio, uri, viewerpreferences

Details This method finishes the generated PDF document, frees all document-related resources, and closes the output file if the PDF document has been opened with `PDF_begin_document()`. This method must be called when the client is done generating pages, regardless of the method used to open the PDF document.

When the document was generated in memory (as opposed to on file), the document buffer will still be kept after this method is called (so that it can be fetched with `PDF_get_buffer()`), and will be freed in the next call to `PDF_begin_document()`, or when the PDFlib object goes out of scope.

Scope *document*; this method terminates *document* scope, and must always be paired with a matching call to one of `PDF_begin_document()` or `PDF_begin_document_callback()`.

Table 3.1 Document options for `PDF_begin_document()` and `PDF_end_document()`

option	description
action¹	(Action list) List of document actions for one or more of the following trigger events (default: empty list): open Actions to be performed when the document is opened. Due to the execution order in Acrobat document-level JavaScript must not be used for open actions. didprint/didsave/willclose/willprint/willsave (not for PDF/A) JavaScript actions to be performed after printing/after saving/before closing/before printing/before saving the document. Individual actions supplied to the action option override actions imported from PDF documents with the useactions option of <code>PDF_open_pdi_document()</code> . Document-level JavaScript can be created with the script and scriptname options of <code>PDF_create_action()</code> .
associated-files¹	(List of asset handles; only for PDF 2.0 and PDF/A-3) Asset handles for files associated with the whole document. The files must have been loaded with <code>PDF_load_asset()</code> and type=attachment.
attachments	(List of option lists or list of asset handles; not for PDF/X-3 and PDF/A-1; PDF/A-2: only PDF/A-1 and PDF/A-2 documents can be attached; PDF/A-3: not allowed, use associatedfiles instead) Document-level file attachments which have been loaded with <code>PDF_load_asset()</code> and type=attachment. It is OK to supply file attachments both in <code>PDF_begin_document()</code> and <code>PDF_end_document()</code> . However, asset handles can only be supplied in <code>PDF_end_document()</code> . Supported suboptions: see Table 13.4
autoxmp	(Boolean; forced to true for PDF/X-3/4/5, PDF/A and PDF 2.0) If true, PDFlib creates XMP document metadata from document info fields (see Section 15.2, »XMP Metadata«, page 277). Default: true
destination	(Option list; ignored if an open action has been specified) An option list specifying the document open action according to Table 12.11.
destname¹	(Hypertext string; ignored if the destination option has been specified) The name of a destination which has been defined with <code>PDF_add_nameddest()</code> and will be used as the document open action.
emitdocinfo	(Boolean; only relevant for PDF 2.0) If true, a document information dictionary is emitted. Default: false
groups²	(List of strings; not allowed in PDF/VT mode or if a document part hierarchy is created) Define the names and ordering of the page groups used in the document. Page groups keep pages together (useful e.g. for attaching page labels); pages can be assigned to one of the page groups defined in the document, and referenced within the respective group. If page groups are defined for a document, all pages must be assigned to a page group.
labels	(List of option lists) A list containing one or more option lists according to Table 3.6 specifying symbolic page names. The page name will be displayed as a page label (instead of the page number) in Acrobat's status line. The combination of style/prefix/start must be unique within a document. Default: no page labels

Table 3.1 Document options for `PDF_begin_document()` and `PDF_end_document()`

option	description
linearize²	(Boolean; forces <code>objectstreams=none</code>) If <code>true</code> , the output document will be linearized. On z/OS this option cannot be combined with an empty filename. Default: <code>false</code>
metadata	(Option list) Supply XMP document metadata (see Section 15.2, »XMP Metadata«, page 277). Individual XMP properties may be overridden by document info fields supplied with <code>PDF_set_info()</code> . In PDF/A mode the supplied XMP metadata must conform to additional requirements (see PDFlib Tutorial).
objectstreams²	(List of keywords; PDF 1.5; forced to <code>none</code> if <code>linearize</code> is <code>true</code>) Generate compressed object streams which significantly reduce output file size (default: <code>{other nodocinfo}</code>): bookmarks Compress bookmark objects. docinfo Compress document info fields. dpartarrays Compress dictionaries related to the document part hierarchy. dpartdicts Compress arrays related to the document part hierarchy. fields Compress form fields. names Compress objects for named destinations. none Don't generate any compressed object streams (except for categories which are explicitly enabled after this option). other All categories which are not explicitly disabled after this keyword, plus other object types which don't have their own keyword. pages Compress the objects comprising the page tree. poca Compress all simple objects created with the POCA interface. tags Compress marked content tags. xref Generate a compressed xref stream. This category is automatically enabled if at least one of the other categories is enabled. <p>Except for <code>none</code> and <code>other</code>, all keywords can be prefixed with <code>no</code> (e.g. <code>nodocinfo</code>) to disable compression for the specified category. If at least one such negative keyword is supplied, the keyword <code>other</code> will be prepended to the list.</p>
openmode	(Keyword) Set the appearance when the document is opened. Default: <code>bookmarks</code> if the document contains any bookmarks, otherwise <code>none</code> . none Open with no additional panel visible. bookmarks Open with the bookmark panel visible. thumbnails Open with the thumbnail panel visible. fullscreen Open in fullscreen mode (does not work in the browser). layers (PDF 1.5) Open with the layer panel visible. attachments (PDF 1.6) Open with the attachments panel visible.
optimize²	(Boolean) If <code>true</code> , the output document will be optimized in a separate pass after generating it. Optimization reduces file size by eliminating redundant duplicate objects. In general optimization will not have any significant effect except for inefficient client code (e.g. loading the same image or ICC profile multiply instead of reusing the handle). On z/OS this option cannot be combined with <code>in-core</code> generation (i.e. an empty filename). Default: <code>false</code>
pagelayout	(Keyword) The page layout to be used when the document is opened (default: <code>default</code>): default The default setting of the PDF viewer. singlepage Display one page at a time. onecolumn Display the pages continuously in one column. twocolumnleft Display the pages in two columns, odd pages on the left. twocolumnright Display the pages in two columns, odd pages on the right twopageleft (PDF 1.5) Display the pages two at a time, with odd-numbered pages on the left. twopageright (PDF 1.5) Display the pages two at a time, with odd-numbered pages on the right.

Table 3.1 Document options for `PDF_begin_document()` and `PDF_end_document()`

option	description
portfolio ¹	(Option list; PDF 1.7) Suboptions for creating a PDF portfolio according to Table 15.5
search ²	(Option list; deprecated and not available in PDF 2.0) Instruct Acrobat to attach a search index when opening the document: filename (Hypertext string; required) Name of a file containing a search index indextype (Name string) Type of the index; must be PDX for Acrobat. Default: PDX
uri	(String) Set the document's base URL. This is useful when a document with relative Web links is moved to a different location. Adjusting the base URL makes sure that relative links will still work. Default: no base URI
viewer-preferences	(Option list) Option list specifying various viewer preferences according to Table 3.7. Default: empty

1. Only for `PDF_end_document()`

2. Only for `PDF_begin_document()` and `PDF_begin_document_callback()`

Table 3.2 Options for PDF compatibility and standards in `PDF_begin_document()`

option	description
compatibility	(Keyword; ignored if one of the <code>pdfa</code> , <code>pdfua</code> , <code>pdfvt</code> or <code>pdfx</code> options is used with a value different from none) Set the document's PDF version to one of the keywords below. This option affects which PDF features are available and which PDF documents can be imported with PDFlib+PDI (default: 1.7ext8): <ul style="list-style-type: none"> 1.4 PDF 1.4 requires Acrobat 5 or above. 1.5 PDF 1.5 requires Acrobat 6 or above. 1.6 PDF 1.6 requires Acrobat 7 or above. 1.7 PDF 1.7 is specified in ISO 32000-1 and requires Acrobat 8 or above. 1.7ext3 PDF 1.7 extension level 3 requires Acrobat 9 or above. 1.7ext8 PDF 1.7 extension level 8 requires Acrobat X or above. 2.0 PDF 2.0 is specified in ISO 32000-2.
limitcheck	If true, the limit for the number of indirect PDF objects (8,388,607) is enforced in PDF/A-1/2/3 and PDF/X-4/5 modes. Default: true
nodenamelist	(List of name strings; required for <code>pdfvt=PDF/VT-1</code>) Names for all levels of the document part hierarchy. All names must consist of ASCII characters and must conform to the rules of an XML NMTOKEN. The first string specifies the name for level 0 in the document part hierarchy.
pdfa	(Keyword) Set the PDF/A conformance level to one of the following (default: none): <ul style="list-style-type: none"> PDF/A-1a:2005, PDF/A-1b:2005 Implies <code>compatibility=1.4</code> PDF/A-2a, PDF/A-2b, PDF/A-2u Implies <code>compatibility=1.7</code> PDF/A-3a, PDF/A-3b, PDF/A-3u Implies <code>compatibility=1.7</code> none No PDF/A output PDF/A1-a:2005, PDF/A-2a, and PDF/A-3a imply <code>tagged=true</code> . PDF/A-1/2/3 are compatible with the following standards: <code>pdfx=PDF/X-3:2003, PDF/X-4</code> <code>pdfvt=PDF/VT-1</code> <code>pdfua=PDF/UA-1</code> If multiple options for PDF standards are specified the lowest compatibility value is used.
pdfua	(Keyword) Set the PDF/UA conformance level to one of the following (default: none): <ul style="list-style-type: none"> PDF/UA-1 Implies <code>compatibility=1.7</code> and <code>tagged=true</code>. none No PDF/UA output
pdfvt	(Keyword) Set the PDF/VT conformance level to one of the following (default: none): <ul style="list-style-type: none"> PDF/VT-1 Implies <code>pdfx=PDF/X-4</code>; any other value for the <code>pdfx</code> option is an error. none No PDF/VT output
pdfx	(Keyword) Set the PDF/X conformance level to one of the following (default: none): <ul style="list-style-type: none"> PDF/X-1a:2003 Implies <code>compatibility=1.4</code>; deprecated PDF/X-3:2003 Implies <code>compatibility=1.4</code>; deprecated PDF/X-4, PDF/X-4p Implies <code>compatibility=1.6</code> PDF/X-5n Implies <code>compatibility=1.6</code> none No PDF/X output
recordlevel	(Non-negative integer; only relevant if a document part hierarchy is created) Zero-based level of the document part hierarchy which corresponds to recipient records.
uses-transparency	(Boolean; not for PDF/A-1 and PDF/X-3) If false, none of the pages in the generated document will contain any transparent objects. PDFlib throws an exception if this assertion is violated. Setting this option to false is allowed only for documents without transparency. It implies that no transparency groups are created automatically for imported PDF pages and SVG graphics. Also, encapsulated Form XObjects for PDF/VT are created without a transparency group which allows more efficient RIP caching. Default: true

Table 3.3 Options for Tagged PDF in `PDF_begin_document()` and `PDF_end_document()`

option	description
checktags¹	<p>(Keyword; must be strict in PDF/UA-1 mode) Specifies whether the structure element nesting rules (see PDFlib Tutorial) are checked for tags. This option also controls the use of tags which are deprecated in PDF 2.0 with pdftagset=2.0. It is only provided as a migration aid. This option does not affect tags in imported pages (see option checktags of <code>PDF_open_pdi_document()</code>). Supported keywords (default: strict):</p> <p>none Tag nesting rules are not enforced. This setting may result in an invalid structure hierarchy and is therefore not recommended.</p> <p>relaxed Similar to strict except that a few PDF 1.7 rules are not enforced (see PDFlib Tutorial).</p> <p>strict If a tag violates the nesting rules an exception will be thrown. In PDF 2.0 mode deprecated tags also trigger an exception.</p>
lang	<p>(String; required for PDF/UA-1) Set the default language of the document as BCP 47 language tag consisting of a two- or three-character language code according to ISO 639-1/2 (e.g. de, en, fr, ja, fil), optionally followed by a hyphen and a two-character ISO 3166 region code (e.g. en-us, en-gb, es-mx). The value zxx indicates non-linguistic content, e.g. program code. An empty string means that the language is unknown (this is not allowed in PDF/UA mode).</p> <p>The default language can be overridden for individual structure elements with the lang tagging option. The lang option is allowed in <code>PDF_end_document()</code> only if it has also been supplied to <code>PDF_begin_document()</code>, i.e. only to change a previously set value. This may be useful e.g. if the document language is chosen based on an imported PDF document.</p>
pdftagset¹	<p>(Keyword; PDF 2.0) The tagset for standard PDF element type. All standard PDF type used in the document must be part of the selected tagset, and the corresponding nesting rules apply (default: 2.0 if compatibility=2.0, otherwise 1.7):</p> <p>2.0 PDF 2.0 tagset and nesting rules</p> <p>1.7 PDF 1.7 tagset and nesting rules</p>
rolemap¹	<p>(List of string lists; the first element in each string list is a name string, the second element is a string; only for Tagged PDF; required if custom structure types are used) Mapping of custom structure types to standard types. Each sublist contains the name of a standard or custom type, and the name of the standard type to which the first type will be mapped. Inline and pseudo types are not allowed for the second entry in a sublist. Standard type names also can be mapped to other standard types in order to assign different semantics to existing types. Indirect mappings are allowed, i.e. a custom type is mapped to another custom type which is then mapped to a standard type. Pairs with identical entries are silently ignored. In PDF/UA-1 it is not allowed to remap standard types.</p>
structure-associated-files²	<p>(List of asset handles; PDF 2.0) Asset handle(s) for file(s) associated with the whole structure tree. The file(s) must have been loaded with <code>PDF_load_asset()</code> and type=attachment.</p>
structuretype¹	<p>(Keyword; only for PDF/UA-1) Type of document structure (default: weak):</p> <p>strong The document is strongly structured, i.e. the structure tree reflects the document's logical organization. The only allowed structure type for headings is H while H1, H2, etc. are not allowed. Each node in the structure tree contains at most one H tag plus one or more paragraph tags P.</p> <p>weak The document is weakly structured, i.e. the structure tree is only a few levels deep with all headings, paragraphs etc. as immediate children. Logical structure may be expressed with heading tags H1, H2, etc. while H is not allowed. Headings may not have any descendants.</p>
tag¹	<p>(Option list) Tagging options according to Table 14.2. The specified structure element comprises the document structure root and is closed automatically in <code>PDF_end_document()</code>. Only grouping elements are allowed for the tagname suboption.</p>

Table 3.3 Options for Tagged PDF in `PDF_begin_document()` and `PDF_end_document()`

option	description
tagged¹	(Boolean) If true, generate Tagged PDF output. Proper structure information must be provided by the client in Tagged PDF mode. If PDF/A-1a:2005, PDF/A-2a, PDF/A-3a or PDF/UA-1 mode is active this option is automatically set to true. Default: false
tagsets¹	(Option list; PDF 2.0) List of custom tagsets which may be used in the document. The PDF 1.7 and PDF 2.0 don't require entries in the tagsets option:
namespace	(String; required) Name for identifying the namespace in the form of a uniform resource identifier (URI)
rolemap	(List of string lists; required) Role map for all structure types in the tagset. The role map contains a pair for each tag in the tagset, where each pair consists of the custom tag from the named tagset and the target tag from the set selected with the <code>pdftagset</code> option.

1. Only for `PDF_begin_document()` and `PDF_begin_document_callback()`

2. Only for `PDF_end_document()`

Table 3.4 Security options for `PDF_begin_document()`; not allowed for PDF/A and PDF/X

option	description
attachment-password¹	(String ² ; PDF 1.6; ignored if <code>userpassword</code> or <code>masterpassword</code> are set; can not be combined with the <code>linearize</code> and <code>optimize</code> options; not for PDF/A and PDF/X) File attachments are encrypted with the supplied string as password. The rest of the document is not encrypted. On EBCDIC platforms the password is expected in ebcdic encoding or EBCDIC-UTF-8.
master-password¹	(String; PDF 1.6; required if <code>permissions</code> has been specified; not for PDF/A and PDF/X; encryption is deprecated for PDF 1.6, 1.7 and 1.7ext3) The master password for the document. If it is empty no master password is applied. On EBCDIC platforms the password is expected in ebcdic encoding or EBCDIC-UTF-8. Default: empty
permissions	(Keyword list; not for PDF/A and PDF/X; requires <code>masterpassword</code>) List of access restrictions for the output document. It contains any number of the following keywords (default: empty):
noprint	Acrobat will prevent printing the file.
nohighresprint	Acrobat will prevent high-resolution printing. If <code>noprint</code> isn't set, printing is restricted to the »print as image« feature which prints a low-resolution rendition of the page.
nomodify	Acrobat will prevent editing or cropping pages and creating or changing form fields.
noassemble	(Implies <code>nomodify</code>) Acrobat will prevent inserting, deleting, or rotating pages and creating bookmarks and thumbnails.
noannots	Acrobat will prevent creating or changing annotations and form fields.
noforms	(implies <code>nomodify</code> and <code>noannots</code>) Acrobat will prevent form field filling.
nocopy	Acrobat will prevent copying and extracting text or graphics; the accessibility interface will be controlled by <code>noaccessible</code> .
noaccessible	(Deprecated and not available in PDF 2.0; not allowed in PDF/UA-1) Acrobat will prevent extracting text or graphics for accessibility (e.g. a screenreader).
plainmetadata	Keep XMP document metadata unencrypted even in an encrypted document.
user-password¹	(String; PDF 1.6; not for PDF/A and PDF/X; encryption is deprecated for PDF 1.6, 1.7 and 1.7ext3) The user password for the document. If it is empty no user password is applied. On EBCDIC platforms the password is expected in ebcdic encoding or EBCDIC-UTF-8. Default: empty

1. In order to pass arbitrary strings with this option the option list syntax described in »Unquoted string values«, page 10, may be useful.

2. Characters outside of Winansi encoding are only allowed in passwords for `compatibility=1.7ext3` or above

Table 3.5 Output processing options for `PDF_begin_document()`

option	description
createoutput	(Boolean) If false, the filename parameter is ignored and no output file or memory area is created. This option implies <code>compress=0</code> , <code>linearize=false</code> and <code>optimize=false</code> . Default: true
createpvf	(Boolean) If true, generate the PDF file in memory instead of on file. The supplied file name is the name of a virtual file which will be created with the call of <code>PDF_end_document()</code> . In this case <code>PDF_get_buffer()</code> cannot be called to fetch the PDF output data; instead, the name of the generated PVF file can be supplied to other PDFlib methods. This may be useful when generating documents which will be included in a PDF Portfolio. Default: false
filemode	(String, z/OS and USS only) Parameter string for setting the file mode of the document file and any temporary file (e.g. with the <code>linearize</code> option). The supplied string is appended to the default file mode of »wb«, «. The option <code>recordsize</code> must be consistent with the parameters specified in this option. Default: empty, or <code>recfm=v</code> for unblocked output (which is default, see option <code>recordsize</code>) Example strings: <code>recfm=fb,lrec1=80,space=(cyl,(1,5))</code> Honor attributes of the dataset if it has already been allocated: <code>recfm=*</code>
flush	(Keyword; only for <code>PDF_begin_document_callback()</code>) Set the flushing strategy (default: <code>page</code>): none flush only once at the end of the document page flush at the end of each page content flush after all fonts, images, file attachments, and pages heavy always flush when the internal 64 KB document buffer is full
inmemory	(Boolean; not for <code>PDF_begin_document_callback()</code>) If true and the <code>linearize</code> or <code>optimize</code> option is true as well, PDFlib will not create any temporary files for linearization, but will process the file in memory. This can result in tremendous performance gains on some systems (especially z/OS), but requires memory twice the size of the document. If false, a temporary file will be created for linearization and optimization. Default: false
recordsize	(Integer; z/OS and USS only) The record size of the output file, and any temporary file which may have to be created for the <code>linearize</code> and <code>optimize</code> options. Default: 0 (unblocked output)
remove-fragments	If true, a partial PDF output document which exists after an exception will be removed in <code>PDF_delete()</code> . Such PDF fragments are never usable as documents. This option has no effect if an empty filename has been specified, i.e. for in-memory PDF generation. Default: false
tempdirname	(String; not for <code>PDF_begin_document_callback()</code>) Directory where temporary files for the <code>linearize</code> and <code>optimize</code> options will be created. If this option is missing, PDFlib will generate temporary files in the current directory. This option is ignored if the <code>tempfilenames</code> option has been supplied. Default: not present
temp-filenames	(List of two strings; only for z/OS and USS) Full file names for two temporary files required for the <code>linearize</code> and <code>optimize</code> options. If empty, PDFlib will generate unique temporary file names. The user is responsible for deleting the temporary files after <code>PDF_end_document()</code> . If this option is supplied the filename parameter must not be empty. Default: not present

Table 3.6 Suboptions for the labels option in `PDF_begin/end_document()` and label option in `PDF_begin/end_page_ext()`

option	description
group	(String; only for <code>PDF_begin_document()</code>); required if the document uses page groups, but not allowed otherwise) The label will be applied to all pages in the specified group and all pages in all subsequent groups until a new label is applied. The group name must have been defined with the groups option in <code>PDF_begin_document()</code> .
hypertext-encoding	(Keyword) Encoding for the prefix option. An empty string is equivalent to unicode. Default: value of the global <code>hypertextencoding</code> option.
pagenumber	(Integer; only for <code>PDF_end_document()</code>); required if the document does not use page groups, but not allowed otherwise) The label will be applied to the specified page and subsequent pages until a new label is applied.
prefix	(Hypertext string) The label prefix for all labels in the range. Default: none
start	(Integer >= 1) Numeric value for the first label in the range. Subsequent pages in the range will be numbered sequentially starting with this value. Default: 1
style	(Keyword) The numbering style to be used. Default: none. <ul style="list-style-type: none"> none no page number; labels will only consist of the prefix. D decimal arabic numerals (1, 2, 3, ...) R uppercase roman numerals (I, II, III, ...) r lowercase roman numerals (i, ii, iii, ...) A uppercase letters (A, B, C, ..., AA, BB, CC, ...) a lowercase letters (a, b, c, ..., aa, bb, cc, ...)

Table 3.7 Suboptions for the viewer preferences option in `PDF_begin_document()` and `PDF_end_document()`

option	description
<code>centerwindow</code>	(Boolean) If true, position the document's window in the center of the screen. Default: false
<code>direction</code>	(Keyword) The reading order of the document, which affects the scroll ordering in double-page view and the side (left/right) of the first page for double-page layout in Acrobat (default l2r): l2r Left to right r2l Right to left (including vertical writing systems)
<code>displaydoctitle</code>	(Boolean; only true allowed in PDF/UA-1 mode) Display the Title document info field in Acrobat's title bar (true) or the file name (false). Default: true for PDF/UA-1, otherwise false
<code>duplex</code>	(Keyword; PDF 1.7) Paper handling option for the print dialog (default: none): DuplexFlipShortEdge Duplex and flip on the short edge of the sheet. DuplexFlipLongEdge Duplex and flip on the long edge of the sheet. none No paper handling specified. Simplex Print single-sided.
<code>enforce</code>	(List of keywords; PDF 2.0) Viewer preference settings which are enforced and shall not be overridden in the PDF viewer's user interface. The following keyword is supported: printscaling (Only relevant if the option <code>printscaling=none</code> has been supplied; otherwise ignored) Enforce printing without page scaling, i.e. at exact size.
<code>fitwindow</code>	(Boolean) Specifies whether to resize the document's window to the size of the first page. Default: false
<code>hidemenubar</code>	(Boolean) Specifies whether to hide the viewer's menu bar. Default: false
<code>hidetoolbar</code>	(Boolean) Specifies whether to hide the viewer's tool bars. Default: false
<code>hide-windowui</code>	(Boolean) Specifies whether to hide the viewer's window controls. Default: false
<code>nonfullscreen-pagemode</code>	(Keyword; only relevant if the <code>openmode</code> option is set to <code>fullscreen</code>) Specifies how to display the document on exiting full-screen mode (default: none): bookmarks display page and bookmark pane thumbnails display page and thumbnail pane layers display page and layer pane none display page only
<code>numcopies</code>	(Integer; PDF 1.7) The number of copies for the print dialog. Default: viewer-specific
<code>picktrayby-pdfsize</code>	(Boolean; PDF 1.7; no effect on macOS) Specifies whether the PDF page size is used to select the input paper tray in the print dialog. Default: viewer-specific
<code>printscaling</code>	(Keyword; PDF 1.6) Page scaling option to be selected when a print dialog is presented for the document. Supported keywords (default: <code>appdefault</code>): none No page scaling; this may be useful for printing page contents at their exact sizes. appdefault Use the current print scaling as specified in the PDF viewer.
<code>printpage-range</code>	(List with pairs of integers; PDF 1.7) Page numbers for the print dialog. Each pair denotes the start and end page numbers of a page range to be printed (first page is 1). Default: viewer-specific

3.2 Fetching PDF Documents from Memory

If a non-empty *filename* parameter has been supplied to `PDF_begin_document()` PDFlib writes PDF documents to a named disk file. Alternatively, PDF document data are generated in memory if the *filename* parameter is empty. In this case the PDF document data must be fetched from memory with `PDF_get_buffer()`. This is especially useful when shipping PDF from a Web server.

C++ `const char *get_buffer(long *size)`

Java C# `final byte[] get_buffer()`

Perl PHP `string get_buffer()`

C `const char *PDF_get_buffer(PDF *p, long *size)`

Get the contents of the PDF output buffer.

size (C, C++ and RPG language bindings only) C-style pointer to a memory location where the length of the returned data in bytes will be stored.

Returns A buffer full of binary PDF data for consumption by the client. The method returns a language-specific data type for binary data. The returned buffer must be used by the client before calling any other PDFlib method.

Details Fetch the full or partial buffer containing the generated PDF data. If this method is called between page descriptions, it will return the PDF data generated so far. If generating PDF into memory, this method must at least be called after `PDF_end_document()`, and will return the remainder of the PDF document. It can be called earlier to fetch partial document data. If there is only a single call to this method which happens after `PDF_end_document()` the returned buffer is guaranteed to contain the complete PDF document in a contiguous buffer.

Since PDF output contains binary characters, client software must be prepared to accept non-printable characters including null values.

Scope *object, document* (in other words: after `PDF_end_page_ext()` and before `PDF_begin_page_ext()`, or after `PDF_end_document()` and before `PDF_delete()`). This method can only be used if an empty filename has been supplied to `PDF_begin_document()`.

If the *linearize* option in `PDF_begin_document()` has been set to *true*, the scope is restricted to *object*, i.e. this method can only be called after `PDF_end_document()`.

Bindings C and C++: the *size* parameter is only used for C and C++ clients.

Other bindings: an object of appropriate length will be returned, and the *size* parameter must be omitted.

3.3 Page Methods

C++ Java C# **void begin_page_ext(double width, double height, String optlist)**

Perl PHP **begin_page_ext(float width, float height, string optlist)**

C **void PDF_begin_page_ext(PDF *p, double width, double height, const char *optlist)**

Add a new page to the document and specify various options.

width, height The *width* and *height* parameters are the dimensions of the new page in points (or user units, if the *userunit* option has been specified). They can be overridden by the options with the same name (the dummy value 0 can be used for the parameters in this case). A list of commonly used page formats can be found in Table 3.8. The PDFlib Tutorial lists applicable page size limits in Acrobat. See also Table 3.9 for more details (options *width* and *height*).

Table 3.8 Common ISO standard page size dimensions in points¹

<i>format</i>	<i>width</i>	<i>height</i>	<i>format</i>	<i>width</i>	<i>height</i>	<i>format</i>	<i>width</i>	<i>height</i>
<i>a0</i>	2380	3368	<i>a4</i>	595	842	<i>letter</i>	612	792
<i>a1</i>	1684	2380	<i>a5</i>	421	595	<i>legal</i>	612	1008
<i>a2</i>	1190	1684	<i>a6</i>	297	421	<i>ledger</i>	1224	792
<i>a3</i>	842	1190	<i>b5</i>	501	709	<i>11x17</i>	792	1224

¹ Note that ISO B5 is different from JIS B5. More information about ISO, Japanese, and U.S. standard formats can be found at www.cl.cam.ac.uk/~mgk25/iso-paper.html

optlist An option list with page options according to Table 3.9. These options have lower priority than identical options specified in *PDF_end_page_ext()*:

action, artbox, associatedfiles, bleedbox, blocks, cropbox, defaultcmyk, defaultgray, defaultrgb, duration, group, height, label, mediabox, metadata, outputintents, pagenumber, rotate, separationinfo, taborder, topdown, transition, transparencygroup, trimbox, userunit, viewports, width

Details This method resets all text, graphics, and color state parameters to their default values and establishes a coordinate system according to the *topdown* option.

PDF/A Restrictions apply to the *transparencygroup* option.

PDF/UA Restrictions apply to the *taborder* option.

PDF/VT The following options are not allowed: *group, pagenumber*.

PDF/X Restrictions apply to the *transparencygroup* and *defaultgray/rgb/cmyk* options.

Scope *document*; this method starts *page* scope, and must always be paired with a matching *PDF_end_page_ext()* call.

C++ Java C# **void end_page_ext(String optlist)**

Perl PHP **end_page_ext(string optlist)**

C **void PDF_end_page_ext(PDF *p, const char *optlist)**

Finish a page and apply various options.

optlist An option list according to Table 3.9. Options specified in `PDF_end_page_ext()` have priority over identical options specified in `PDF_begin_page_ext()`. The following options can be used:

associatedfiles, action, artbox, bleedbox, blocks, cropbox, defaultcmyk, defaultgray, defaultrgb, duration, group, height, label, mediabox, metadata, rotate, taborder, transition, transparency-group, trimbox, userunit, viewports, width

Scope *page*; this method terminates *page* scope, and must always be paired with a matching `PDF_begin_page_ext()` call.

In Tagged PDF mode all direct and pseudo items must be closed before calling this method.

Table 3.9 Page options for `PDF_begin_page_ext()` and `PDF_end_page_ext()`

option	description
action	(Action list; not for PDF/A) List of page actions for one or more of the following events (default: empty list): open Actions to be performed when the page is opened. close Actions to be performed when the page is closed. Individual actions supplied to the action option of <code>PDF_end_page_ext()</code> override actions imported from PDF pages with the useactions option of <code>PDF_fit_pdi_page()</code> . Rendition actions which have been created with the target option are only allowed on the page where the target annotation is located.
associatedfiles	(List of asset handles; only for PDF 2.0 and PDF/A-3) Asset handles for files associated with the page. The files must have been loaded with <code>PDF_load_asset()</code> and type=attachment.
artbox bleedbox cropbox	(Rectangle) Specify the ArtBox, BleedBox, or CropBox for the current page, respectively. The coordinates are specified in the default coordinate system. Default: no box entries
blocks	(POCA container handle; may be supplied to <code>PDF_begin_page_ext()</code> or <code>PDF_end_page_ext()</code> , but not to both methods for the same page; only available in PPS) Handle for a dictionary container created with <code>PDF_poca_new()</code> which contains PDFlib Block definitions for the PDFlib Personalization Server (PPS). The specified Blocks will be attached to the page. The dictionary must have been created with the option usage=blocks. Default: no Blocks
defaultgray ¹ defaultrgb ¹ defaultcmyk ¹	(ICC handle or keyword) Set a default gray, RGB, or CMYK color space for the page according to the supplied ICC profile handle. The option <code>defaultrgb</code> also supports the keyword <code>srgb</code> . The specified color space is used to map device-dependent gray, RGB, or CMYK colors on the page.
duration	(Float) Set the page display duration in seconds for the current page if <code>openmode=fullscreen</code> (see Table 3.1). Default: 1
group ¹	(String; required if the document uses page groups, but not allowed otherwise; not allowed in PDF/VT mode or if a document part hierarchy is created) Name of the page group to which the page will belong. This name can be used to keep pages together in a page group and to address pages with <code>PDF_resume_page()</code> . The group name must have been defined with the <code>groups</code> option in <code>PDF_begin_document()</code> .

Table 3.9 Page options for `PDF_begin_page_ext()` and `PDF_end_page_ext()`

option	description
height	(Float or keyword; not allowed in <code>PDF_end_page_ext()</code> if the <code>topdown</code> option is true) Dimensions of the new page in points (or user units, if the <code>userunit</code> option has been specified). In order to produce landscape pages use <code>width > height</code> or the <code>rotate</code> option. PDFlib uses width and height to construct the page's MediaBox, but the MediaBox can also explicitly be set using the <code>mediabox</code> option. The width and height options override the parameters with the same name. The following symbolic ISO page size names can be used as keywords by appending <code>.width</code> or <code>.height</code> (e.g. <code>a4.width</code> , <code>a4.height</code>): <code>a0</code> , <code>a1</code> , <code>a2</code> , <code>a3</code> , <code>a4</code> , <code>a5</code> , <code>a6</code> , <code>b5</code> , <code>letter</code> , <code>legal</code> , <code>ledger</code> , <code>11x17</code>
label	(Option list) An option list according to Table 3.6 specifying symbolic page names. The page name will be displayed as a page label (instead of the page number) in Acrobat's status line. The specified numbering scheme will be used for the current and subsequent pages until it is changed again. The combination of style/prefix/start values must be unique within a document.
mediabox	(Rectangle; not allowed if the <code>topdown</code> option is true) Change the MediaBox for the current page. The coordinates are specified in the default coordinate system. By default, the MediaBox will be created by using the width and height parameters. The <code>mediabox</code> option overrides the width and height options and parameters.
metadata	(Option list) Metadata for the page (see Section 15.2, »XMP Metadata«, page 277)
outputintents¹	(Output intent handle; PDF 2.0) ICC profile to be used as output intent for the page. The handle must have been retrieved with <code>PDF_load_iccprofile()</code> and <code>usage=pageoutputintent</code> or <code>usage=outputintent</code> or with <code>PDF_process_pdi()</code> and <code>action=copypageoutputintent</code> .
pagenumber¹	(Integer; not allowed in PDF/VT mode or if a document part hierarchy is created) If this option is specified with a value <code>n</code> , the page will be inserted before the existing page <code>n</code> within the page group specified in the <code>group</code> option (or the document if the document doesn't use page groups). If this option is not specified the page is inserted at the end of the group.
rotate	(Integer) The page rotation value. The rotation will affect page display, but does not modify the coordinate system. Possible values are <code>0</code> , <code>90</code> , <code>180</code> , <code>270</code> . Default: <code>0</code>
separation-info¹	(Option list; deprecated and not available in PDF 2.0) Color separation details for the current page: pages (Integer; required for the first page of a set of separation pages, but not allowed for subsequent pages of the same set) The number of pages which belong to the same set of separation pages comprising the color data for a single composite page. All pages in the set must appear sequentially in the file. spotname (String; required unless <code>spotcolor</code> has been supplied) The name of the colorant for the current page. spotcolor (Spot color handle) A color handle describing the colorant for the current page.
taborder	(Keyword; PDF 1.5; only structure allowed in PDF/UA-1) Keyword specifying the tab order for form fields and annotations (Default: structure in Tagged PDF mode for PDF 1.5 and above, otherwise none): annotations (PDF 2.0) The tab order is defined by the order of annotations. column Column by column from top to bottom, where columns are ordered as specified by the <code>direction</code> suboption of the <code>viewerpreferences</code> option of <code>PDF_begin/end_document()</code> . none The tab order is unspecified. structure Form fields and annotations are visited in the order in which they appear in the structure tree. row Row by row starting at the topmost row, where the direction within a row is as specified by the <code>direction</code> suboption of the <code>viewerpreferences</code> option of <code>PDF_begin/end_document()</code> . widgets (PDF 2.0) The tab order is defined by the order of form fields, then other annotations.

Table 3.9 Page options for `PDF_begin_page_ext()` and `PDF_end_page_ext()`

option	description
topdown ¹	(Boolean) If <code>true</code> , the origin of the coordinate system at the beginning of the page is located in the top left corner of the page, and <code>y</code> coordinates increase downwards; otherwise the default coordinate system is used. Default: <code>false</code>
transition	(Keyword) Set the page transition for the current page in order to achieve special effects which may be useful when displaying the PDF in Acrobat's full-screen mode as presentations if <code>openmode=fullscreen</code> (see Table 3.1). Default: <code>replace</code>
split	Two lines sweeping across the screen reveal the page
blinds	Multiple lines sweeping across the screen reveal the page
box	A box reveals the page
wipe	A single line sweeping across the screen reveals the page
dissolve	The old page dissolves to reveal the page
glitter	The dissolve effect moves from one screen edge to another
replace	The old page is simply replaced by the new page
fly	(PDF 1.5) The new page flies into the old page.
push	(PDF 1.5) The new page pushes the old page off the screen
cover	(PDF 1.5) The new page slides on to the screen and covers the old page.
uncover	(PDF 1.5) The old page slides off the screen and uncovers the new page.
fade	(PDF 1.5) The new page gradually becomes visible through the old one.
transparency-group	(Option list or keyword; not for PDF/A-1 and PDF/X-3; restrictions apply to PDF/A-2/3 and PDF/X-4/5) Create a transparency group for the page. The following keywords are supported (default: <code>auto</code>):
auto	If transparent objects are present on the page itself, on a placed template or image, on imported graphics (not counting objects in isolated transparency groups since these don't affect objects outside the group) or on annotations, a page transparency group is created with a suitable color space; otherwise no transparency group is created.
none	(Not allowed for PDF/A-2/3 without output intent and PDF/X-5n if transparency is used on the page) Don't create a transparency group for the page.
	The following suboptions can be used to explicitly create a transparency group:
colorspace	(Keyword or ICC profile handle; required for PDF/A-2/3 without output intent and PDF/X-5n if transparency is used on the page) Blending color space for the page (default: <code>none</code>):
DeviceCMYK	PDF/A-2/3 and PDF/X-4: only allowed with a CMYK output intent or if the <code>defaultcmyk</code> option has been supplied. PDF/X-5n: only allowed if the output intent contains the Colorants Cyan, Magenta, Yellow and Black or if the <code>defaultcmyk</code> option has been supplied.
DeviceGray	PDF/A-2/3: only allowed with a gray, RGB or CMYK output intent or if the <code>defaultgray</code> option has been supplied. PDF/X-4: only allowed with a gray or CMYK output intent or if the <code>defaultgray</code> option has been supplied. PDF/X-5n: only allowed if the output intent contains the Colorant Black or if the <code>defaultgray</code> option has been supplied.
DeviceRGB	PDF/A-2/3 and PDF/X-4: only allowed with an RGB output intent or if the <code>defaultrgb</code> option has been supplied. PDF/X-5n: only allowed if the <code>defaultrgb</code> option has been supplied.
none	(Not allowed for PDF/A-2/3 without output intent and PDF/X-5n if transparency is used on the page) No color space is emitted for the transparency group.
srgb	Keyword for selecting the sRGB color space
knockout	(Boolean) Specifies whether the page group is a knockout group, which means that the group elements do not composite with each other; objects knock out earlier elements in the group. Default: <code>false</code>
trimbox	(Rectangle) Specify the TrimBox for the current page. The coordinates are specified in the default coordinate system. Default: no TrimBox entry

Table 3.9 Page options for `PDF_begin_page_ext()` and `PDF_end_page_ext()`

option	description
<code>userunit</code>	(Float or keyword; PDF 1.6) A number in the range 1..75 000 specifying the size of a user unit in points, or one of the keywords mm, cm, or m which scales to the respective unit. User units don't change the actual page contents; they are only a hint to Acrobat which is used when printing the page or using the measurement tools. Default: 1 (i.e. one unit is one point)
<code>viewports</code>	(List of option lists; PDF 1.7ext3) One or more georeferenced areas (viewports) on the page; see Section 13.4, »Geospatial Features«, page 260, for details. Viewports allow different geospatial references (specified by the georeference option) to be used on different areas of the page, e.g. for multiple maps. The ordering of the option lists in the viewports list is relevant for overlapping viewports: the last viewport which contains a point will be used for that point.
<code>width</code>	(Float or keyword; not allowed in <code>PDF_end_page_ext()</code> if the <code>topdown</code> option is true) See height option.

1. Only for `PDF_begin_page_ext()`

C++ Java C# **void suspend_page(String optlist)**

Perl PHP **suspend_page(string optlist)**

C **void PDF_suspend_page(PDF *p, const char *optlist)**

Suspend the current page so that it can later be resumed.

optlist An option list for future use.

Details The full graphics, color, text and layer states of the current page are saved internally. The page can later be resumed with `PDF_resume_page()` to add more content. Suspended pages must be resumed before they can be closed.

Scope *page*; this method starts *document* scope, and must always be paired with a matching `PDF_resume_page()` call. In Tagged PDF mode all direct and pseudo items must be closed before calling this method.

C++ Java C# **void resume_page(String optlist)**

Perl PHP **resume_page(string optlist)**

C **void PDF_resume_page(PDF *p, const char *optlist)**

Resume a page to add more content to it.

optlist An option list according to Table 3.10. The following options can be used:
group, *pagenumber*

Details The page must have been suspended with `PDF_suspend_page()`. It will be opened again so that more content can be added. All suspended pages must be resumed before they can be closed, even if no more content has been added.

In Tagged PDF mode it must be kept in mind that resuming a page does not restore any structure item. Instead, the item which is active when `PDF_resume_page()` is called will be the current item for subsequent page contents. It is recommended to use `PDF_activate_item()` to restore a specific structure element on the page as parent for subsequently generated contents.

Scope *document*; this method starts *page* scope, and must always be paired with a matching `PDF_suspend_page()` call.

Table 3.10 Options for `PDF_resume_page()`

option	description
group	<i>(String; required if the document uses page groups, but not allowed otherwise) Name of the page group of the resumed page. The group name must have been defined with the groups option in <code>PDF_begin_document()</code>.</i>
pagenumber	<i>(Integer) If this option is supplied, the page with the specified number within the page group chosen in the group option (or in the document if the document doesn't use page groups) will be resumed. If this option is missing the last page in the group will be resumed.</i>

3.4 Layers

C++ Java C# **int define_layer(String name, String optlist)**

Perl PHP **int define_layer(string name, string optlist)**

C **int PDF_define_layer(PDF *p, const char *name, int len, const char *optlist)**

Create a new layer definition.

name (Hypertext string) The name of the layer.

len (C language bindings only) Length of *name* (in bytes). If *len* = 0 a null-terminated string must be provided.

optlist An option list with layer settings:

- ▶ Layer control options according to Table 3.11:
creatorinfo, defaultstate, initialexportstate, initialprintstate, initialviewstate, intent, language, onpanel, pageelement, printsubtype, removeunused, zoom
- ▶ Encoding options for C, Perl, PHP, and Ruby with *stringformat=legacy*:
hypertextencoding, hypertextformat (see Table 2.1)

Returns A layer handle which can be used in calls to *PDF_begin_layer()* and *PDF_set_layer_dependency()* until the end of the enclosing *document* scope.

Details PDFlib issues a warning if a layer was defined but hasn't been used in the document. Layers which are used on multiple pages should be defined only once (e.g. before creating the first page). If *PDF_define_layer()* is called repeatedly on multiple pages, the layer definitions will accumulate (even if they have the same name), which is usually not desired.

Once this method has been called *PDF_open_pdi_document()* must not be called for a layered PDF document and the option *uselayers=false*. Vice versa, this method must not be called if *PDF_open_pdi_document()* has already been called for a layered PDF document and the option *uselayers=false*.

PDF/A-2/3: some options are restricted.

PDF/UA Some options are restricted.

PDF/X-4/5: some options are restricted.

Scope any except *object*; requires PDF 1.5

Table 3.11 Options for *PDF_define_layer()*

option	description
creatorinfo	(Option list; not for PDF/A-2/3, PDF/X-4/5, and PDF/UA-1) An option list describing the content and the creating application. Both of the following entries are required if this option is used: creator (Hypertext string) The name of the application which created the layer subtype (String) The type of content. Suggested values are Artwork and Technical.
defaultstate	(Boolean) Specifies whether or not the layer is visible by default. Default: true
initial-exportstate	(Boolean; not for PDF/A-2/3, PDF/X-4/5, and PDF/UA-1) The layer's recommended export state. If true, Acrobat includes the layer when converting/exporting to older PDF versions or other document formats. Default: true

Table 3.11 Options for `PDF_define_layer()`

option	description
initial-printstate	(Boolean; not for PDF/A-2/3, PDF/X-4/5, and PDF/UA-1) The layer's recommended printing state. If true, Acrobat includes the layer when printing the document. Default: true
initial-viewstate	(Boolean; not for PDF/A-2/3, PDF/X-4/5, and PDF/UA-1) The layer's recommended viewing state. If true, Acrobat displays the layer when opening the document. Default: true
intent	(Keyword) Intended use of the graphics: View or Design. Default: View
language	(Option list; not for PDF/A-2/3, PDF/X-4/5, and PDF/UA-1) SLanguage of the layer: lang (String; required) The language and possibly locale in the format described in Table 3.1 for the lang option preferred (Boolean) If true this layer is used if there is only a partial match between the layer and the system language. Default: false
onpanel	(Boolean; not for PDF/A-2/3, PDF/X-4/5, and PDF/UA-1) If false, the layer name will not be visible in Acrobat's layer panel, and therefore cannot be manipulated by the user. Default: true
pageelement	(Keyword; not for PDF/A-2/3, PDF/X-4/5, and PDF/UA-1) Specifies that the layer contains a pagination artifact: one of HF (header/footer), FG (foreground image or graphic), BG (background image or graphic), or L (logo).
printssubtype	(Option list; not for PDF/A-2/3, PDF/X-4/5, and PDF/UA-1) Specifies whether the layer is intended for printing: subtype (Keyword) One of Trapping, PrintersMarks, or Watermark specifying the kind of content in the layer. printstate (Boolean) If true, Acrobat activates the layer contents upon printing.
removeunused	(Boolean) If true and the layer is not used on a page, the layer will not be included in the page's layer list. A layer is considered used on a page if it has been supplied to <code>PDF_begin_layer()</code> at least once on that page. Default: false
zoom	(List of floats or percentages; not for PDF/A-2/3, PDF/X-4/5, and PDF/UA-1) One or two values specifying the layer's visibility depending on the zoom factor (1.0 means a zoom factor of 100 percent). If one value is provided, it will be used as the maximum zoom factor at which the layer should be visible; if two values are provided they specify the minimum and maximum zoom factor. The keyword maxzoom can be used to specify the largest possible zoom factor.

C++ Java C# **void set_layer_dependency(String type, String optlist)**

Perl PHP **set_layer_dependency(string type, string optlist)**

C **void PDF_set_layer_dependency(PDF *p, const char *type, const char *optlist)**

Define layer relationships.

type The type of dependency or relationship according to Table 3.12.

Table 3.12 Dependency and relationship types for layers

type	notes; options specific for this type
GroupAllOn	The layer specified in the depend option will be visible if all layers specified in the group option are visible. Options specific for this type: depend, group
GroupAnyOn	The layers specified in the depend option will be visible if any layer specified in the group option is visible. Options specific for this type: depend, group
GroupAllOff	The layer specified in the depend option will be visible if all layers specified in the group option are invisible. Options specific for this type: depend, group

Table 3.12 Dependency and relationship types for layers

type	notes; options specific for this type
GroupAnyOff	The layer specified in the <code>depend</code> option will be visible if any layer specified in the <code>group</code> option is invisible. Options specific for this type: <code>depend</code> , <code>group</code>
Lock	(PDF 1.6) The layers specified in the <code>group</code> option are locked, i.e. their state cannot be changed interactively in Acrobat. Options specific for this type: <code>group</code>
Parent	Specify a hierarchical relationship between the layer specified in the <code>parent</code> option and the layers specified in the <code>children</code> option. Setting the parent to invisible automatically sets its children to invisible. A layer cannot belong to more than one parent layer. Options specific for this type: <code>children</code> , <code>parent</code>
Radiobtn	Specify a radio button relationship between the layers specified in the <code>group</code> option. This means that at most one layer in the group is visible at a time, which is particularly useful for multiple language layers. Option specific for this type: <code>group</code>
Title	The layer specified in the <code>parent</code> option does not control any page contents directly, but serves as a hierarchical separator for the layers specified in the <code>children</code> option. Options specific for this type: <code>children</code> , <code>parent</code>

optlist An option list for layer dependencies:

- ▶ Layer dependency options according to Table 3.13: `children`, `depend`, `group`, `parent`
- ▶ Encoding options for C and Perl, PHP, Ruby with `stringformat=legacy`: `hypertextencoding` (see Table 2.1)

Details Layer relationships specify the presentation of layer names in Acrobat’s layer pane as well as the visibility of one or more layers when the user interactively enables or disables layers.

Scope any except *object*; layer relationships should be specified after all layers have been defined; requires PDF 1.5

Table 3.13 Options for `PDF_set_layer_dependency()`

option	description
children	(List of layer handles; only for <code>type=Parent</code> and <code>Title</code>) One or more layer handles specifying the layers subordinate to the provided parent layer.
depend	(Layer handle; only for <code>type=GroupAllOn</code> , <code>GroupAnyOn</code> , <code>GroupAllOff</code> , and <code>GroupAnyOff</code>) The layer which is controlled by the layers specified in the <code>group</code> option.
group	(List of layer handles; only for <code>type=GroupAllOn</code> , <code>GroupAnyOn</code> , <code>GroupAllOff</code> , <code>GroupAnyOff</code> , <code>Lock</code> , and <code>Radiobtn</code>) One or more layer handles comprising the group. For <code>type=Lock</code> all layers in the group will be locked.
parent	(Layer handle; only for <code>type=Parent</code> and <code>Title</code>) The layer which is the parent of the layers specified in the <code>children</code> option.

C++ Java C# **void begin_layer(int layer)**

Perl PHP **begin_layer(int layer)**

C **void PDF_begin_layer(PDF *p, int layer)**

Start a layer for subsequent output on the page.

layer The layer's handle, which must have been retrieved with `PDF_define_layer()`.

Details All content placed on the page after this call, but before any subsequent call to `PDF_begin_layer()` or `PDF_end_layer()` will be part of the specified layer. The content's visibility depends on the layer's settings.

This method activates the specified layer, and deactivates any layer which may be currently active.

Layers for annotations, images, graphics, templates, and form fields can be controlled with the *layer* option of the respective methods.

Scope *page*; requires PDF 1.5

C++ Java C# **void end_layer()**

Perl PHP **end_layer()**

C **void PDF_end_layer(PDF *p)**

Deactivate all active layers.

Details Content placed on the page after this call will not belong to any layer. All layers must be closed at the end of a page.

In order to switch from layer A to layer B a single call to `PDF_begin_layer()` is sufficient; it is not required to explicitly call `PDF_end_layer()` to close layer A. `PDF_end_layer()` is only required to create unconditional content (which is always visible), and to close all layers at the end of a page.

Scope *page*; requires PDF 1.5

4 Font and Text Methods

API methods in this chapter:

- ▶ `PDF_load_font()`
- ▶ `PDF_close_font()`
- ▶ `PDF_info_font()`
- ▶ `PDF_set_text_option()`
- ▶ `PDF_set_font()`
- ▶ `PDF_set_text_pos()`
- ▶ `PDF_show()`
- ▶ `PDF_show_xy()`
- ▶ `PDF_continue_text()`
- ▶ `PDF_stringwidth()`
- ▶ `PDF_begin_font()`
- ▶ `PDF_end_font()`
- ▶ `PDF_begin_glyph_ext()`
- ▶ `PDF_end_glyph()`
- ▶ `PDF_encoding_set_char()`

4.1 Font Handling

C++ Java C# `int load_font(String fontname, String encoding, String optlist)`

Perl PHP `int load_font(string fontname, string encoding, string optlist)`

C `int PDF_load_font(PDF *p, const char *fontname, int len, const char *encoding, const char *optlist)`

Search for a font and prepare it for later use.

fontname (Name string) Name of the font. It can alternatively be provided via the *fontname* option which overrides this parameter. See option *fontname* in Table 4.1 for details.

len (C language binding only) Length of *fontname* in bytes. If *len* = 0 a null-terminated string must be provided.

encoding Name of the encoding. It can alternatively be provided via the *encoding* option which overrides this parameter. See option *encoding* in Table 4.1 for details.

optlist An option list with the following options:

- ▶ General option: *errorpolicy* (see Table 1.5)
- ▶ Font loading options according to Table 4.1:
ascender, autosubsetting, capheight, colormode, descender, embedding, encoding, fallbackfonts, fontname, fontstyle, foregroundcolor, foregroundopacity, initialssubset, keepfont, keepnative, linegap, metadata, optimizeinvisible, preservepua, readfeatures, readkerning, readselectors, readshaping, readverticalmetrics, replacementchar, simplefont, skipembedding, strikeselect, subsetlimit, subsetminsize, subsetting, unicomemap, vertical, xheight

Returns A font handle for later use with `PDF_info_font()`, text output methods, and the *font* text appearance option. If the requested font/encoding combination cannot be loaded due

to a configuration problem (e.g. the font couldn't be found), an error code of -1 (in PHP: o) will be returned or an exception raised. The error behavior can be changed with the *errorpolicy* option.

If the method returns an error you can request the reason of the failure with *PDF_get_errmsg()*. Otherwise the value returned by this method can be used as font handle when calling other font-related methods. The returned handle doesn't have any significance to the user other than serving as a font handle.

The returned font handle is valid until the font is closed with *PDF_close_font()*. Finishing the document with *PDF_end_document()* closes each open font handle unless the option *keepfont* has been supplied or the font has been loaded in *object* scope (i.e. outside of any document).

Details This method prepares a font for later use.

Repeated calls: when this method is called again with the same font name and the same encoding, the same font handle as in the first call is returned. Trying to load a font again will fail if *embedding=false* in the first call and *embedding=true* in the second call.

Implicit font loading: in addition to explicitly loading a font with *PDF_load_font()*, some API methods (e.g. *PDF_add/create_textflow()* or *PDF_fill_textblock()*) can implicitly load a font for which the font name has been specified in an option list. A new font handle is created unless the font has already been loaded earlier.

In non-Unicode language bindings, the option *textformat=auto* behaves as follows (all UTF formats are allowed for both cases):

- ▶ Wide character encodings: text in the loaded font is expected in text format *utf16*
- ▶ Byte- and multibyte encodings: text in the loaded font is expected in text format *bytes*.

PDF/A All fonts must be embedded.

PDF/UA All fonts must be embedded.

PDF/X All fonts must be embedded.

Scope any

Table 4.1 Font options for *PDF_load_font()* and implicit font loading with option lists

option	description
<i>ascender</i>	(Integer between -2048 and 2048) Force the corresponding typographic property to the specified value. This will override any values found in the font, and is especially useful if the font does not contain any such information (e.g. Type 3 fonts). Default: the value in the font if present, or an estimated value otherwise (which can be queried with <i>PDF_info_font()</i>)
<i>auto-subsetting</i>	(Boolean; ignored if the subsetting option has been supplied) Automatically decide whether the font will be subset, subject to the <i>subsetlimit</i> and <i>subsetminsize</i> options and actual usage of glyphs. Default: true
<i>capheight</i>	(Integer between -2048 and 2048) See <i>ascender</i> above.

Table 4.1 Font options for `PDF_load_font()` and implicit font loading with option lists

option	description
colormode	<p>(Keyword; only relevant for OpenType color fonts) Processing mode for OpenType color fonts (see PDFlib Tutorial for details). The color mode of a font cannot be changed; if the font is loaded again, the colormode option is ignored and the color mode of the first load operation is used (default: combined):</p> <p>ignorecolor Ignore color glyphs and create a Type 3 font only from monochromatic glyphs; recommended for black/white workflows. However, some color fonts (especially sbix fonts) contain only dummy mono glyphs without outlines and are therefore unusable with ignorecolor.</p> <p>ignoremono Ignore monochromatic glyphs and create a Type 3 font from color glyphs; recommended if the color font is intended as fallback for another font. OpenType features, ligatures and shaping are not available in this mode.</p> <p>combined Create a Type 3 font from monochromatic and color glyphs. Glyph IDs in the resulting font are identical to those in the original OpenType font.</p>
descender	(Integer between -2048 and 2048) See ascender above.
embedding	(Boolean; must be true for PDF/A, PDF/UA and PDF/X) Controls whether the font is embedded. If a font is not embedded, only glyph widths and a general description are included in the PDF output. Default: true Font embedding can also be controlled with the skipembedding option.
encoding	<p>(String; required as parameter of <code>PDF_load_font()</code>, but optional for implicit font loading) Encoding in which incoming text for this font is interpreted:</p> <p>Wide character encodings: unicode or glyphid</p> <p>Byte and multibyte encodings:</p> <ul style="list-style-type: none"> ▶ name of an 8-bit encoding or (deprecated) a non-Unicode (legacy) CMap ▶ (not for Unicode-capable language bindings) cp932, cp936, cp949, or cp950 for CJK codepages ▶ host or auto or the name of a user-defined encoding or an encoding known to the operating system ▶ builtin to select the font's internal encoding (mostly for legacy symbol fonts); <p>Default (only for implicit font loading with option lists): unicode</p> <p><code>PDF_load_font()</code>: this option can alternatively be provided as method parameter.</p> <p><code>PDF_fill_textblock()</code>: this option is required unless the string in the text parameter is empty and the defaulttext property is used, or the font option has been supplied.</p>
fallbackfonts	<p>(List of option lists according to Table 4.2 Specify one or more fallback fonts for the loaded font. Each fallback font must be defined by a font handle in the font suboption or suitable suboptions for implicit font loading.</p> <p>If glyphcheck=replace and the text contains a character which is not part of the base font's 8-bit encoding, or the base font does not contain a glyph for the character, or glyph replacement is forced via the forcechars suboption, PDFlib will search a glyph for this character in all specified fallback fonts in the order in which they are listed. If a suitable glyph is found in one of the fallback fonts, the character will be rendered with this glyph; otherwise the usual glyph replacement mechanism applies.</p>
fontname	<p>(Name string; required for implicit font loading except for <code>PDF_fill_textblock()</code> if the text appearance option font is not specified) Real or alias name of the font (case is significant). This name is used to locate the font data. If the font file name is supplied the suffix must be omitted. On Windows font style names can be appended to the font name after a comma (see PDFlib Tutorial for details). If fontname starts with an '@' character the font is applied in vertical writing mode.</p> <p><code>PDF_load_font()</code>: the font name can alternatively be provided as method parameter.</p>
fontstyle	(Keyword; deprecated) Controls the creation of artificial font styles. Possible keywords are normal, bold, italic, bolditalic. Text created with this font will be styled with the fakebold and/or italicangle text appearance options as appropriate. Unless another value of italicangle has been set, -12 is used. Default: normal

Table 4.1 Font options for `PDF_load_font()` and implicit font loading with option lists

option	description
foreground-color	(Color; only relevant for OpenType color fonts with COLR table) Color of glyph layers which are designated as »foregroundcolor«. Supported color spaces are lab, gray, rgb (interpreted as sRGB) and cmyk. Default: black
foreground-opacity	(Float or percentage; only relevant for OpenType color fonts with COLR table; in PDF/A-1 and PDF/X-3 only the value 1 is allowed) Opacity of glyph layers which are designated as »foregroundcolor«. Default: 1
initialsubset	(List of Unichars or Unicode ranges, or list of keywords; only relevant for embedding=true and sub-setting=true) Specify the Unicode values for which glyphs will be included in the initial font subset. This can be used to reduce the PDF output file size by creating identical subsets, which facilitates later optimizations when merging multiple documents. The Unicode values can be specified explicitly by Unichars or Unicode ranges or implicitly by the name of an 8-bit encoding. Supported keywords (default: empty): empty The initial font subset is empty. any 8-bit encoding name All Unicode values in the encoding are included in the initial subset.
keepfont	(Boolean; not for Type 3 fonts and OpenType color fonts) If false the font will be deleted automatically in <code>PDF_end_document()</code> . If true the font can also be used in subsequent documents until <code>PDF_close_font()</code> has been called. Default: true if <code>PDF_load_font()</code> is called in object scope, otherwise false
keepnative	(Boolean; only relevant for non-Unicode-aware language bindings; forced to false for embedded fonts; deprecated) If false, text in this font is converted to CID values when creating PDF output. The text supplied to API methods must still match the selected CMap. However, the font can be used in Textflow and all simple text output methods (but not in form fields). If true, text in this font is written to the PDF output in its native format according to the specified CMap. The font can be used in form fields and all simple text output methods, but not in Textflow. Default: false for TrueType fonts, true otherwise
linegap	(Integer between -2048 and 2048) See ascender above.
metadata	(Option list) Supply metadata for the font (see Section 15.2, »XMP Metadata«, page 277)
optimize-invisible	(Boolean; not for PDF/X-3) If true, fonts which are exclusively used for invisible text (i.e. text-rendering=3) are not embedded even if embedding=true. This may be useful to avoid font embedding for PDF/A output with invisible text containing OCR results. Default: false
preservepua	(Boolean) If true, characters which are mapped to a Unicode value in the Private Use Area (PUA) in the font retain their PUA value in the PDF output. This may be useful if the PUA characters carry locally defined semantics such as Japanese Gaiji/EUDC characters. If false, PUA characters are mapped to U+FFFD (Unicode replacement character) in the ToUnicode CMap in the PDF output. Default: false
readfeatures	(Boolean) Specifies whether the OpenType feature tables are read from the font. Applying OpenType features to text is controlled by the features option (see Table 5.4). Setting this option to false may speed up font loading if OpenType features are not required. Default: true
readkerning	(Boolean) Controls whether or not kerning values will be read from the font. Actually applying the kerning values to text is controlled by the kerning text option (see Table 4.6). Setting this option to false may speed up font loading if kerning is not required. Default: true for horizontal writing mode, false for vertical writing mode
readselectors	(Boolean; only relevant for TrueType and OpenType fonts) If true, variation selectors are read from the font if available. This is required for automatically substituting Ideographic Variation Sequences (IVS) within Unicode text. Default: true
readshaping	(Boolean) Specifies whether the shaping tables of a TrueType or OpenType font will be read, which is a requirement for complex script shaping. Actually shaping text is controlled by the shaping option (see Table 5.4). Setting readfeatures to false can save memory if shaping is not required. Default: true

Table 4.1 Font options for `PDF_load_font()` and implicit font loading with option lists

option	description
readvertical-metrics	(Boolean) If true and the option <code>vertical</code> is true the vertical metrics of a TrueType or OpenType font (if present) are used for formatting the text output. Default: false
replacementchar	(Unicode or keyword; only relevant for <code>glyphcheck=replace</code>) Glyphs which are not available in the selected font and which cannot be substituted by fallback fonts or typographically similar characters will be replaced with the specified Unicode value. If the font doesn't contain any glyph for the specified Unicode character, the behavior of <code>auto</code> will be applied. <code>U+0000</code> can be used to specify the font's »missing glyph« symbol (not allowed in PDF/A, PDF/UA and PDF/X-4/5). For symbolic fonts loaded with <code>encoding=builtin</code> the byte code must be supplied instead of the Unicode value. The following keywords can be used as an alternative to a Unicode character (default: <code>auto</code>): <ul style="list-style-type: none"> auto The first character from the following list for which a glyph is available in the font is used as replacement character: <code>U+00A0</code> (NO-BREAK SPACE), <code>U+0020</code> (SPACE), <code>U+0000</code> (missing glyph symbol). drop No output is created for the character. error An exception is thrown if a typographically similar character is not available. This may be used to avoid unreadable text output.
simplefont	(Boolean; only relevant for fonts with TrueType outlines and an 8-bit encoding) If this option is true and <code>subsetting=false</code> the font is emitted as simple font instead of as a CID font. For some PDF viewers this is required for successful font substitution if the font is not installed on the viewing machine. Default: false
skip-embedding	(List of keywords; only relevant for <code>embedding=true</code> ; in PDF/A, PDF/X and PDF/UA only an empty list is allowed) Ignore font embedding if the font satisfies one or more conditions. This may be useful in situations where font embedding is generally desired, but an unembedded font is preferable for specific situations. Supported keywords: <ul style="list-style-type: none"> fstype The font is a TrueType or OpenType font and cannot be embedded because it doesn't permit embedding according to the <code>fsType</code> flag in the font's OS/2 table. hostfont The font was loaded as a host font. latincore The font is included in the set of Standard Type 1 fonts (also called Latin core fonts) (see PDFlib Tutorial for full list), but cannot be embedded because no font file is available. Default: empty list in PDF/A, PDF/UA or PDF/X mode or if the embedding option was specified, otherwise {latincore}
strikeselect	(Integer; only relevant for OpenType color fonts with <code>sbix</code> color table) Select a bitmap strike based on <code>ppem</code> value, i.e. the number of bitmap pixels in the design unit: <ul style="list-style-type: none"> 0 Select the strike with the smallest <code>ppem</code> value to minimize output file size. 100 Select the strike with the largest <code>ppem</code> value to optimize rendering quality. The <code>strikeselect</code> value is ignored when the font is loaded again. Default: 100
subsetlimit	(Float or percentage; ignored for Type 3 fonts) Disable automatic font subsetting if the percentage of glyphs used in the document related to the total number of glyphs in the font exceeds the specified percentage. Default: 100%
subsetminsize	(Float) Disable automatic TrueType/OpenType font subsetting if the size of the original font file is less than the specified value in KB. Default: 50
subsetting	(Boolean) Controls whether an embedded font will be subset. Subsetting for Type 3 fonts requires two-pass definition of the font (see PDFlib Tutorial). Default: subsetting is enabled automatically based on the <code>subsetlimit</code> and <code>subsetminsize</code> settings. For Type 3 fonts created from OpenType color fonts subsetting is enabled by default.
unicodemap	(Boolean; must not be set to false for PDF/A-1a/2a/2u/3a/3u and PDF/UA-1) Controls generation of To-Unicode CMaps. This option is ignored in Tagged PDF mode. Default: true

Table 4.1 Font options for `PDF_load_font()` and implicit font loading with option lists

option	description
vertical	(Boolean; only for TrueType and OpenType fonts; orced to true if the font name starts with @) If true, the font is prepared for vertical writing mode and the <code>vrt2</code> and <code>vert</code> OpenType features are activated if supported by the font.
xheight	(Integer between -2048 and 2048) See ascender above.

Table 4.2 Suboptions for the `fallbackfonts` option of `PDF_load_font()`

option	description				
font options	If the font is specified implicitly (i.e. via the fontname as opposed to the font option) all font options according to Table 4.1 except <code>fallbackfonts</code> can be supplied as suboptions.				
font	(Font handle) A font handle returned by a call to <code>PDF_load_font()</code> without the <code>fallbackfonts</code> option. If this option is supplied, all font loading options (including fontname and encoding) are ignored.				
fontsize	(Float or percentage) Size of the fallback font in user coordinates or as a percentage of the current font size. This option can be used to adjust the size of the fallback font if the design size of the fallback font doesn't match that of the base font. Default: 100%				
forcechars	(List of Unichars or Unicode ranges, or single keyword) Specify characters which are always rendered with glyphs from the fallback font (regardless of the <code>glyphcheck</code> setting). The fallback font must contain glyphs for the specified characters (if individual characters are specified), or at least glyphs for the first and last characters in the specified Unicode range. Unicode values can be specified for this option even if an 8-bit encoding has been specified for the base font. One of the following keywords can be supplied: <table border="0"> <tr> <td>gaji</td> <td>The fallback font must refer to a SING font, and this keyword can be used as a shortcut for the Unicode value of the main glyph of the SING font.</td> </tr> <tr> <td>all</td> <td>All glyphs in the fallback font will be used to replace the corresponding characters in the base font, even if the character is available in the base font.</td> </tr> </table>	gaji	The fallback font must refer to a SING font, and this keyword can be used as a shortcut for the Unicode value of the main glyph of the SING font.	all	All glyphs in the fallback font will be used to replace the corresponding characters in the base font, even if the character is available in the base font.
gaji	The fallback font must refer to a SING font, and this keyword can be used as a shortcut for the Unicode value of the main glyph of the SING font.				
all	All glyphs in the fallback font will be used to replace the corresponding characters in the base font, even if the character is available in the base font.				
textrise	(Float or percentage) Text rise value (see Table 4.6). Percentages are based on the size specified for the fallback font (i.e. after applying the <code>fontsize</code> suboption if present). This option can be used to adjust the position of text in the fallback font if the design size of the fallback font doesn't match that of the base font. Default: 0				

C++ Java C# **void close_font(int font)**

Perl PHP **close_font(int font)**

C **void PDF_close_font(PDF *p, int font)**

Close an open font handle which has not yet been used in the document.

font A font handle returned by `PDF_load_font()` which has not already been used in the document or closed.

Details This method closes a font handle, and releases all resources related to the font. The font handle must not be used after this call. Usually fonts will automatically be closed at the end of a document. However, closing a font is useful in the following situations:

- ▶ After querying font properties with `PDF_info_font()` it was determined that the font will not be used in the current PDF document.

- ▶ A font was retained across document boundaries (with the *keepfont* option of `PDF_load_font()`), but now it should be disposed because it is no longer required.

If the font has already been used in the current document it must not be closed.

Scope any

C++ Java C# **`double info_font(int font, String keyword, String optlist)`**

Perl PHP **`float info_font(int font, string keyword, string optlist)`**

C **`double PDF_info_font(PDF *p, int font, const char *keyword, const char *optlist)`**

Query detailed information about a loaded font.

font A font handle returned by `PDF_load_font()`, or -1 (in PHP: 0) for some keywords.

keyword A keyword specifying the requested information according to Table 4.4. The following keywords can be used:

- ▶ Keywords for glyph mapping: *cid, code, glyphid, glyphname, unicode*
- ▶ Font metrics: *ascender, capheight, descender, italicangle, linegap, xheight*
- ▶ Font file, name, and type: *cidfont, familyname, fontfile, fontname, fonttype, outlineformat, singfont, standardfont*
- ▶ Technical font information: *feature, featurelist, hostfont, kerningpairs, numglyphs, vertical*
- ▶ Keywords for Ideographic Variation Selectors: *maxuvsunicode, minuvsunicode, selector, selectorlist*
- ▶ Font/encoding relationship: *codepage, codepagelist, encoding, fallbackfont, keepnative, maxcode, numusableglyphs, replacementchar, symbolfont, unicodefont, unmappedglyphs*
- ▶ Results of font processing for the current document: *numusedglyphs, usedglyph, willembed, willsubset*
- ▶ Color font processing: *colormode, colortype, numcolorglyphs*

optlist An option list which additionally qualifies the selected keyword. The following options can be used:

- ▶ Keyword-specific options which are detailed along with the corresponding keyword in Table 4.4.
- ▶ Mapping options according to Table 4.3 for specifying glyphs: *cid, code, glyphid, glyphname, selector, unicode*.

These options define the source value for the mapping keywords *cid*, *code*, *glyphid*, *glyphname*, and *unicode*. The mapping options are mutually exclusive. The *code*, *glyphname*, and *unicode* options can be combined with the *encoding* option.

Table 4.3 Options for specifying glyphs in `PDF_info_font()`

option	description
<i>cid</i>	(Number) CID value of the glyph; only reasonable if <i>cidfont=1</i>
<i>code</i>	(Number in the range 0...255; only for fonts with 8-bit encoding) Encoding slot
<i>glyphid</i>	(Number in the range 0...65535) Internal glyph id
<i>glyphname</i>	(String) Name of a glyph; not reasonable if <i>cidfont=1</i>
<i>selector</i>	(Unichar) Unicode value of a variation selector in the range U+0xFE00..U+FE0F or U+E0100..U+E01EF. All values returned by the <i>selector</i> keyword can be supplied here.

Table 4.3 Options for specifying glyphs in `PDF_info_font()`

option	description
<code>unicode</code>	(Unichar) Unicode character

Returns The value of some font or encoding property as requested by *keyword* and in some cases auxiliary options. For unspecified combinations of keyword and options `-1` (in PHP: `o`) will be returned. If the requested keyword produces text, a string index is returned, and the corresponding string must be retrieved with `PDF_get_string()`.

Details This method supplies information from the following distinct sources:

- ▶ If a valid font handle is supplied it returns information gathered from the font. Examples: font metrics, name, or type; *unicode* value for a particular *glyphid*.
- ▶ If *font* = `-1` (in PHP: `o`) and the *encoding* option is supplied it returns information about this encoding. Example: *unicode* value for a *code* in the encoding.
- ▶ If *font* = `-1` (in PHP: `o`) and the *encoding* option is not supplied it returns information gathered from PDFlib's internal tables. Example: *unicode* value for a particular *glyphname*.

Scope any

Table 4.4 Keywords and options for `PDF_info_font()`

keyword	description
<i>ascender</i>	Metrics value for the ascender. Supported options (default: <code>fontsize=1000</code>): <i>faked</i> (Boolean) <code>1</code> if the value had to be estimated because it was not available in the font file, otherwise <code>o</code> <i>fontsize</i> (FontSize) Value will be scaled to the specified font size
<i>capheight</i>	Metrics value for the capheight. See <i>ascender</i> .
<i>cid</i>	CID for the specified glyph, or <code>-1</code> if not available. Supported options: <code>cid</code> , <code>glyphid</code> , <code>unicode</code> , <code>selector</code>
<i>cidfont</i>	<code>1</code> if the font will be embedded as a CID font, otherwise <code>o</code>
<i>code</i>	Number in the range <code>0...255</code> specifying an encoding slot or <code>-1</code> if no such slot was found in the font or in the encoding (if the <i>encoding</i> option was supplied and <i>font</i> = <code>-1</code> (in PHP: <code>o</code>)). Supported options are the mapping options <code>code</code> , <code>glyphid</code> , <code>glyphname</code> , <code>unicode</code> plus the following: <i>encoding</i> (String) Name of an 8-bit encoding
<i>codepage</i>	Check whether the font supports a specific codepage. The information will be taken from the OS/2 table of TrueType/OpenType fonts if available. Supported option: <i>name</i> (String; required) Name of a codepage in the form <code>cpXXXX</code> , where XXXX indicates the decimal number of a codepage (e.g. <code>cp437</code> , <code>cp1252</code>) The following return values indicate whether the specified codepage is supported by the font: <code>-1</code> Unknown because the font is not a TrueType or OpenType font. <code>o</code> Font does not support the specified codepage. <code>1</code> Font supports the specified codepage.
<i>codepagelist</i>	String index of a space-separated list of all codepages supported by the font (in the form <code>cpXXXX</code>), or <code>-1</code> if the codepage list is unknown because the font is not a TrueType or OpenType font (see <i>codepage</i>).
<i>colormode</i>	(String index) (String index; only relevant if <code>colortype</code> is different from <code>none</code>) Processing mode for glyphs in a color font. Returns a string index for one of the following strings: <code>ignorecolor</code> , <code>ignoremono</code> , <code>combined</code> . If the font is not an OpenType color font a string index for <code>ignorecolor</code> is returned.

Table 4.4 Keywords and options for `PDF_info_font()`

keyword	description
colortype	(String index) Indicates whether the font contains color glyphs and, if so, in which format. Returns a string index for one of the following strings: COLR, none, sbix, SVG
descender	Metrics value for the descender. See ascender.
encoding	String index of the name of the font's encoding. Supported options (default: actual): api (Boolean) If true, the encoding name as specified in the API actual (Boolean) If true, the name of the actual encoding used for the font
fallbackfont	Handle of the base or fallback font which will be used to render the character specified in the unicode option. This can be used to check which font in the chain of fallback fonts actually provides the glyph used for the specified character. If the character cannot be rendered with any of the base or fallback fonts, -1 will be returned. Supported option: unicode
familyname	String index of the name of the font family, or -1 if unavailable
feature	Check whether the font contains a specific OpenType feature which is supported by PDFlib. Supported options: language (Keyword; only if script is supplied) Language name. Default: <code>_none</code> name (Keyword; required) Four-character name of an OpenType feature, e.g. liga, ital, vert. Feature kern can not be queried. script (Keyword) Script name. Default: <code>_none</code> An exception is thrown if an unknown keyword for language, name, or script is supplied; see PDFlib Tutorial for lists of known keywords. The following return values indicate whether the specified OpenType feature is present in the font and supported by PDFlib: -1 No feature tables are available in the font. 0 The feature is not available for the specified script and language in the font, or is not known to PDFlib. 1 The feature is available for the specified script and language.
featurelist	String index of a space-separated list of all features which are available in the font and supported by PDFlib, or -1 if no feature tables are available.
fontfile	String index of the path name for the font file, or -1 if unavailable
fontname	String index of the font name, or -1 if unavailable. Supported options (default: <code>acrobat</code>): acrobat (Boolean) If true, the BaseFont entry in the PDF font descriptor api (Boolean) If true, the font name as specified in the API full (Boolean) If true, the FontName entry in the PDF font descriptor
fonttype	String index of the font type, or -1 if unavailable. Known font types are OpenType, TrueType, TrueType (CID), Type 1, Type 1 (CID), Type 1 CFF, Type 1 CFF (CID), Type 3
glyphid	Number in the range 0...65535 specifying the font-internal id (GID) of the specified glyph, or -1 if no such glyph was found. Supported options are the mapping options cid, code, glyphid, glyphname, unicode, selector.
glyphname	String index of the name of the specified glyph, or -1 if no such glyph was found in the font or in the specified encoding (if the encoding option was supplied and font=-1 (in PHP: 0)). Supported options are the mapping options code, glyphid, glyphname, unicode plus the following: encoding (String) Name of an 8-bit encoding
hostfont	1 if the font is a host font, 0 otherwise
italicangle	Italic angle of the font (ItalicAngle in the PDF font descriptor)
keepnative	(Deprecated) Resulting value of the keepnative option

Table 4.4 Keywords and options for `PDF_info_font()`

keyword	description
kernamount	Amount of kerning between the two glyphs identified with the options <code>gidright</code> and <code>gidleft</code> in font coordinates.
kerningpairs	Number of kerning pairs in the font
linegap	Metrics value for the linegap. See <code>ascender</code> .
maingid	Glyph ID of the main glyph (member <code>mainGID</code> of <code>SING</code> table).
maxcode	Highest code value for the font's encoding, in particular: <code>0xFF</code> for single-byte encodings, <code>numglyphs-1</code> for <code>encoding=glyphid</code> , and the highest Unicode value in the encoding otherwise.
maxvus-unicode	Largest Unicode value which may be contained in a valid Ideographic Variation Sequence (IVS).
minvus-unicode	Smallest Unicode value which may be contained in a valid Ideographic Variation Sequence (IVS).
numcolor-glyphs	(Integer) Number of color glyphs in a font created from an OpenType color font. The count may differ from the number of original color glyphs in the OpenType font since some glyphs may be used for more than one Unicode value, e.g. <code>HYPHEN-MINUS U+002D</code> vs. <code>SOFT HYPHEN U+00AD</code> .
numglyphs	Number of glyphs in the font (including the <code>.notdef</code> glyph). Since GIDs start at 0 the highest possible GID value is one smaller than <code>numglyphs</code> .
numusable-glyphs	Number of glyphs in the font which can be reached by the encoding supplied in <code>PDF_load_font()</code>
numused-glyphs	Number of glyphs used in generated text so far.
outlineformat	Font format; one of <code>PFA</code> , <code>PFB</code> , <code>TTF</code> , <code>OTF</code> . For <code>TTC</code> and <code>WOFF/WOFF2</code> fonts the underlying base font format is returned, e.g. <code>TTF</code> . For color fonts the underlying OpenType font format is returned.
replacementchar	Unicode value of the character specified in the <code>replacementchar</code> option. For symbolic fonts loaded with <code>encoding=builtin</code> the code will be returned instead of the Unicode value.
selector	Unicode value of the variation selector with the number specified in the <code>index</code> option. If the <code>index</code> option is not specified or the specified selector is not available in the font, <code>-1</code> is returned. Supported option: index (Non-negative Integer) Index of a selector.
selectorlist	String index of a string containing a space-separated list of the Unicode values of all variation selectors in the font. Each value is provided in the form <code>hhhhh</code> where <code>h</code> is a hexadecimal digit.
singfont	1 if the font is a <code>SING</code> (<code>gaiji</code>) font, otherwise 0
standardfont	1 if the font is a PDF core font otherwise 0
symbolfont	1 if the font is a symbolic font, 0 otherwise (<code>symbol</code> flag in the PDF font descriptor)
unicode	Unicode UTF-32 value for the specified glyph, or <code>-1</code> if no Unicode value was found in the font or encoding (if the encoding option was supplied and <code>font=-1</code> (in PHP: <code>o</code>)). Supported options are the mapping options <code>cid</code> , <code>code</code> , <code>glyphid</code> , <code>glyphname</code> , <code>unicode</code> , <code>selector</code> plus the following: encoding (String) Name of an 8-bit encoding
unicodelfont	1 if the font/encoding combination provides Unicode mapping for the glyphs, otherwise 0.
unmapped-glyphs	Number of glyphs in the font which are mapped to Unicode PUA values, regardless of whether the PUA value was already present in the font or has been assigned by <code>PDFlib</code> .
usedglyph	1 if the specified glyph ID was used in the text, otherwise 0. Supported option: <code>glyphid</code>
vertical	1 if the font is for vertical writing mode, otherwise 0

Table 4.4 Keywords and options for `PDF_info_font()`

keyword	description
weight	Font weight in the range 100...900; 400=normal, 700=bold
willembed	1 if the font will be embedded, otherwise 0
willsubset	1 if a font subset will be created, otherwise 0
xheight	Metrics value for the xheight. See ascender.

4.2 Text Filter and Appearance Options

In this section the term *text* designates content strings, i.e. text with a specified appearance (font, color, etc.). In contrast, name strings and hypertext strings (e.g. file names) don't have any appearance; see PDFlib Tutorial for details.

Text options can be used with `PDF_set_text_option()`, `PDF_fit/info_textline()`, `PDF_fill_textblock()` and `PDF_add/create_textflow()`. Text options also apply to table cells and text Blocks. The following groups of text options are available:

- ▶ text filter options according to Table 4.5;
- ▶ text appearance options according to Table 4.6;
- ▶ shaping and typographic options according to Table 5.4 (not for `PDF_set_text_option()`).

Table 4.5 Text filter options for `PDF_set_text_option()`, `PDF_fit/info_textline()`, `PDF_fill_textblock()` and `PDF_add/create_textflow()`

option	description
actualtext	(Boolean; only for <code>PDF_set_text_option()</code> , <code>PDF_fit_textline()</code> , and <code>PDF_fill_textblock()</code>) If true, PDFlib creates a marked content Span with suitable ActualText if the Unicode value(s) derived from the ToUnicode CMap wouldn't be correct and no user-provided Alt or ActualText was provided. This ensures correct text extraction for glyphs which are used to represent multiple similar looking Unicode values in a font, e.g. Ohm and Omega. Default: true
charref	(Boolean) If true, enable substitution of numeric and character entity references and glyph name references in content strings. ¹ Default: the global charref option
escape-sequence	(Boolean) If true, enable substitution of escape sequences in content strings. ¹ Default: the global escapesequences option
glyphcheck	(Keyword) Glyph checking policy: what happens if a code in the text cannot be mapped to a glyph in the selected font (default: the global glyphcheck option): <ul style="list-style-type: none">error An exception is thrown for unavailable glyphs. A detailed error message can be retrieved with <code>PDF_get_errmsg()</code>.none No checking. notdef glyphs trigger an exception in PDF/A, PDF/UA-1 or PDF/X-4/5 mode; otherwise notdef glyphs may appear in the output.replace Try to replace unavailable glyphs with typographically similar characters in the base and fallback fonts and decompose ligatures. If no suitable glyph could be found, the character is handled according to the replacementchar option.

Table 4.5 Text filter options for `PDF_set_text_option()`, `PDF_fit/info_textline()`, `PDF_fill_textblock()` and `PDF_add/create_textflow()`

option	description
normalize	(Keyword; ignored for encoding=glyhid) Normalize incoming text to one of the Unicode normalization forms (default: none):
none	Do not apply any normalization. This is the default behavior; the client is responsible for supplying text which can be represented with glyphs from the selected font.
nfc	Normalization Form C (NFC): canonical decomposition followed by canonical composition. NFC replaces combining sequences with precomposed characters. This is useful for workflows with combining sequences since fonts usually contain only glyphs for the precomposed character. Without NFC normalization PDFlib emits a sequence of multiple characters instead of the precomposed character.
nfkc	Normalization Form KC (NFKC): compatibility decomposition followed by canonical composition. This is useful for workflows which are only interested in the semantics of characters, but not in formatting differences, e.g. convert shaped Arabic characters to their base form, resolve ligatures and fractions, replace vertical forms with horizontal forms, wide characters with regular characters.
nfd	Normalization Form D (NFD): canonical decomposition
nfkd	Normalization Form KD (NFKD): compatibility decomposition
	Since NFD and NFKD can create combining sequences they are unlikely to be useful in PDFlib workflows.
textformat	(Keyword; only for C as well as Perl, PHP, Ruby with stringformat=legacy) Format used to interpret content strings. Supported keywords: bytes, utf8, ebcdicutf8 (only on IBM System i and IBM Z), utf16, utf16le, utf16be, and auto. ¹ Default: the global textformat option

1. The value may be overridden by a subsequent call to `PDF_set_option()` with the same option.

Table 4.6 Text appearance options for `PDF_set_text_option()`, `PDF_fit/info_textline()`, `PDF_fill_textblock()` and `PDF_add/create_textflow()`

option	description
charspacing	(Float or percentage) Character spacing, i.e. the shift of the current point after placing individual characters in a string. Float values specify units of the user coordinate system; percentages are based on fontsize. In order to spread characters apart use positive values for horizontal writing mode, and negative values for vertical writing mode. Default: 0
dasharray	(List of non-negative floats or keyword) The lengths of dashes and gaps for stroked (outline) text and decoration. The keyword none can be used to create solid lines. Default: none
decoration-above	(Boolean) If true, the text decoration enabled with the underline, strikeout, and overline options will be drawn above the text, otherwise below the text. Changing the drawing order affects visibility of the decoration lines, i.e. you can control whether the text overprints the lines or vice versa. Default: false
fakebold	(Boolean) If true, simulate bold text by stroking glyph outlines or multiple overprinting. Default: false
fillcolor	(Color) Fill color of the text. ¹ Default for simple text output methods and <code>PDF_fit_textline()</code> with <code>inittextstate=false</code> : the corresponding option in the current graphics state. Default for <code>Textflow</code> and <code>PDF_fit_textline()</code> with <code>inittextstate=true</code> : {gray 0} (in PDF/A mode: {lab 0 0 0})
font	(Font handle) Handle for the font to be used. If this option is supplied, all font loading options (including fontname and encoding) are ignored. Default: the implicitly loaded font if available, else the font selected with <code>PDF_setfont()</code> for simple text output and <code>PDF_fit_textline()</code> with <code>inittextstate=false</code> . Otherwise no font is available which will trigger an error.

Table 4.6 Text appearance options for `PDF_set_text_option()`, `PDF_fit/info_textline()`, `PDF_fill_textblock()` and `PDF_add/create_textflow()`

option	description
fontsize	(FontSize) Size of the font, measured in units of the current user coordinate system. In <code>PDF_fit_textline()</code> percentages relate to the box width (for <code>orientate=north</code> and <code>south</code>) or box height (for <code>orientate=east</code> and <code>west</code>). In <code>PDF_set_text_option()</code> and <code>Textflow</code> percentages relate to the size of the preceding text. Default: <code>PDF_setfont()</code> sets the default only for simple text output methods and <code>PDF_fit_textline()</code> with <code>inittextstate=false</code> . Otherwise no font size is available which will trigger an error.
gstate	(Option list with options for <code>PDF_create_gstate()</code> or <code>gstate handle</code>) Graphics state options or handle. The graphics state affects all text created with this method. Default: no graphics state, i.e. current settings are used
horizscaling	(Float or percentage; must be different from 0) Horizontal text scaling to the given percentage. Text scaling shrinks or expands the text by a given percentage. Text scaling always relates to the horizontal coordinate. Default: 100%
inittextstate	(Boolean; only for <code>PDF_fit_textline()</code>) If <code>true</code> all text appearance options are initialized with the default values. If <code>false</code> the current text state values are used. Default: <code>false</code>
italicangle	(Float; not supported for vertical writing mode) The italic (slant) angle of text in degrees (between -90° and 90°). Negative values can be used to simulate italic text when only a plain upright font is available, especially for CJK fonts. Default: 0
kerning	(Boolean) If <code>true</code> , enable kerning for fonts which have been opened with the <code>readkerning</code> option; disable kerning otherwise. ² Default: the global option <code>kerning</code>
leading	(Float or percentage) Specify the leading for multi-line text, i.e. the distance between baselines of adjacent lines of text as absolute value in user coordinates or as a percentage of <code>fontsize</code> . Setting the leading equal to the font size results in dense line spacing. However, ascenders and descenders of adjacent lines generally don't overlap (<code>leading=0</code> results in overprinting lines). Default: 100% The leading for <code>PDF_add/create_textflow()</code> is determined as follows: if there are option lists at the beginning of a line, the leading is determined by the last relevant option (<code>font</code> , <code>fontsize</code> , <code>leading</code> , etc.). If there are additional option lists on the same line, any options relevant for leading are only taken into account if <code>fixedleading=false</code> . If there are no option lists in the line, the previous leading value is used.
overline	(Boolean) If <code>true</code> , a line will be drawn above the text. Default: <code>false</code>
shadow	(Option list; only for <code>PDF_fit_textline()</code> , <code>PDF_fill_textblock()</code> , <code>PDF_add/create_textflow()</code>) Create a shadow effect for the text (default: no shadow): disable (Boolean; only for <code>PDF_add/create_textflow()</code>) If <code>true</code> , a previously specified shadow is disabled. Default: <code>false</code> fillcolor (Color) Fill color of the shadow. Default: <code>{gray 0.8}</code> gstate (Option list with options for <code>PDF_create_gstate()</code> or <code>gstate handle</code>) Graphics state options or handle which are applied to the shadow. Default: no graphics state, i.e. current settings are used offset (List of 2 floats or percentages) The shadow's offset from the reference point of the text in user coordinates or as a percentage of the font size. Default: <code>{5% -5%}</code> strokecolor (Color; only effective if <code>textrendering</code> is set to <code>stroke text</code>) Stroke color of the shadow. Default: current stroke color strokewidth (Float, percentage or keyword; only effective if <code>textrendering</code> is set to <code>stroke text</code>) Line width for outline text in the shadow (in user coordinates or as a percentage of the font size). The keyword <code>auto</code> or the equivalent value <code>0</code> uses a built-in default. Default: current stroke width if the main text is also set to <code>stroke text</code> , otherwise <code>auto</code> textrendering (Integer) Text rendering mode of the shadow. Default: current value of <code>textrendering</code>
strikeout	(Boolean) If <code>true</code> , a line will be drawn through the text; see also <code>decorationabove</code> . Default: <code>false</code>

Table 4.6 Text appearance options for `PDF_set_text_option()`, `PDF_fit/info_textline()`, `PDF_fill_textblock()` and `PDF_add/create_textflow()`

option	description		
strokecolor	(Color; only effective for stroked text, see <code>textrendering</code>) Stroke color of the text. Default: see <code>fillcolor</code>		
strokewidth	(Float, percentage, or keyword; only effective if <code>textrendering</code> is set to stroke text) Line width for outline text (in user coordinates or as a percentage of the fontsize). The keyword <code>auto</code> or the equivalent value <code>o</code> uses a built-in default. Default: <code>auto</code>		
tagtrailing-hyphen	(Unichar or keyword; only relevant for Tagged PDF) If the last character in the text (after possibly applying glyph replacements) is equal to the specified Unicode value, it will be tagged as Span with Actual-Text soft hyphen <code>U+00AD</code> if required by the font, and no autospace will be added. The keyword <code>none</code> results in no tagging for soft hyphens. Default: <code>U+00AD</code>		
textrendering	(Integer) Text rendering mode (default: <code>o</code>): <table border="0" style="width: 100%; margin-top: 10px;"> <tr> <td style="width: 50%; vertical-align: top;"> <p>0  fill text</p> <p>1  stroke text (outline)</p> <p>2  fill and stroke text</p> <p>3 invisible text</p> </td> <td style="width: 50%; vertical-align: top;"> <p>4  fill text and add it to the clipping path</p> <p>5  stroke text and add it to the clipping path</p> <p>6  fill and stroke text and add it to the clipping path</p> <p>7  add text to the clipping path</p> </td> </tr> </table> <p>Behavior of <code>textrendering=4/5/6/7</code> (clipping modes):</p> <ul style="list-style-type: none"> ▶ There is no clipping effect after <code>PDF_fit_textflow()</code>, <code>PDF_fit_table()</code>, <code>PDF_fill_textblock()</code> and <code>PDF_fit_textline()</code> if the <code>textpath</code> option is specified. ▶ Clipping areas can be accumulated across multiple calls to simple text output methods, but not across multiple calls to <code>PDF_fit_textline()</code>. ▶ <code>PDF_fit_textline()</code>: the specified <code>fillcolor</code> and <code>strokecolor</code> remain in effect after the method call. The text rendering mode doesn't have any effect on Type 3 fonts including those created from OpenType color fonts, except that <code>textrendering=3</code> and <code>7</code> result in invisible text. 	<p>0  fill text</p> <p>1  stroke text (outline)</p> <p>2  fill and stroke text</p> <p>3 invisible text</p>	<p>4  fill text and add it to the clipping path</p> <p>5  stroke text and add it to the clipping path</p> <p>6  fill and stroke text and add it to the clipping path</p> <p>7  add text to the clipping path</p>
<p>0  fill text</p> <p>1  stroke text (outline)</p> <p>2  fill and stroke text</p> <p>3 invisible text</p>	<p>4  fill text and add it to the clipping path</p> <p>5  stroke text and add it to the clipping path</p> <p>6  fill and stroke text and add it to the clipping path</p> <p>7  add text to the clipping path</p>		
textrise	(Float or percentage) Textrise value, which specifies the distance between the desired text position and the baseline. Positive values of <code>textrise</code> move the text up. Textrise always relates to the vertical coordinate. This may be useful for superscripts and subscripts. Percentages are based on fontsize. Default: <code>o</code>		
underline	(Boolean) If <code>true</code> , a line will be drawn below the text. Default: <code>false</code>		
underline-position	(Float, percentage, or keyword) Position of the stroked line for underlined text relative to the baseline (absolute values or relative to the fontsize; a typical value is <code>-10%</code>). The keyword <code>auto</code> specifies a font-specific value which is retrieved from the font. Default: <code>auto</code>		
underline-width	(Float, percentage, or keyword) Line width for underlined text (absolute value or percentage of the fontsize). The keyword <code>auto</code> or the value <code>o</code> uses a font-specific value from the font if available, otherwise <code>5%</code> . Default: <code>auto</code>		
wordspacing	(Float or percentage) Wordspacing, i.e. the shift of the current point after placing individual words in a line. In other words, the current point is moved horizontally after each space character (<code>U+0020</code>). The value is specified in user coordinates or a percentage of the fontsize. Default: <code>o</code>		

1. The value may be overridden by a subsequent call to `PDF_setcolor()` for simple text output methods and `PDF_fit_textline()` with `inittextstate=false`.
 2. The value may be overridden by a subsequent call to `PDF_set_option()` with the same option.

C++ Java C# **void set_text_option(String optlist)**

Perl PHP **set_text_option(string optlist)**

C **void PDF_set_text_option(PDF *p, const char *optlist)**

Set one or more text filter or text appearance options for simple text output methods.

optlist An option list specifying font and text options as follows:

- ▶ Text filter options according to Table 4.5:
actualtext, charref, escapesequence, glyphcheck, normalize, textformat
- ▶ Text appearance options according to Table 4.6:
charspacing, dasharray, decorationabove, fakebold, fillcolor, font, fontsize, gstate, horzscaling, inittextstate, italicangle, kerning, leading, overline, strikeout, strokecolor, strokewidth, tagtrailinghyphen, textrendering, textrise, underline, underlineposition, underlinewidth, wordspacing

Details The values of text options are relevant for all simple text output methods and *PDF_fit_textline()* with *inittextstate=false*. Calls to *PDF_set_text_option()* should not be mixed with calls to *PDF_setfont()* and *PDF_setcolor()*.

All text options are reset to their default values at the beginning of a page, pattern, template, or glyph description, and retain their values until the end of the current *page*, *pattern*, *template*, or *glyph* scope. However, the text options can also be reset with the *inittextstate* option.

Scope *page, pattern, template, glyph*

4.3 Simple Text Output

The methods listed in this section can be used for low-level text output. It is recommended to use the more powerful Textline and Textflow methods for more advanced text output (see Section 5.1, »Single-Line Text with Textlines«, page 89, and Section 5.2, »Multi-Line Text with Textflows«, page 96).

C++ Java C# **void PDF_setfont(int font, double fontsize)**
Perl PHP **setfont(int font, float fontsize)**
C **void PDF_setfont(PDF *p, int font, double fontsize)**

Set the current font in the specified size.

font A font handle returned by `PDF_load_font()`.

fontsize Size of the font, measured in units of the current user coordinate system. The font size must not be 0; a negative font size will result in mirrored text relative to the current transformation matrix.

Details This method sets the font and font size to be used by simple text output methods (e.g. `PDF_show()`) and `PDF_fit_textline()`. It is almost equivalent to a call to `PDF_set_text_option()` with the option list `font= fontsize=<fontsize>`. However, unlike `PDF_set_text_option()` this method additionally sets the *leading* text option to *fontsize*.

The font must be set on each page before calling any of the simple text output methods. Font settings are not retained across pages.

The use of `PDF_set_text_option()` is recommended over `PDF_setfont()`. Calls to `PDF_setfont()` should not be mixed with calls to `PDF_set_text_option()`.

Scope *page, pattern, template, glyph*

C++ Java C# **void set_text_pos(double x, double y)**
Perl PHP **set_text_pos(float x, float y)**
C **void PDF_set_text_pos(PDF *p, double x, double y)**

Set the position for simple text output on the page.

x, y New text position

Details The text position is set to the default value of (0, 0) at the beginning of each page. The current point for graphics output and the current text position are maintained separately.

Scope *page, pattern, template, glyph*

C++ Java C# **void show(String text)**

Perl PHP **show(string text)**

C **void PDF_show(PDF *p, const char *text)**

C **void PDF_show2(PDF *p, const char *text, int len)**

Print text in the current font and size at the current text position.

text (Content string) The text to be printed. In C *text* must not contain null bytes when using `PDF_show()`, since it is assumed to be null-terminated; use `PDF_show2()` for strings which may contain null characters.

len (C language binding only) Length of *text* (in bytes). If *len* = 0 a null-terminated string must be provided.

Details The font and font size must have been set before with `PDF_setfont()` or `PDF_set_text_option()`. The current text position is moved to the end of the printed text. The *char-spacing* parameter is taken into account after the last glyph.

Scope *page, pattern, template, glyph*

Bindings `PDF_show2()` is only available in C since in all other bindings arbitrary string contents can be supplied with `PDF_show()`.

C++ Java C# **void show_xy(String text, double x, double y)**

Perl PHP **show_xy(string text, float x, float y)**

C **void PDF_show_xy(PDF *p, const char *text, double x, double y)**

C **void PDF_show_xy2(PDF *p, const char *text, int len, double x, double y)**

Print text in the current font at the specified position.

text (Content string) The text to be printed. In C *text* must not contain null bytes when using `PDF_show_xy()`, since it is assumed to be null-terminated; use `PDF_show_xy2()` for strings which may contain null characters.

x, y The position in the user coordinate system where the text will be printed.

len (C language binding only) Length of *text* (in bytes). If *len* = 0 a null-terminated string must be provided.

Details The font and font size must have been set before with `PDF_setfont()` or `PDF_set_text_option()`. The current text position is moved to the end of the printed text. The *char-spacing* parameter is taken into account after the last glyph.

Scope *page, pattern, template, glyph*

Bindings `PDF_show_xy2()` is only available in C since in all other bindings arbitrary string contents can be supplied with `PDF_show_xy()`.

C++ Java C# **void continue_text(String text)**

Perl PHP **continue_text(string text)**

C **void PDF_continue_text(PDF *p, const char *text)**

C **void PDF_continue_textz(PDF *p, const char *text, int len)**

Print text at the next line.

text (Content string) The text to be printed. If this is an empty string, the text position will be moved to the next line anyway. In C *text* must not contain null bytes when using *PDF_continue_text()*, since it is assumed to be null-terminated; use *PDF_continue_textz()* for strings which may contain null bytes.

len (C language binding only) Length of *text* (in bytes). If *len = 0* a null-terminated string must be provided as in *PDF_continue_text()*.

Details The positioning of text (*x* and *y* position) and the spacing between lines is determined by the *leading* text option (which can be set with *PDF_set_text_option()*) and the most recent call to *PDF_show_xy()* or *PDF_set_text_pos()*. The current point will be moved to the end of the printed text; the *x* position for subsequent calls of this method will not be changed.

Scope *page, pattern, template, glyph*; this method should not be used in vertical writing mode.

Bindings *PDF_continue_textz()* is only available in C since in all other bindings arbitrary string contents can be supplied with *PDF_continue_text()*.

C++ Java C# **double stringwidth(String text, int font, double fontsize)**

Perl PHP **float stringwidth(string text, int font, float fontsize)**

C **double PDF_stringwidth(PDF *p, const char *text, int font, double fontsize)**

C **double PDF_stringwidthz(PDF *p, const char *text, int len, int font, double fontsize)**

Calculate the width of *text* in an arbitrary font.

text (Content string) The text for which the width will be queried. In C *text* must not contain null bytes when using *PDF_stringwidth()*, since it is assumed to be null-terminated; use *PDF_stringwidthz()* for strings which may contain null bytes.

len (C language binding only) Length of *text* (in bytes). If *len = 0* a null-terminated string must be provided.

font A font handle returned by *PDF_load_font()*.

fontsize Size of the font, measured in units of the user coordinate system.

Returns The width of *text* in a font which has been selected with *PDF_load_font()* and the supplied *fontsize*. The returned width value may be negative (e.g. when negative horizontal scaling has been set). In vertical writing mode the width of the widest glyph will be returned (use *PDF_info_textline()* to determine the actual height of the text). If character spacing has been specified, it will be applied after the last glyph as well (this behavior differs from *PDF_info_textline()*).

Details The width calculation takes into account the values of the following text options (which can be set with *PDF_set_text_option()*): *horzscaling*, *Kerning*, *charspacing*, and *wordspacing*.

Scope any except *object*

Bindings `PDF_stringwidth2()` is only available in C since in all other bindings arbitrary string contents can be supplied with `PDF_stringwidth()`.

4.4 User-defined Type 3 Fonts

```
C++ Java C# void begin_font(String fontname,  
                        double a, double b, double c, double d, double e, double f, String optlist)  
Perl PHP begin_font(string fontname, float a, float b, float c, float d, float e, float f, string optlist)  
C void PDF_begin_font(PDF *p, const char *fontname, int reserved,  
                     double a, double b, double c, double d, double e, double f, const char *optlist)
```

Start a Type 3 font definition.

fontname (Name string) The name under which the font will be registered, and can later be used with `PDF_load_font()`.

reserved (C language binding only) Reserved, must be 0.

a, b, c, d, e, f (Ignored in pass 2 for Type 3 font subsetting) Elements of the font matrix. This matrix defines the coordinate system in which the glyphs are drawn. The six values constitute a PDF matrix; $a*d$ must not be equal to $b*c$. The font matrix for a typical 1000 x 1000 coordinate system is `[0.001, 0, 0, 0.001, 0, 0]`.

optlist (Ignored in pass 2 for Type 3 font subsetting) Option list according to Table 4.7. The following options can be used: `defaultcmymk`, `defaultgray`, `defaultrgb`, `familynname`, `stretch`, `weight`, `widthonly`

Details The font may contain an arbitrary number of glyphs and can be used with `encoding=unicode` although Type 3 fonts require 8-bit addressing in PDF. The generated font can be used until the end of the enclosing `document` scope.

Type 3 fonts without subsetting are created in a single pass in which all parameters and options are honored. Subsetting for Type 3 fonts requires two passes:

- ▶ In pass 1 with the `widthonly` option of `PDF_begin_font()` only metrics and Unicode information is provided. Pass 1 defines only font and glyph metrics; the font matrix in `PDF_begin_font()` as well as `wx` and the glyph bounding box in `PDF_begin_glyph_ext()` must be supplied. Only `PDF_begin_glyph_ext()` and `PDF_end_glyph()` are required for each glyph, but not any other calls for defining the actual glyph shapes. If other methods are called between start and end of a glyph description, they are ignored.
- ▶ Pass 2 must be done after creating all text in the font. It defines the glyph outlines or bitmaps. Matrix values and options are ignored. After the last page has been created, PDFlib knows which glyphs have been used in the document and only embeds the required glyph descriptions to build the font subset. API method calls for the descriptions of unused glyphs are silently ignored. Methods which can return an error code (e.g. `PDF_load_image()`) return an error value which must be ignored by the application. In order to distinguish unused glyphs from real errors the error message number returned by `PDF_get_errnum()` must be checked: if it is zero, the glyph is ignored, i.e. no real error happened.

The same set of glyphs must be defined in both passes.

Scope any except `object`; this method starts `font` scope and must always be paired with a matching `PDF_end_font()` call.

Table 4.7 Options for `PDF_begin_font()` (Ignored in pass 2 for Type 3 font subsetting)

option	description
colorized	(Boolean; deprecated) Set the default of the <code>colorized</code> option of <code>PDF_begin_glyph_ext()</code> . Default: false
defaultgray defaultrgb defaultcmyk	(ICC handle or keyword) Set a default gray, RGB, or CMYK color space for the glyph descriptions in the font according to the supplied ICC profile handle. The option <code>defaultrgb</code> also supports the keyword <code>srgb</code> .
familyname	(String; PDF 1.5) Name of the font family
stretch	(Keyword; PDF 1.5) Font stretch value: <code>ultracondensed</code> , <code>extracondensed</code> , <code>condensed</code> , <code>semicondensed</code> , <code>normal</code> , <code>semiexpanded</code> , <code>expanded</code> , <code>extraexpanded</code> , <code>ultraexpanded</code> . Default: <code>normal</code>
weight	(Integer or keyword; PDF 1.5) Font weight: <code>100=thin</code> , <code>200=extralight</code> , <code>300=light</code> , <code>400=normal</code> , <code>500=medium</code> , <code>600=semibold</code> , <code>700=bold</code> , <code>800=extrabold</code> , <code>900=black</code> . Default: <code>normal</code>
widthonly	(Boolean) The value <code>true</code> initiates pass 1 for Type 3 font subsetting in which only font and glyph metrics are defined. No other API methods must be called between <code>PDF_begin_glyph_ext()</code> and <code>PDF_end_glyph()</code> . If other methods are called nevertheless, they will not have any effect on the PDF output and will not raise any exception. If <code>widthonly=false</code> pass 2 for Type 3 font subsetting is initiated so that the glyph outlines or bitmaps can be defined. Default: false

C++ Java C# **void end_font()**

Perl PHP **end_font()**

C **void PDF_end_font(PDF *p)**

Terminate a Type 3 font definition.

Scope *font*; this method terminates *font* scope, and must always be paired with a matching `PDF_begin_font()` call.

C++ Java C# **void begin_glyph_ext(int uv, String optlist)**

Perl PHP **begin_glyph_ext(int uv, string optlist)**

C **void PDF_begin_glyph_ext(PDF *p, int uv, const char *optlist)**

Start a glyph definition for a Type 3 font.

uv Unicode value of the glyph. Each Unicode value can be supplied only for one glyph description. The glyph with the Unicode value 0 is always assigned glyph ID 0 and glyph name `.notdef`, regardless of whether or not the glyph was specified.

If `uv=-1` the Unicode value is derived from the `glyphname` option according to PDFlib's internal glyph name list. If a glyph name is unknown, consecutive PUA values (starting at U+E000) are assigned. This value can be queried with `PDF_info_font()`.

optlist (Ignored in pass 2 for Type 3 font subsetting) Option list according to Table 4.8. The following options can be used: `boundingbox`, `colorized`, `glyphname`, `width`

Details Glyph descriptions can be defined using text, graphics, and image methods. This method resets all text, graphics, and color state parameters to their default values.

By default, glyph descriptions can specify only the shape of the glyph, but not its color. The glyph will be colored with the current fill color when it is placed. The glyph description must also avoid all operations for setting transparency and must not place

images other than masks. If the *colorized* option is *true*, these restrictions are lifted and the glyph description can specify color and transparency in addition to glyph shape.

Scope *font*; this method starts *glyph* scope, and must always be paired with a matching *PDF_end_glyph()* call. If *widthsonly=true* in *PDF_begin_font()* all API calls between *PDF_begin_glyph_ext()* and *PDF_end_glyph()* are ignored.

Table 4.8 Options for *PDF_begin_glyph_ext()*

option	description
bounding-box	(List of 4 floats; optional, but recommended) Coordinates of the lower left and upper right corners of the glyph's bounding box. Default: {0 0 0 0}
colorized	(Boolean) If <i>true</i> , the glyph description may specify color and transparency. If <i>false</i> , the glyph can specify only the shape and will be drawn with the current fill or stroke color when it is placed. Default: the <i>colorized</i> option of <i>PDF_begin_font()</i> if specified, otherwise <i>false</i>
glyphname	(String; required if <i>uv=-1</i> ; ignored if <i>uv=0</i>) Unique name of the glyph. It should correspond to the supplied Unicode value. The name of glyph <i>o</i> with Unicode <i>o</i> is forced to <i>.notdef</i> . Default: AGL name according to the supplied Unicode value
width	(Float; required) Width of the glyph in the glyph coordinate system as specified by the font matrix.

C++ Java C# **void end_glyph()**

Perl PHP **end_glyph()**

C **void PDF_end_glyph(PDF *p)**

Terminate a glyph definition for a Type 3 font.

Scope *glyph*; this method changes from *glyph* scope to *font* scope, and must always be paired with a matching *PDF_begin_glyph_ext()* call.

4.5 User-defined 8-Bit Encodings

C++ Java C# `void encoding_set_char(String encoding, int slot, String glyphname, int uv)`

Perl PHP `encoding_set_char(string encoding, int slot, string glyphname, int uv)`

C `void PDF_encoding_set_char(PDF *p, const char *encoding, int slot, const char *glyphname, int uv)`

Add a glyph name and/or Unicode value to a custom 8-bit encoding.

encoding Name of the encoding. This is the name which must be used with `PDF_load_font()`. It must be different from any built-in encoding and all previously used encodings.

slot The position of the character to be defined, with $0 \leq \text{slot} \leq 255$. A particular slot must only be filled once within a given encoding.

glyphname The character's name

uv The character's Unicode value

Details This method is only required for specialized applications which must work with non-standard 8-bit encodings. It can be called multiply to define up to 256 character slots in an encoding. More characters may be added to a particular encoding until it has been used for the first time. Not all code points must be specified; undefined slots are filled with `.notdef` and U+0000.

It is strongly recommended to supply each glyph name/Unicode value only once in an encoding (with the exception of `.notdef/U+0000`). If slot 0 is used, it should contain the `.notdef` character.

The defined encoding can be used until the end of the current *object* scope.

Scope *any*

5 Text and Table Formatting

API methods in this chapter:

- ▶ `PDF_fit_textline()`
- ▶ `PDF_info_textline()`
- ▶ `PDF_add_textflow()`
- ▶ `PDF_create_textflow()`
- ▶ `PDF_fit_textflow()`
- ▶ `PDF_info_textflow()`
- ▶ `PDF_delete_textflow()`
- ▶ `PDF_add_tablecell()`
- ▶ `PDF_fit_table()`
- ▶ `PDF_info_table()`
- ▶ `PDF_delete_table()`

5.1 Single-Line Text with Textlines

C++ Java C# `void fit_textline(String text, double x, double y, String optlist)`

Perl PHP `fit_textline(string text, float x, float y, string optlist)`

C `void PDF_fit_textline(PDF*p, const char *text, int len, double x, double y, const char *optlist)`

Place a single line of text at position (x, y) subject to various options.

text (Content string) The text to be placed on the page.

len (C language binding only) Length of *text* (in bytes). If *len* = 0 a null-terminated string must be provided.

x, y The coordinates of the reference point in the user coordinate system where the text will be placed, subject to various options. See Section 6.1, »Object Fitting«, page 123, for a description of the fitting algorithm.

optlist An option list specifying font, text, and formatting options. The following options are supported:

- ▶ General option: *errorpolicy* (see Table 1.5)
- ▶ Font loading options according to Table 4.1 for implicit font loading (i.e. *font* option in the text appearance group not supplied):
ascender, autosubsetting, capheight, colormode, descender, embedding, encoding, fallbackfonts, fontname, keepnative, linegap, metadata, readfeatures, replacementchar, subsetlimit, subsetminsize, subsetting, unicodemap, vertical, xheight
- ▶ Text filter options according to Table 4.5:
actualtext, charref, escapesequence, glyphcheck, normalize, textformat
- ▶ Text appearance options according to Table 4.6:
charspacing, dasharray, decorationabove, fakebold, fillcolor, font, fontsize, gstate, horzscaling, inittextstate, italicangle, kerning, leading, overline, shadow, strikeout, strokecolor, strokewidth, textrendering, textrise, underline, underlineposition, underlinewidth, wordspacing

- ▶ Options for Textline formatting according to Table 5.1:
justifymethod, leader, textpath, xadvancelist
- ▶ Shaping and typographic options according to Table 5.4:
features, language, script, shaping
- ▶ Fitting options according to Table 6.1:
alignchar, blind, boxsize, fitmethod, margin, matchbox, orientate, position, rotate, stamp, showborder, shrinklimit
- ▶ Option for abbreviated structure element tagging according to Table 14.2 (only allowed in *page* scope): *tag*

Details If *inittextstate=false* (which is the default), the current text and graphics state options are used to control the appearance of the text output unless they are explicitly overridden by options.

If *inittextstate=true* the default values of the text and graphics state options are used to control the appearance of the text output unless they are explicitly overridden by options. The Textline options will not affect any output created after this call to *PDF_fit_textline()*.

The current text and graphics state are not modified by this method (in particular, the current font will be unaffected). However, the *textx/texty* options are adjusted to point to the end of the generated text output. The *charspacing* parameter is not taken into account after the last glyph.

The reference point for *PDF_continue_text()* is not set to the beginning of the text. In order to use *PDF_continue_text()* after *PDF_fit_textline()* you must query the starting point with *PDF_info_textline()* and the *startx/starty* keywords and set the text position with *PDF_set_text_pos()*.

Scope *page, pattern, template, glyph*

Table 5.1 Additional options for *PDF_fit_textline()*

option	description
justifymethod	(List of keywords; only relevant for <i>fitmethod=auto</i> and <i>stamp=none</i> ; requires <i>boxsize</i> ; ignored in vertical writing mode) Ensure that the text will not extend beyond the fitbox by applying one or more formatting methods without changing the fontsize. One or more of the following keywords can be supplied; if multiple keywords are present justification will be applied in the following order of decreasing priority: <i>wordspacing, charspacing, horizscaling</i> (default: <i>none</i>): charspacing Justify with an appropriate <i>charspacing</i> value. horizscaling Justify with an appropriate <i>horizscaling</i> value. none No justification wordspacing Justify with an appropriate <i>wordspacing</i> value. If the text does not contain any space characters <i>wordspacing</i> justification will not be applied.
leader	(Option list; ignored if <i>boxsize</i> is not specified or the width of the box is 0) Filler text (e.g. dot leaders) and formatting options. Leaders will be inserted repeatedly between the border of the text box and the text. See Table 5.3 for a list of supported suboptions. Default: no leader

Table 5.1 Additional options for `PDF_fit_textline()`

option	description
textpath	<p>(Option list) Draw text along a path. Text beyond the end of the path will not be displayed. If <code>showborder=true</code> the flattened path will be drawn with the current linewidth and stroke color. The options listed in Table 5.2 plus the following options of <code>PDF_draw_path()</code> are supported:</p> <p><code>align</code>, <code>attachmentpoint</code>, <code>boxsize</code>, <code>fitmethod</code>, <code>orientate</code>, <code>scale</code> (see Table 6.1)</p> <p><code>close</code>, <code>round</code>, <code>subpaths</code> (see Table 7.7)</p> <p><code>bboxexpand</code>, <code>boundingbox</code> (see Table 7.7)</p> <p>The following options of <code>PDF_fit_textline()</code> have modified meaning for text on a path:</p> <p>matchbox A separate box will be created for each glyph.</p> <p>position The first value specifies the starting position of the text relative to the length of the path (<code>left/center/right</code>). If the text is longer than the path it will always begin at <code>startoffset</code>. The second value specifies the vertical position of each glyph relative to the path, i.e. which part of the glyph box will touch the path (<code>bottom/center/top</code>).</p> <p>rotate Rotation angle for each glyph.</p> <p>The following fitbox-related options are ignored:</p> <p><code>boxsize</code>, <code>margin</code>, <code>fitmethod</code>, <code>orientate</code>, <code>alignchar</code>, <code>showborder</code>, <code>stamp</code>, <code>leader</code></p> <p>Kerning and text with CJK legacy encodings are not supported for text on a path.</p>
xadvancelist	<p>(List of floats; ignored if <code>shaping=true</code>) Advance width of the glyphs in the text in user coordinates. The length of the list must be less than or equal to the number of glyphs in the text. The <code>xadvance</code> values override the standard glyph widths. In most situations modified spacing can easier be achieved with the <code>charspacing</code> option.</p>

Table 5.2 Additional suboptions for the `textpath` option of `PDF_fit_textline()`

option	description
path	<p>(Path handle; required) Path to use as baseline for text output. By default, the text will be placed at the left side of the path and the path will serve as the text baseline. However, if the second keyword in the <code>position</code> option is <code>top</code> the text will be placed on the other side of the path and the top of the text will touch the path. The parameters <code>x</code> and <code>y</code> of <code>PDF_fit_textline()</code> are used as reference point for the path.</p>
startoffset	<p>(Float or percentage) Offset of the starting point of the text along the path in user coordinates or as percentage of the path length. Default: 0</p>
tolerance	<p>(Float or percentage) Indicates how much the last glyph on the path is allowed to extend beyond the path. The value is specified in user coordinates or as a percentage of the fontsize. Default: 25%</p>

Table 5.3 Suboptions for the leader option for `PDF_fit_textline()` and `PDF_add/create_textflow()` and inline options in `PDF_create_textflow()`

option	description
font loading options	If the font is specified implicitly (i.e. via the fontname option as opposed to the font option) all font loading options according to Table 4.1 can be supplied as suboptions.
alignment	(One or two keywords) Textline: The first keyword specifies the alignment of the leader between the left border of the fitbox and the Textline; the second keyword specifies the alignment of the leader between the Textline and the right border of the fitbox. If only one keyword is specified it will be used for the leader between the Textline and the right border of the fitbox. Supported keywords (default for Textline: {none grid}; default for Textflow: grid): <ul style="list-style-type: none"> center Textline: the leader is justified between the Textline and the border of the fitbox. Textflow: the leader is centered between the last text fragment (or the start of the line if there is no text) and the tab position (or the end of the line if there is no tab). grid PDFlib snaps the position of the leader text to the next multiple of one half of the width of the leader text to the left or right of the Textline. This may result in a gap between the Textline and the leader text which spans at most 50% of the width of the leader text. justify Textline: the leader is justified between the Textline and the border of the fitbox by applying a suitable character spacing. Textflow: the leader is justified between the last text fragment (or the start of the line if there is no text) and the tab position (or the end of the line if there is no tab) by applying a suitable character spacing. left The leader is repeated starting from the left border of the fitbox or the end of the Textline, respectively. This may result in a gap at the start of the Textline or the right border of the fitbox, respectively. none No leader right The leader is repeated starting from the right border of the fitbox or the beginning of the Textline, respectively. This may result in a gap at the end of the Textline or the left border of the fitbox, respectively.
fillcolor	(Color) Color of the leader. Default: color of the text line
font	(Font handle) Handle for the font to be used for the leader. Default: font of the text line
fontsize	(Fontsize) Size of the leader. Default: font size of the Textline
text	(Content string) The text which will be used for the leader. Default: U+002E '.' (period)
yposition	(Float or keyword) Vertical position of the leader relative to the baseline as a numerical value or as one of the keywords fontsize, ascender, xheight, baseline, descender, textrise. Default: baseline

C++ Java C# **double** `info_textline(String text, String keyword, String optlist)`

Perl PHP **float** `info_textline(string text, string keyword, string optlist)`

C **double** `PDF_info_textline(PDF *p, const char *text, int len, const char *keyword, const char *optlist)`

Perform Textline formatting without creating output and query the resulting metrics.

text (Content string) The contents of the Textline.

len (C language binding only) The length of text in bytes, or 0 for null-terminated strings.

Table 5.4 Shaping and typographic options for `PDF_fit/info_textline()`, `PDF_add/create_textflow()`, and `PDF_fill_textblock()`

option	description
features	<p>(List of keywords) Specifies which typographic features of an OpenType font are applied to the text, subject to the script and language options. Features which are not present in the font are ignored:</p> <p>_none Apply none of the features in the font.</p> <p><name> Enable a feature by supplying its four-character OpenType name. Common feature names are <code>aalt</code>, <code>liga</code>, <code>ital</code>, <code>tnum</code>, <code>smcp</code>, <code>swsh</code>, <code>zero</code>. A list with names and descriptions of all supported features can be found in the PDFlib Tutorial.</p> <p>no<name> The prefix <code>no</code> in front of a feature name (e.g. <code>no liga</code>) disables this feature.</p> <p>The following features are enabled by default: <code>calt</code>, <code>ccmp</code>, <code>clig</code>, <code>liga</code>, <code>locl</code>. The features <code>vrt2</code> and <code>vert</code> are automatically enabled for fonts in vertical writing mode.</p>
language	<p>(Keyword; only relevant if script is supplied) The text is processed according to the specified language which is relevant for the features and shaping options. A full list of keywords can be found in the PDFlib Tutorial, e.g. <code>ARA</code> (Arabic), <code>JAN</code> (Japanese), <code>HIN</code> (Hindi). Default: <code>_none</code> (undefined language)</p>
script	<p>(Keyword; required if shaping=true) The text is processed according to the specified script which is relevant for the options features, shaping, and advancedlinebreak (default: <code>_none</code>):</p> <p>_auto (Only relevant for shaping) Selects the script to which the majority text belongs, where <code>latn</code> and <code>_none</code> are ignored.</p> <p>_none Undefined script.</p> <p>The following scripts are supported for complex script shaping:</p> <p>arab, hebr Middle East scripts</p> <p>sinh, tibt Central Asian scripts</p> <p>khmr, thai Southeast Asian scripts</p> <p>hang, hani East Asian scripts</p> <p>beng, deva, gujr, knnda, mlym, orya, guru, taml, telu Indic scripts</p> <p>Additional script tags according to the OpenType specification, e.g. <code>latn</code>, <code>cyrl</code>, <code>grek</code>, can be used to control OpenType feature processing. A list of supported script keywords can be found in the PDFlib Tutorial.</p>
shaping	<p>(Boolean) If <code>true</code>, complex script shaping and bidirectional reordering are applied to the text according to the script and language options. The script option must have a value different from <code>_none</code> and the font must obey certain conditions (see PDFlib Tutorial). Shaping is not available for right-to-left text in Textflows (only in Textlines). Default: <code>false</code></p>

keyword A keyword specifying the requested information:

- ▶ Keywords for querying the results of object fitting according to Table 6.3: `boundingbox`, `fitscalex`, `fitscaley`, `height`, `objectheight`, `objectwidth`, `width`
- ▶ Additional keywords according to Table 5.5: `angle`, `ascender`, `capheight`, `descender`, `endx`, `endy`, `missingglyphs`, `pathlength`, `perpendiculardir`, `replacedchars`, `righttoleft`, `scriptlist`, `startx`, `starty`, `textwidth`, `textheight`, `unmappedchars`, `wellformed`, `writingdirx`, `writingdiry`, `xheight`

optlist An option list specifying options for `PDF_fit_textline()`. Options which are not relevant for the requested keyword are silently ignored.

Returns The value of some text metric value as requested by *keyword*.

Details This method will perform all calculations required for placing the text according to the supplied options, but will not actually create any output on the page. The text reference position is assumed to be {0 0}.

If *errorpolicy=return* this method returns 0 in case of an error. If *errorpolicy=exception* this method throws an exception in case of an error (even for the keyword *wellformed*).

Scope any except *object*

Table 5.5 Keywords for *PDF_info_textline()*

keyword	description
angle	Rotation angle of the baseline in degree, i.e. the text rotation
ascender capheight descender	Corresponding typographic value in user coordinates
endx, endy	x/y coordinates of the logical text end position in user coordinates
missingglyphs	(Only for text on a path) Number of glyphs which couldn't be placed on the path.
pathlength	(Only for text on a path) Length of the path covered by the text from its starting point to the end point. This value can be queried even if <i>PDF_fit_textline()</i> was called in blind mode. The value can be used for the <i>startoffset</i> option of <i>PDF_fit_textline()</i> to continue labeling a path with additional text.
perpendiculardir	Unit vector perpendicular to <i>writingdir</i> ; for standard horizontal text this would be (0, 1), for vertical text (1, 0)
replacedchars	Number of characters which have been replaced with a slightly different glyph from the internal list of typographically similar characters or with a glyph from a fallback font because they couldn't be mapped to a code in the current encoding or to a glyph in the font. This value can only be different from 0 if <i>glyphcheck=replace</i> .
righttoleft	1 if the global output direction for the text is right-to-left, and 0 for left-to-right or vertical text. The global direction will be determined based on the initial characters and any directional markers which may be present in the text (e.g. U+202D or &LRO; LEFT-TO-RIGHT OVERRIDE).
scriptlist	String containing the space-separated list of the names of all scripts in the text. This may be useful to prepare text shaping. The script names are sorted by frequency in descending order. The scripts <i>_none</i> and <i>_latn</i> are ignored since they are not relevant for shaping. If only <i>_none</i> and <i>_latn</i> characters are present in the text, -1 will be returned.
startx, starty	x/y coordinates of the logical text start position in the user coordinate system
textwidth, textheight	Width and height of the text. The height is subject to the <i>matchbox</i> definition of <i>boxheight</i> , which defaults to { <i>capheight none</i> }. For text on a path both values are 0.
unknownchars	If <i>glyphcheck=none</i> : number of skipped characters. The number includes character references which couldn't be resolved, and characters which couldn't be mapped to a code in the current encoding or to a glyph in the font. If <i>glyphcheck=replace</i> : number of characters which were replaced with the specified replacement character (option <i>replacementchar</i>). The number includes characters which couldn't be mapped to a code in the current encoding or to a glyph in the font, and characters which couldn't be replaced with typographically similar characters.
unmappedchars	The number of characters which have been skipped or replaced, i.e. the sum of <i>replacedchars</i> and <i>unknownchars</i> .

Table 5.5 Keywords for `PDF_info_textline()`

keyword	description
wellformed	1 if the text is well-formed according to the selected font/encoding (and <code>textformat</code> , if applicable), otherwise 0.
writingdirx writingdiry	<i>x/y</i> coordinates of the dominant writing direction (i.e. the direction of inline text progression) which describes a unit vector from (startx, starty) to (endx, endy). For left-to-right horizontal text the values will be (1, 0), for vertical text (0, -1), and for right-to-left horizontal text (-1, 0). The writing direction will be determined based on the <code>shaping</code> and <code>vertical</code> options as well as the directionality properties of the text.
xheight	<i>xheight</i> in user coordinates

5.2 Multi-Line Text with Textflows

C++ Java C# `int add_textflow(int textflow, String text, String optlist)`

Perl PHP `int add_textflow(int textflow, string text, string optlist)`

C `int PDF_add_textflow(PDF *p, int textflow, const char *text, int len, const char *optlist)`

Create a Textflow object, or add text and explicit options to an existing Textflow.

textflow Textflow handle returned by an earlier call to `PDF_create_textflow()` or `PDF_add_textflow()`, or -1 (in PHP: 0) to create a new Textflow.

text (Content string) The contents of the Textflow. The text may contain inline options if the `inlineoptions` option is set. The text shouldn't be empty.

len (C language binding only) The length of text in bytes, or 0 for null-terminated strings.

optlist An option list specifying Textflow options as follows:

- ▶ General option: `errorpolicy` (see Table 1.5)
- ▶ Font loading options according to Table 4.1 for implicit font loading (i.e. `font` option in the text appearance group not supplied):
`ascender, autosubsetting, capheight, colormode, descender, embedding, encoding, fallbackfonts, fontname, fontstyle, keepnative, linegap, metadata, readfeatures, replacementchar, subsetlimit, subsetminsize, subsetting, unicodemap, xheight`
- ▶ Text filter options according to Table 4.5:
`charref, escapesequence, glyphcheck, normalize, textformat`
- ▶ Text appearance options according to Table 4.6:
`charspacing, dasharray, decorationabove, fakebold, fillcolor, font, fontsize, gstate, horzscaling, inittextstate, italicangle, kerning, leading, overline, shadow, strikeout, strokecolor, strokewidth, textrendering, textrise, underline, underlineposition, underlinewidth, wordspacing`
- ▶ Shaping and typographic options according to Table 5.4:
`features, language, script, shaping`
- ▶ Options for Textflow formatting according to Table 5.6:
`alignment, avoidemptybegin, fixedleading, hortabmethod, hortabsize, lastalignment, leader, leftindent, minlinecount, parindent, rightindent, ruler, tabalignment`
- ▶ Options for controlling the line break algorithm according to Table 5.7:
`adjustmethod, advancedlinebreak, avoidbreak, locale, maxspacing, minspacing, nofitlimit, shrinklimit, spreadlimit`
- ▶ Command options according to Table 5.8:
`comment, inlineoptions, mark, matchbox, nextline, nextparagraph, restore, resetfont, return, save, space`
- ▶ Text semantics options according to Table 5.9:
`charclass, charmapping, hyphenchar, tabalignchar`
- ▶ Tagging options according to Table 5.10: `tagbegin, tagend`

Returns A Textflow handle which can be used in Textflow-related method calls. The handle is valid until the end of the enclosing `document` scope, or until `PDF_delete_textflow()` is called with this handle.

If the *textflow* parameter is -1 (in PHP: 0), a new Textflow will be created and its handle will be returned. Otherwise the handle supplied in the *textflow* parameter will be returned. By default, this method returns -1 (in PHP: 0) in case of an error. However, this behavior can be changed with the *errorpolicy* option. In case of an error the handle supplied in the *textflow* parameter can no longer be used in subsequent method calls (except in *PDF_delete_textflow()* if it was different from -1).

Details This method processes the supplied text and creates an internal data structure from it. It determines text portions (e.g. words) which will later be used by the formatter, converts the text to Unicode if possible, determines potential line breaks, and calculates the width of text portions based on font and text options.

As opposed to *PDF_create_textflow()*, which expects all text contents and options in a single call, this method is useful for supplying the text contents and options for a Textflow in separate calls. It will add the supplied *text* and *optlist* to a new or existing Textflow. Options specified in *optlist* are evaluated before processing *text*.

If *textflow=-1* (in PHP: 0) this method is almost equivalent to *PDF_create_textflow()*. However, unlike *PDF_create_textflow()* this method will not search for inline options in *text*. It is therefore not necessary to redefine the start character for inline option lists or to specify the length of the text with an inline option (not even for non-Unicode text and UTF-16 text).

This method preprocesses the supplied text and options, but does not create any output in the generated PDF document, but only prepares the text. Use *PDF_fit_textflow()*, *PDF_fit_table()*, or *PDF_fill_textblock()* to create output with the preprocessed Textflow handle.

By default, a new line will be forced by the characters U+000B (VT), U+2028 (LS), U+000A (LF), U+000D (CR), CRLF, U+0085 (NEL), U+2029 (PS), and U+000C (FF). These control characters will not be interpreted for symbolic fonts loaded with *encoding=builtin*. All of these except VT and LS force a new paragraph (which means that the *parindent* option will be effective). FF immediately stops the process of fitting text to the current fitbox (the method *PDF_fit_textflow()* returns the string *nextpage*).

A horizontal tab character (HT) sets a new start position for subsequent text. The details of this are controlled by the *hortabmethod* and *hortabszise* options.

Soft hyphen characters (SHY) will be replaced with the character specified in the *hyphenchar* option if there is a line break after the soft hyphen.

Vertical writing mode is not supported.

Scope any except *object*

Table 5.6 Additional formatting options for *PDF_add/create_textflow()*, inline options in *PDF_create_textflow()* and *PDF_fill_textblock()* with a Textflow Block

option	description
alignment	(Keyword) Formatting for lines in a paragraph (default: left):
left	Left-aligned, starting at <i>leftindent+parindent</i> (for the first line of a paragraph) and at <i>leftindent</i> (for all other lines)
center	Centered between <i>leftindent</i> and <i>rightindent</i>
right	Right-aligned, ending at <i>rightindent</i>
justify	Left- and right-aligned
	The value <i>alignment=justify</i> is ignored for a line containing the <i>nextline</i> option. The alignment of a line containing <i>nextparagraph</i> is not controlled by the <i>alignment</i> option, but rather the option <i>last-alignment</i> .

Table 5.6 Additional formatting options for `PDF_add/create_textflow()`, inline options in `PDF_create_textflow()` and `PDF_fill_textblock()` with a Textflow Block

option	description
avoid-emptybegin	(Boolean) If true, empty lines at the beginning of a fitbox will be deleted. Default: false
fixedleading	(Boolean) If true, the first leading value found in each line will be used. Otherwise the maximum of all leading values in the line will be used. <code>fixedleading</code> will be forced to true if the wrap option of <code>PDF_fit_textflow()</code> or the <code>createrwrapbox</code> suboption of the <code>matchbox</code> option will be used to wrap the text around shapes. Default: false
hortabmethod	(Keyword) Treatment of horizontal tabs in the text. If the calculated position is to the left of the current text position, the tab is ignored (default: relative): relative The position will be advanced by the amount specified in <code>hortabsize</code> . typewriter The position will be advanced to the next multiple of <code>hortabsize</code> . ruler The position will be advanced to the n-th tab value in the ruler option, where n is the number of tabs found in the line so far. If n is larger than the number of tab positions the relative method will be applied.
hortabsize	(Float or percentage) Width of a horizontal tab ¹ . The interpretation depends on the <code>hortabmethod</code> option. Default: 7.5%
lastalignment	(Keyword) Formatting for the last line in a paragraph. All keywords of the alignment option are supported, plus the following (default: auto): auto Use the value of the alignment option if it is different from justify, else left
leader	(Option list) Filler text (e.g. dot leaders) and formatting options. Leaders will be inserted until the next tab position, or the end of the line if no tab is available. Leaders never span more than one line. See Table 5.3 for a list of supported suboptions. Default: no leader
leftindent	(Float or percentage) Left indent of text lines ¹ . If <code>leftindent</code> is specified within a line and the resulting position is to the left of the current text position, this option is ignored for this line. Default: 0
minlinecount	(Integer) Minimum number of lines in the last paragraph of the fitbox. If there are fewer lines they will be placed in the next fitbox. The value 2 can be used to prevent single lines of a paragraph at the end of a fitbox («orphans»). Default: 1
parindent	(Float or percentage) Left indent of the first line of a paragraph ¹ . The amount is added to <code>leftindent</code> . Specifying this option within a line will act like a tab. Default: 0
rightindent	(Float or percentage) Right indentation of text lines ¹ . Default: 0
ruler	(List of floats or percentages) List of absolute tab positions for <code>hortabmethod=ruler</code> ¹ . The list may contain up to 32 non-negative entries in ascending order. Default: integer multiples of <code>hortabsize</code>
tabalignment	(List of keywords; only with <code>hortabmethod=ruler</code>) Alignment for tab stops. The list may contain up to 32 entries. If more than 32 horizontal tabs per line occur in the text the list will be extended with the last value. Each entry in the list defines the alignment for the corresponding entry in the ruler option. Default: left. center Text will be centered at the tab position. decimal The first instance of <code>tabalignchar</code> will be left-aligned at the tab position. If no <code>tabalignchar</code> is found, right alignment will be used instead. left Text will be left-aligned at the tab position. right Text will be right-aligned at the tab position.

1. In user coordinates or as a percentage of the width of the fitbox

Table 5.7 Additional options for controlling the line break algorithm for `PDF_add/create_textflow()`, inline options in `PDF_create_textflow()` and `PDF_fill_textblock()` with a Textflow Block

option	description
adjustmethod	(Keyword) Method used to adjust a line when a text portion doesn't fit into a line after compressing or expanding the distance between words subject to the limits specified by the <code>minspacing</code> and <code>maxspacing</code> options. Default: <code>auto</code> . auto The following methods are applied in order: <code>shrink</code> , <code>spread</code> , <code>nofit</code> , <code>split</code> . clip Same as <code>nofit</code> , except that the long part at the right edge of the fitbox (taking into account the <code>rightindent</code> option) will be clipped. nofit The last word will be moved to the next line provided the remaining (short) line will not be shorter than the percentage specified in the <code>nofitlimit</code> option. Even justified paragraphs may look slightly ragged. shrink If a word doesn't fit in the line the text will be compressed subject to <code>shrinklimit</code> . If it still doesn't fit the <code>nofit</code> method will be applied. split The last word will not be moved to the next line, but will forcefully be split after the last character in the box. If <code>hyphenchar</code> is different from none a hyphen character will be inserted. Setting <code>hyphenchar=none</code> must be used to suppress the hyphen character (e.g. in formulae or URLs) since PDFlib does not automatically detect such situations. spread The last word will be moved to the next line and the remaining (short) line will be justified by increasing the distance between characters in a word, subject to <code>spreadlimit</code> . If justification still cannot be achieved the <code>nofit</code> method will be applied.
advanced-linebreak	(Boolean) Enable advanced line breaking algorithm which is required for complex scripts. This is required for line-breaking in scripts which do not use space characters for designating word boundaries, e.g. Thai. The options <code>locale</code> and <code>script</code> will be honored. Default: <code>false</code>
avoidbreak	(Boolean) If <code>true</code> , line breaking opportunities (e.g. at space characters) are ignored until <code>avoidbreak</code> is reset to <code>false</code> . Mandatory line breaks (e.g. at a newline) and methods defined by <code>adjustmethod</code> will be still performed. In particular, <code>adjustmethod=split</code> may still create hyphenation. Default: <code>false</code>
locale	(Keyword) The locale which will be used for localized line-breaking methods if <code>advancedlinebreak=true</code> . The keywords consists of one or more components, where the optional components are separated by an underscore character '_' (the syntax slightly differs from NLS/POSIX locale IDs): <ul style="list-style-type: none"> ▶ A required two- or three-letter lowercase language code according to ISO 639-2, e.g. <code>en</code>, (English), <code>de</code> (German), <code>ja</code> (Japanese). This differs from the <code>language</code> option. ▶ An optional two-letter uppercase country code according to ISO 3166, e.g. <code>DE</code> (Germany), <code>CH</code> (Switzerland), <code>GB</code> (United Kingdom) The keyword <code>_none</code> specifies that no locale-specific processing will be done. Specifying a locale is required for advanced line breaking for some scripts, e.g. Thai. Default: <code>_none</code> Examples: <code>tha</code> , <code>de_DE</code> , <code>en_US</code> , <code>en_GB</code>
maxspacing minspacing	(Float or percentage; only relevant if the line contains at least one space character <code>U+0020</code> and <code>alignment=justify</code>) Maximum or minimum distance between words (in user coordinates, or as a percentage of the width of the space character). The calculated word spacing is limited by the provided values (but the <code>wordspacing</code> option will still be added). Defaults: <code>minspacing=50%</code> , <code>maxspacing=500%</code>
nofitlimit	(Float or percentage; only relevant with <code>alignment=justify</code>) Lower limit for the length of a line with the <code>nofit</code> method ¹ . Default: <code>75%</code>
shrinklimit	(Percentage) Lower limit for compressing text with <code>adjustmethod=shrink</code> ; the calculated shrinking factor is limited by the provided value, but will be multiplied with the <code>horizscaling</code> option. Default: <code>85%</code>
spreadlimit	(Float or percentage) Upper limit for the distance between characters for the <code>spread</code> method ¹ ; the calculated distance will be added to the value of the <code>charspacing</code> option. Default: <code>0</code>

1. In user coordinates or as a percentage of the width of the fitbox

Table 5.8 Additional command options for `PDF_add/create_textflow()`, inline options in `PDF_create_textflow()` and `PDF_fill_textblock()` with a Textflow Block

option	description
comment	(String) Arbitrary text which is ignored; useful for commenting option lists or macros
inlineoptions	(Boolean; ignored in <code>PDF_create_textflow()</code> and inline options in <code>PDF_add_textflow()</code>) If true, inline options are searched in the supplied text. Default: false
mark	(Integer) Store the supplied number internally as a mark. The mark which has been stored most recently can later be retrieved with <code>PDF_info_textflow()</code> and the <code>lastmark</code> keyword. This may be useful for determining which portions of text have already been placed on the page.
matchbox	(Option list) Option list for creating a matchbox according to Table 6.4
nextline	(Boolean) Force a new line; equivalent to one of the characters <code>U+000B</code> or <code>U+2028</code> . The options <code>alignment=justify</code> and <code>lastalignment</code> don't have any effect on the line containing the <code>nextline</code> option. This option should not be used in the same inline option list together with the <code>matchbox</code> suboption <code>end</code> .
nextparagraph	(Boolean) Force a new paragraph; equivalent to one of the characters <code>U+000A</code> , <code>U+000D</code> , <code>U+000D</code> plus <code>U+000A</code> , <code>U+0085</code> , <code>U+2029</code> and <code>U+00FF</code> . The alignment of the line containing the <code>nextparagraph</code> option is determined by the option <code>lastalignment</code> ; the option alignment is ignored.
resetfont	(Boolean) Reset font and fontsize to the most recently values which were different from the current settings (either different font or font size). This may be useful to reset the font after inserts, such as italic text. The font option has precedence over this option. This command only makes sense after the first change of any font-related options which differ from the first setting, and is ignored otherwise.
restore	(Boolean) If true, the values of all text and Textflow options saved by the corresponding <code>save</code> command will be restored. A matchbox created within a <code>save/restore</code> pair is retained after <code>restore</code> . Default: false
return	(String; must not start with an underscore character) Exit <code>PDF_fit_textflow()</code> with the supplied string as return value. A new line will automatically be created.
save	(Boolean) If true, the values of all text and Textflow options will be saved, except tagging options and the non-state options <code>charmapping</code> , <code>nextline</code> , <code>nextparagraph</code> , <code>resetfont</code> , <code>return</code> , <code>space</code> , and <code>textlen</code> . <code>Save/restore</code> pairs can be nested to an arbitrary depth. Default: false
space	(Float or percentage) The text position will be advanced horizontally by the specified value ¹ .

1. In user coordinates or as a percentage of the font size

Table 5.9 Additional text semantics options for `PDF_add/create_textflow()`, inline options in `PDF_create_textflow()` and `PDF_fill_textblock()` with a Textflow Block

option	description
charclass	(List of pairs, where the first element in each pair is a keyword, and the second element is either a <code>Uni-char</code> or a list of <code>Unichars</code> ; the <code>Unichars</code> must be <code>< 0xFFFF</code> ; ignored if <code>advancedlinebreak=true</code>) The specified <code>Unichars</code> will be classified by the specified keyword to determine the line breaking behavior of those character(s): <ul style="list-style-type: none"> letter behave like a letter, e.g. <code>a B</code> punct behave like a punctuation character, e.g. <code>+ / ; :</code> open behave like an open parenthesis, e.g. <code>[</code> close behave like a close parenthesis, e.g. <code>]</code> default reset all character classes to PDFlib's builtin defaults Example: <code>charclass={ close » open « letter {/ : =} punct & }</code>

Table 5.9 Additional text semantics options for `PDF_add/create_textflow()`, inline options in `PDF_create_textflow()` and `PDF_fill_textblock()` with a Textflow Block

option	description
charmapping	<p>(List of pairs, where each pair either contains two Unichars or a Unichar and a list of Unichar and integer; the Unichars must be < U+FFFF) Replace individual characters with one or more instances of another character. The option list contains one or more pairs of Unichars. The first character in each pair will be replaced with the second character. Instead of one-to-one mapping the second element in each pair may be an option list containing a unichar and a count:</p> <p>count > 0 The replacement character will be repeated count times.</p> <p>count < 0 A sequence of multiple instances of the character will be reduced to the absolute value of the specified number.</p> <p>count = 0 The character will be deleted.</p> <p>Examples:</p> <pre>charmapping={ hortab space CRLF space LF space CR space } charmapping={ shy {shy 0} } charmapping={ hortab {space 4} }</pre>
hyphenchar	<p>(Unichar < 0xFFFF or keyword) Character which replaces a soft hyphen at line breaks. The value 0 and the keyword none completely suppress hyphens. The hyphen character is always taken from the main font; fallback fonts are not taken into account. Default: U+00AD (soft hyphen) if available in the font, U+002D (hyphen-minus) otherwise</p>
tabalignchar	<p>(Unichar < 0xFFFF) Character at which decimal tabs will be aligned. Default: U+002E ''</p>

Table 5.10 Additional tagging options for `PDF_add/create_textflow()`, inline options in `PDF_create_textflow()` and `PDF_fill_textblock()` with a Textflow Block

option	description
tagbegin¹	<p>(Option list; only for Tagged PDF) Open a structure element. All options for abbreviated structure element tagging according to Table 14.2 can be supplied. The specified tags must be properly nested and must obey the nesting rules for structure elements.</p>
tagend¹	<p>(Boolean; only for Tagged PDF) Close the most recently opened structure element in the Textflow.</p>

1. This option is ignored if the tag option for abbreviated tagging has been supplied to `PDF_fit_textflow()`.

Macros for Textflow options. Option lists for Textflows (either in the *optlist* parameter of `PDF_create_textflow()` or `PDF_add_textflow()`, or inline in the text supplied to `PDF_create_textflow()`) may contain macro definitions and macro calls according to Table 5.11. Macros may be useful for having a central definition of multiply used option values, such as font names, indentation amounts, etc. Before parsing an option list each contained macros will be substituted with the contents of the corresponding option list provided in the macro definition. The resulting option list will then be parsed. The following example demonstrates a macro definition for two macros:

```
<macro {
  comment { The following macros are used as paragraph styles }
  H1 {fontname=Helvetica-Bold fontsize=14 }
  body {fontname=Helvetica fontsize=12 }
}>
```

These macros could be used as follows in an option list:

```
<H1>Chapter 1
<body>This chapter talks about...
```

The following rules apply to macro definition and use:

- ▶ Macros may be nested to an arbitrary depth (macro definitions may contain calls to other macros).
- ▶ Macros can not be used in the same option list where they are defined. In `PDF_create_textflow()` a new inline option list which uses the macro can be started immediately after the end of the inline option list in which the macro is defined. When using `PDF_add_textflow()` one method call is required to define the macro, and another one to use it (since `PDF_add_textflow()` accepts only a single option list at a time).
- ▶ Macro names are case-insensitive.
- ▶ Undefined macros will result in an exception.
- ▶ Macros can be redefined at any time.

Table 5.11 Option list macro definitions and calls for `PDF_add/create_textflow()`, `PDF_fit_textflow()` and `PDF_fill_textblock()` with a Textflow Block

option	description
macro	(List of pairs) Each pair describes the name and definition of a macro as follows (note that there must not be any equals character '=' between the macro name and its definition): name (string) The name of the macro which can later be used for macro calls. Macros which have already been defined can be redefined later. The special name <code>comment</code> is ignored. suboptlist An option list which will literally replace the macro name when the macro is called. Leading and trailing whitespace is ignored.
&name	The macro with the specified name will be expanded, and the macro name (including the <code>&</code> character) will be replaced by the macro's contents, i.e. the suboptlist which has been defined for the macro (without the surrounding braces). The macro name is terminated by whitespace, <code>{</code> , <code>}</code> , <code>=</code> , or <code>&</code> . Therefore, these characters can not be used as part of a macro name. Nested macros will be expanded without any nesting limit. Macros contained in string options will also be expanded. Macro substitution must result in a valid option list.

C++ Java C# **int create_textflow(String text, String optlist)**

Perl PHP **int create_textflow(string text, string optlist)**

C **int PDF_create_textflow(PDF *p, const char *text, int len, const char *optlist)**

Create a Textflow object from text contents, inline options, and explicit options.

text (Content string) The contents of the Textflow. It may contain text, macros (see »Macros for Textflow options«, page 101) and inline option lists according to Table 5.6 and Table 5.12 (see also »Inline option lists for Textflows«, page 104). If `text` is an empty string, a valid Textflow handle will be returned nevertheless.

len (C language binding only) The length of text in bytes, or 0 for null-terminated strings.

optlist An option list specifying Textflow options. Options specified in the `optlist` parameter will be evaluated before those in inline option lists in `text` so that inline options

have precedence over options provided in the *optlist* parameter. The following options can be used:

- ▶ All options of *PDF_add_textflow()*
- ▶ Options for controlling inline option list processing according to Table 5.12: *begoptlistchar*, *endoptlistchar*, *fixedtextformat*, *textlen*

Returns A Textflow handle which can be used in calls to *PDF_add_textflow()*, *PDF_fit_textflow()*, *PDF_info_textflow()*, and *PDF_delete_textflow()*. The handle is valid until the end of the enclosing *document* scope, or until *PDF_delete_textflow()* is called with this handle. By default this method returns -1 (in PHP: 0) in case of an error. This behavior can be changed with the *errorpolicy* option.

Details This method accepts options and text to be prepared for Textflow. Unlike *PDF_add_textflow()* the text may contain inline options. Searching for inline option lists can be disabled for parts or all of the text by supplying the *textlen* option in the *optlist* parameter (see »Inline option lists for Textflows«, page 104).

This method does not create any output in the generated PDF document, but only prepares the text according to the supplied options. Use *PDF_fit_textflow()* to create output with the resulting Textflow handle.

See the *Details* section of *PDF_add_textflow()* for more information regarding special characters, line breaking, etc.

Scope any except *object*

Table 5.12 Additional options for inline option list processing in *PDF_create_textflow()*, *PDF_add_textflow()* with option *inlineoptions*, and *PDF_fill_textblock()* with a Textflow Block

option	description
<i>begoptlistchar</i>	(Unichar < 0xFFFF or keyword) Character which starts inline option lists. Replacing the default character may be useful if this character appears in the text literally (see »Inline option lists for Textflows«, page 104). If <i>textlen</i> is not specified, the <i>begoptlistchar</i> character in the text must be encoded in the same text format and encoding as the preceding text. This means that the Unicode value of <i>begoptlistchar</i> must be chosen such that it is contained in the encoding of the preceding text. The keyword <i>none</i> can be used to completely disable the search for option lists. Default: U+003C (<)
<i>endoptlistchar</i>	(Unichar < 0xFFFF; U+007D '}' is not allowed) Character which terminates inline option lists. Default: U+003E '>'
<i>fixedtextformat</i>	(Boolean; only for C as well as for Perl, PHP, Ruby with <i>stringformat=legacy</i> ; this option doesn't make sense in inline option lists, and can only be used in the <i>optlist</i> parameter) If true, all text fragments and inline options lists use the same <i>textformat</i> , which must be one of <i>utf8</i> , <i>utf16</i> , <i>utf16be</i> , or <i>utf16le</i> . This is useful if text and inline options come from the same source. If false, inline option lists including the delimiters must be encoded in <i>textformat=bytes</i> regardless of the format of the text. This allows the combination e.g. of UTF-16 text with ASCII-encoded inline option lists (the text may come from a Unicode database, while inline options are constructed as ASCII text within the application). Default: false
<i>textlen</i>	(Integer or keyword; required for text fragments with <i>encoding=glyphid</i> as well as for text fragments with <i>fixedtextformat=false</i> and <i>textformat=utf16xx</i> in non-Unicode aware languages; not allowed in <i>PDF_add_textflow()</i> with <i>inlineoptions=true</i>) Number of code units (e.g. bytes in language bindings with <i>stringformat=utf8</i> or UTF-16 units in Java or .NET; C++ in UTF-32 mode also expects the number of UTF-16 code units) before the next inline option list (see »Inline option lists for Textflows«, page 104). The code units are counted before character references are resolved, e.g. <textlen=8>①<...>. The keyword <i>all</i> specifies all of the remaining text. Default: the text is searched for the next occurrence of <i>begoptlistchar</i> .

Inline option lists for Textflows. The content provided in the *text* parameter of `PDF_create_textflow()`, or in `PDF_add_textflow()` with *inlineoptions=true* may include an arbitrary number of option lists (inline options) specifying Textflow options according to Table 5.6. All of these options can alternatively be provided in the *optlist* parameter of `PDF_create_textflow()` and `PDF_add_textflow()`. The same option can be specified multiply in a single option list; in this case only the last occurrence of an option will be taken into account.

Inline option lists must be enclosed with the characters specified in the *begoptlistchar* and *endoptlistchar* options (by default: < and >). Obviously, conflicts could arise if the character used for starting inline option lists must also be used in the actual text. There are several methods to resolve this conflict, depending on whether or not the text contains any inline option lists. Remember that `PDF_add_textflow()` completely separates text and options, so the conflict doesn't arise there.

If the text does not contain any inline options lists you can completely disable the search for inline option lists by one of the following methods:

- ▶ Set *begoptlistchar=none* in the *optlist* parameter of `PDF_create_textflow()`.
- ▶ Set the *textlen* option in the *optlist* parameter of `PDF_create_textflow()` to the length of the full text.

If the text actually contains inline option lists you can avoid the conflict between text contents and the *begoptlistchar* for starting an inline option list by using one of the following methods:

- ▶ Replace all occurrences of the < character in the text with the corresponding numeric or character entity reference (`<` or `<`) and start inline option lists with the literal < character:

```
A&lt;B<fontname=Helvetica>
```

Note that this method does not work for fonts with *encoding=builtin*.

- ▶ Set the *begoptlistchar* option in the *optlist* parameter of `PDF_create_textflow()` or an inline option list to a character which is not used in the text (e.g. \$), and use this character to start inline option lists:

```
<begoptlistchar=$>A<B$fontname=Helvetica>
```

- ▶ Specify the length of the next text fragment (until the start of the next inline option list) in the preceding inline option list using the *textlen* option:

```
<textlen=3>A<B<fontname=Helvetica>
```

- ▶ Specify the *begoptlistchar* as an escape sequence and set the *escapesequences* global option to *true*. However, escape sequences don't work within inline option lists including the *endoptlistchar*.

Note Consecutive inline option lists without any text between them should be avoided; it is recommended to aggregate all options in a single option list. If an inline option list is provided immediately after another option list, they are assumed to enclose a text fragment of zero length. This is important when supplying the *textlen* option in the first option list.

C++ Java C# **String fit_textflow(int textflow, double llx, double lly, double urx, double ury, String optlist)**

Perl PHP **string fit_textflow(int textflow, float llx, float lly, float urx, float ury, string optlist)**

C **const char *PDF_fit_textflow(PDF *p,
int textflow, double llx, double lly, double urx, double ury, const char *optlist)**

Format the next portion of a Textflow.

textflow A Textflow handle returned by a call to [PDF_create_textflow\(\)](#) or [PDF_add_textflow\(\)](#).

llx, lly, urx, ury *x* and *y* coordinates of the lower left and upper right corners of the target rectangle (the *fitbox*) in user coordinates. The corners can also be specified in reverse order. Shapes other than a rectangle can be filled with the *wrap* option.

optlist An option list specifying processing options. The following options can be used:

- ▶ Textflow options according to Table 5.13:
avoidwordsplitting, blind, createfittext, createlastindent, exchangefillcolors, exchange-strokecolors, firstlinedist, fitmethod, fontscale, lastlinedist, linespreadlimit, maxlines, minfontsize, orientate, returnatmark, rewind, rotate, showborder, showtabs, stamp, truncatetrailingwhitespace, verticalalign, wrap
- ▶ Matchbox option according to Table 6.1: *matchbox*
- ▶ Option for abbreviated structure element tagging according to Table 14.2 (only allowed in *page* scope): *tag*

Returns A string which specifies the reason for returning from the method:

- ▶ *_stop*: all text in the Textflow has been processed. If the text was empty, *_stop* will always be returned, even if the *return* or *mark/returnatmark* option was supplied.
- ▶ *_nextpage*: Waiting for the next page (caused by a form feed character U+000C). Another call to [PDF_fit_textflow\(\)](#) is required for processing the remaining text.
- ▶ *_boxfull*: Some text was placed in the fitbox, but no more space is available, or the maximum number of lines (as specified via the *maxlines* option) has been placed in the fitbox, or *fitmethod=auto* and *minfontsize* has been specified but the text didn't fit into the fitbox. Another call to [PDF_fit_textflow\(\)](#) is required for processing the remaining text.
- ▶ *_boxempty*: The box doesn't contain any text at all after processing. This may happen if the size of the fitbox is too small to hold any text, or a wrapbox was larger than the fitbox. No more calls to [PDF_fit_textflow\(\)](#) with the same fitbox should be issued in order to avoid infinite loops.
- ▶ *_mark#*: The option *returnatmark* has been specified with the number #, and the mark with the number specified in this option has been placed.
- ▶ Any other string: The string supplied to the *return* command in an inline option list.

If there are multiple simultaneous reasons for returning, the first in the list (from top to bottom) will be reported. The returned string is valid until the next call to this method.

Details The current text and graphics states do not influence the text output created by this method (this is different from [PDF_fit_textline\(\)](#)). Use *fillcolor*, *strokecolor* and other text appearance options (see Table 4.6) in [PDF_create_textflow\(\)](#) or [PDF_add_textflow\(\)](#) to control the appearance of the text. After returning from this method the text state is

unchanged. However, the *textx/texty* options are adjusted to point to the end of the generated text output (unless the *blind* option has been set to *true*).

Scope *page, pattern, template, glyph*

Table 5.13 Options for *PDF_fit_textflow()* and *PDF_fill_textblock()* with a *Textflow Block*

option	description
avoidword-splitting	(Boolean) If <i>true</i> and <i>fitmethod=auto</i> , <i>Textflow</i> attempts to fit the text completely into the fitbox by decreasing the fontsize and avoiding word splitting (see <i>adjustmethod</i>).
blind	(Boolean) If <i>true</i> , no output will be generated, but all calculations will be performed and the formatting results can be checked with <i>PDF_info_textflow()</i> . Default: <i>false</i>
createfittext	(Boolean) If <i>true</i> the text placed in the current fitbox will be saved in memory so that it can later be retrieved with a call to <i>PDF_info_textflow()</i> and the keyword <i>fittext</i> . Default: <i>true</i>
createlast-indent	(Option list) Reserve some space at the end of the last line in the fitbox and optionally create a matchbox which can be used to fill the reserved space. The reserved space may be useful to add continuation dots, an image, a link to the continuation of the text, etc. at the end of the text: rightindent (Float or percentage) Additional right indent of the last text line in the fitbox in user coordinates or as percentage of the width of the fitbox. The value will be added to the value of the <i>rightindent</i> option of <i>PDF_add/create_textflow()</i> . Default: <i>0</i> matchbox (Option list according to Table 6.4) Create a matchbox at the end of the last line. If the <i>matchbox</i> option <i>boxwidth</i> is not specified, the value of <i>rightindent</i> will be used as <i>boxwidth</i> . If <i>boxwidth=0</i> no box will be created.
exchange-fillcolors	(List with an even number of colors) Each pair in the list specifies an original fill color and a replacement color. Whenever the <i>Textflow</i> specifies the original fill color within the fitbox it will be replaced with the specified replacement color. This may be useful to adjust the colors to the background. The character <i>''</i> as original color means that all remaining colors will be replaced with the replacement color. Example: <i>exchangefillcolors={{gray 0} white Orchid DeepPink {rgb 1 0 1} MediumBlue}</i>
exchange-strokecolors	(List with an even number of colors) Each pair in the list specifies an original stroke color and a replacement color. Whenever the <i>Textflow</i> specifies the original stroke color within the fitbox it will be replaced with the specified replacement color. This may be useful to adjust the colors to the background. The character <i>''</i> as original color means that all remaining colors will be replaced with the replacement color.
firstlinedist¹	(Float, percentage, or keyword) Distance between the top of the fitbox and the baseline for the first line of text, specified in user coordinates, as a percentage of the relevant font size (the first font size in the line if <i>fixedleading=true</i> , and the maximum of all font sizes in the line otherwise), or as a keyword (default: <i>leading</i>): leading The leading value determined for the first line; typical diacritical characters such as <i>À</i> will touch the top of the fitbox. ascender The ascender value determined for the first line; typical characters with larger ascenders, such as <i>d</i> and <i>h</i> will touch the top of the fitbox. capheight The <i>capheight</i> value determined for the first line; typical capital uppercase characters such as <i>H</i> will touch the top of the fitbox. xheight The <i>xheight</i> value determined for the first line; typical lowercase characters such as <i>x</i> will touch the top of the fitbox. If <i>fixedleading=false</i> the maximum of all <i>leading</i> , <i>ascender</i> , <i>xheight</i> , or <i>capheight</i> values found in the first line will be used.

Table 5.13 Options for `PDF_fit_textflow()` and `PDF_fill_textblock()` with a Textflow Block

option	description
fitmethod	(Keyword) Method to fit the text into the fitbox: auto <code>PDF_fit_textflow()</code> will repeatedly be called in blind mode with reduced font size and other font-related options (see <code>fontscale</code>) until the text fits into the fitbox (but see also option <code>minfontsize</code>). clip The text will be truncated at the bottom of the fitbox. nofit The text can extend beyond the bottom of the fitbox (for <code>verticalalign=top</code>), the top of the fitbox (for <code>verticalalign=bottom</code>), or both (for <code>verticalalign=center</code>). The default for <code>PDF_fit_textflow()</code> is <code>clip</code> . The default for <code>PDF_fill_textblock()</code> with a Textflow Block is <code>clip</code> if the <code>textflowhandle</code> option is supplied, otherwise <code>auto</code>
fontscale	(Positive float or percentage) Values of <code>fontsize</code> and absolute values (but not percentages) of <code>leading</code> , <code>minspacing</code> , <code>maxspacing</code> , <code>spreadlimit</code> , and <code>space</code> will be multiplied with the supplied scaling factor or percentage. Default: 1 if <code>rewind=0</code> , otherwise the value supplied with the corresponding call to <code>PDF_fit_textflow()</code> .
gstate	(Option list with options for <code>PDF_create_gstate()</code> or <code>gstate handle</code>) Graphics state options or handle. The graphics state affects all text created with this method. If other graphics states are supplied to <code>PDF_add/create_textflow()</code> the graphics states are merged. Default: no graphics state, i.e. current settings are used
lastlinedist¹	(Float, percentage, or keyword; ignored for <code>fitmethod=nofit</code>) Minimum distance between the baseline for the last line of text and the bottom of the fitbox, specified in user coordinates, as a percentage of the font size (the first font size in the line if <code>fixedleading=true</code> , and the maximum of all font sizes in the line otherwise), or as a keyword. Default: 0, i.e. the bottom of the fitbox will be used as baseline, and typical descenders will extend below the fitbox. The following keyword can be used: descender The descender value determined for the last line; typical characters with descenders, such as <code>g</code> and <code>j</code> will touch the bottom of the fitbox. If <code>fixedleading=false</code> the maximum of all descender values found in the last line is used.
linespread-limit	(Float or percentage; only for <code>verticalalign=justify</code>) Maximum amount in user coordinates or as percentage of the leading for increasing the leading for vertical justification. Default: 200%
maxlines	(Integer or keyword) Maximum number of lines in the fitbox, or the keyword <code>auto</code> which means that as many lines as possible will be placed in the fitbox. When the maximum number of lines has been placed <code>PDF_fit_textflow()</code> will return the string <code>_boxfull</code> . Default: <code>auto</code>
minfontsize	(Float or percentage) Minimum font size allowed when text is scaled down to fit into the fitbox, especially for <code>fitmethod=auto</code> . The limit is specified in user coordinates or as a percentage of the height of the fitbox. If the limit is reached and the text still does not fit the string <code>_boxfull</code> will be returned. Default: 0.1%
mingapwidth	(Float or percentage) Minimal horizontal width for fitting text between shapes (e.g. between wrap contours) in user coordinates or as a percentage of the <code>fontsize</code> . This may be useful to avoid ugly formatting results in cases where only small gaps are left between wrap contours. Default: 10%
orientate	(Keyword) Desired orientation of the text when it is placed (default: <code>north</code>): north upright east pointing to the right south upside down west pointing to the left
returnatmark	(Integer) <code>PDF_fit_textflow()</code> will return prematurely at the text position where the Textflow option <code>mark</code> is defined with the specified number. The return reason string will be <code>_mark#</code> , where <code>#</code> is the number specified in this option.

Table 5.13 Options for `PDF_fit_textflow()` and `PDF_fill_textblock()` with a Textflow Block

option	description
rewind	(Integer: -2, -1, 0, or 1; not allowed if the Textflow contains inline tags) The state of the supplied Textflow is reset to the state before some other call to <code>PDF_fit_textflow()</code> with the same Textflow handle and then the Textflow is placed as usual (default: 0): <ul style="list-style-type: none"> 1 Rewind to the state before the first call to <code>PDF_fit_textflow()</code>. 0 Don't reset the Textflow. -1 Rewind to the state before the previous call to <code>PDF_fit_textflow()</code>. -2 Rewind to the state before the second last call to <code>PDF_fit_textflow()</code>.
rotate	(Float) Rotate the coordinate system, using the lower left corner of the fitbox as center and the specified value as rotation angle in degrees. This results in the fitbox and the text being rotated. The rotation will be reset when the text has been placed. Default: 0
showborder	(Boolean) If true, the border of the fitbox and all wrap boxes is stroked (using the current graphics state). This may be useful for development and debugging. Default: false
showtabs	(Keyword) Tab stops and left indents are visualized with vertical lines as a debugging aid. The lines will be drawn according to the graphics state which was active before calling <code>PDF_fit_textflow()</code> (default: none): <ul style="list-style-type: none"> none no lines will be drawn fitbox lines will be drawn over the full height of the fitbox validarea lines will be drawn only in vertical area where they are valid
stamp	(Keyword) This option can be used to create a diagonal stamp within the fitbox. Line breaks for the stamp text should be specified explicitly (i.e. with newline characters or the <code>newline</code> option). If the text does not contain any explicit line breaks a single-line stamp will be created. The generated stamp text will be as large as possible, but not larger than the specified fontsize. Supported keywords (default: none): <ul style="list-style-type: none"> llzur The stamp will run diagonally from the lower left corner to the upper right corner. ulzlr The stamp will run diagonally from the upper left corner to the lower right corner. none No stamp will be created.
truncate-trailing-whitespace	(Boolean) Control treatment of fitboxes which contain only trailing whitespace, i.e. the text in the fitbox starts with whitespace and there is only whitespace until the end of the Textflow. If this option is true, trailing whitespace is removed, i.e. the fitbox is treated as empty and the return value is <code>_stop</code> . If this option is false, the whitespace is processed like regular text, i.e. the method may return a value different from <code>_stop</code> (depending on the amount of trailing whitespace) and the <code>textendx/y</code> and other keywords of <code>PDF_info_textflow()</code> take the whitespace into account. <code>truncatetrailingwhitespace=false</code> may be useful if the original text must be processed without any whitespace removal. Default: true
verticalalign¹	(Keyword) Vertical alignment of the text in the fitbox; the <code>firstlinedist</code> and <code>lastlinedist</code> options will be taken into account as appropriate (default: top): <ul style="list-style-type: none"> top Formatting will start at the first line, and continue downwards. If the text doesn't fill the fitbox there may be whitespace below the text. center The text will be vertically centered in the fitbox. If the text doesn't fill the fitbox there may be whitespace both above and below the text. bottom Formatting will start at the last line, and continue upwards. If the text doesn't fill the fitbox there may be whitespace above the text. justify The text will be aligned with top and bottom of the fitbox. In order to achieve this the leading will be increased up to the limit specified by <code>linespreadlimit</code>. If this limit is exceeded no justification will be performed. The height of the first line will only be increased if <code>firstlinedist=leading</code>.

Table 5.13 Options for `PDF_fit_textflow()` and `PDF_fill_textblock()` with a Textflow Block

option	description
wrap	<p>(Option list according to Table 5.14) The text will run around the areas specified with the suboptions listed in Table 5.14. This can be used to place images or paths within the Textflow and wrap the text around it, or to fill arbitrary shapes with text. The fitbox will be filled according to the <code>fillrule</code> option, starting at the border of the fitbox.</p> <p>By default, the specified areas will not contain any text (except where they overlap), i.e. the text is wrapped around the shapes. Using the <code>addfitbox</code> and <code>inversefill</code> options the opposite effect can be achieved: the specified areas will be filled with text, and the rest of the fitbox remains empty. This can be used to fill arbitrary shapes (and not only the rectangle supplied in the <code>llx/lly/urx/ury</code> parameters) with text.</p> <p>Absolute and relative coordinate values will be interpreted in the user coordinate system. A relative coordinate will be added to the previous absolute coordinate. Up to 256 values can be supplied as relative values. Percentages will be interpreted in the fitbox coordinate system, i.e. the lower left corner of the fitbox is (0, 0) and the upper right corner is (100, 100) (even in a topdown system). Up to 256 values can be supplied as percentage. Examples:</p> <p>Exclude a box with relative coordinates: <code>wrap={ boxes={{120r 340r 50r 60r}} }</code> (equivalent to <code>wrap={ boxes={{120 340 170 400}} }</code>)</p> <p>Exclude the upper right quarter of the fitbox: <code>wrap={ boxes={{50% 50% 100% 100%}} }</code></p> <p>Fill a triangular shape: <code>wrap={ addfitbox polygons={{50% 80% 30% 40% 70% 40% 50% 80%}} }</code></p> <p>Exclude the area of an image with a matchbox called <code>image1</code>: <code>wrap={ usematchboxes={{ image1 }} }</code></p>

1. The `firstlinedist`, `lastlinedist` and `verticalalign` options always refer to the fitbox, even in the presence of wrap elements. This means – especially in the case of inverse filling, i.e. the wrap elements are filled with text – that Textflow will not use the bounding box of the wrap elements to determine the distance between text and fitbox borders and the position of the text box according to the `verticalalign` option. This may lead to unexpected results, especially if the outer edges of the wrap elements don't touch the fitbox. This effect can almost completely be avoided by supplying wrap elements which touch the fitbox.

C++ Java C# **double** `info_textflow(int textflow, String keyword)`

Perl PHP **float** `info_textflow(int textflow, string keyword)`

C **double** `PDF_info_textflow(PDF *p, int textflow, const char *keyword)`

Query the current state of a Textflow after a call to `PDF_fit_textflow()`.

textflow A Textflow handle returned by a call to `PDF_add/create_textflow()` or `PDF_fill_textblock()` with the `textflowhandle` option.

keyword A keyword specifying the requested information according to Table 5.15.

Returns The value of some Textflow parameter as requested by `keyword`. This method returns correct geometry information even in blind mode (unlike the `textx/texty` options). If the requested keyword produces text, a string index is returned, and the corresponding string must be retrieved with `PDF_get_string()`.

Scope any except `object`

Table 5.14 Suboptions for the wrap option of `PDF_fit_textflow()` and `PDF_fill_textblock()` with a Textflow Block

option	description
addfitbox	(Boolean) If true, the fitbox will be added to the wrap area. As a result, the shapes specified with other wrapping options will be filled with text instead of wrapping the text around the shapes. Default: false
beziers	(List of two or more Bézier curves) Bézier curves which will be added to the wrap area.
boxes	(List of rectangles) One or more rectangles which will be added to the wrap area.
circles	(List of circles) One or more circles which will be added to the wrap area.
creatematch-boxes	(List of option lists) Create matchboxes from one or more rectangles in the boxes option. Each option list corresponds to one entry in the boxes option (ordering is relevant), and controls the creation of a matchbox. All relevant matchbox options in Table 6.4 can be used. A suboption list can be empty; in this case no matchbox will be created for the corresponding wrap box.
fillrule	(Keyword) Method for determining the interior of overlapping wrap shapes (default: evenodd). See Table 7.1 for details: evenodd Use the even-odd rule. winding Use the non-zero winding number rule. Use this rule to process the interior of overlapping circles (i.e. to avoid »doughnut holes«), or to process the union of overlapping shapes (instead of the intersection).
inversefill	(Boolean) If true, wrap shape processing starts at the first intersection of the text line with the border of a wrap element inside the fitbox. If false, processing starts at the fitbox border. If fillrule=evenodd, the option inversefill=true has the same effect as addfitbox=true. If fillrule=winding, the option addfitbox=true leads to an empty or a full fitbox (for inversefill=false or true, respectively).
lineheight	(List with two elements, each being a positive float or a keyword) Defines the vertical extent of the text line to be used for calculating the intersection with wrap areas. Two keywords/floats specify the extent above and below the text baseline. Supported keywords: none (no extent), xheight, descender, capheight, ascender, fontsize, leading, textrise Default: {ascender descender}
usematch-boxes	(List of string lists) The first element in each list is a name string which specifies a matchbox. The second element is either an integer specifying the number of the desired rectangle, or the keyword all to specify all rectangles referring to the selected matchbox. If the second element is missing, it defaults to all. The bounding box of each rectangle will be added to the wrap area.
offset	(Float or percentage) Horizontal distance between the text and the contour of the wrap area, supplied in user coordinates or as a percentage of the width of the fitbox. This can be used to horizontally extend the wrap area. Default: 0
paths	(List of option lists) One or more path objects which will be added to the wrap area: path (Path handle; required) Handle for the path to be added to the wrap area. refpoint (List of two floats or percentages) Coordinates of the reference point for the path in user coordinates or as percentages of the width and height of the fitbox. Default: {0 0} The following options of <code>PDF_draw_path()</code> can also be used (see Table 6.1 and Table 7.7): align, attachmentpoint, boxsize, close, fitmethod, orientate, position, round, scale, subpaths
polygons	(List of polylines) One or more polylines (not necessarily closed) which will be added to the wrap area.

Table 5.15 Keywords for `PDF_info_textflow()`

keyword	description
boundingbox	Handle of the path containing the Textflow's bounding box in user coordinates or -1 (0 in PHP). <code>firstlinedist</code> and <code>lastlinedist</code> will be taken into account.
boxlinecount	Number of lines in the last fitbox
firstparalinecount	Number of lines in the first paragraph of the fitbox
firstlinedist	Distance between the first text baseline and the fictitious baseline above (if <code>verticalalign=top</code> this will be the upper border of the fitbox)
fittext	String index for the text placed in the previous call to <code>PDF_fit_textflow()</code> . This can be used to determine the amount of text which could be placed in the fitbox. The string will be normalized as follows: encoding is UTF-16 in Unicode-capable languages or (EBCDIC-)UTF-8 otherwise, line breaks will be marked with U+000A, and horizontal tabs will be replaced with a space character U+0020.
fontscale	Value of <code>fontscale</code> after the most recent call to <code>PDF_fit_textflow()</code> with <code>fitmethod=auto</code>
lastfont	Handle of the font used in the last text line in the fitbox
lastfontsize	Font size used in the last text line in the fitbox
lastmark	Number of the last mark found in the processed part of the Textflow in the last fitbox (marks can be set with the <code>mark</code> option)
lastlinedist	Distance between the last text baseline and the fictitious baseline below, assuming unmodified leading (if <code>verticalalign=bottom</code> this will be the lower border of the fitbox)
lastparalinecount	Number of lines in the last paragraph of the fitbox
leading	The current value of the <code>leading</code> option, as determined by the text and options within the Textflow
leftlinex¹, leftliney¹	The x and y coordinates of the line with the leftmost start in the most recently filled fitbox, in current user coordinates
maxlinelength	Length of the longest text line in the most recently filled fitbox
maxliney¹	The y coordinate of the baseline of the longest text line in the most recently filled fitbox, in current user coordinates
minlinelength	Length of the shortest text line in the most recently filled fitbox
minliney¹	The y coordinate of the baseline of the shortest text line in the most recently filled fitbox, in current user coordinates
returnreason	String index for the return reason of the most recent direct or indirect call to <code>PDF_fit_textflow()</code> . The retrieved return reason will be one of the return strings of <code>PDF_fit_textflow()</code> . This is useful for querying the result of indirect Textflow calls issued internally by <code>PDF_fill_textblock()</code> .
rightlinex¹, rightliney¹	The x and y coordinates of the line with the rightmost end in the most recently filled fitbox, in current user coordinates
split	Specifies whether word splitting occurred in the last fitbox: 0 No word had to be split. 1 At least one word had to be split.
textendx, textendy	The x or y coordinate of the current text position after the most recently filled fitbox in current user coordinates
textheight	Height of the bounding box of the whole text (taking <code>firstlinedist</code> and <code>lastlinedist</code> into account) in current user coordinates

Table 5.15 Keywords for `PDF_info_textflow()`

keyword	description
<code>textwidth</code>	Width of the bounding box of the whole text in current user coordinates
<code>used</code>	Percentage of text (0...100) which has been placed so far
<code>x1, y1, ..., x4, y4</code>	Coordinates of the bounding box of the whole text in current user coordinates. <code>firstlinedist</code> and <code>lastlinedist</code> will be taken into account. <code>x1, y1</code> correspond to the lower left, <code>x2, y2</code> to the lower right, <code>x3, y3</code> to the upper right and <code>x4, y4</code> to the upper left corner.

1. If `rotate` is different from 0 this value refers to the rotated system.

C++ Java C# **`void delete_textflow(int textflow)`**

Perl PHP **`delete_textflow(int textflow)`**

C **`void PDF_delete_textflow(PDF *p, int textflow)`**

Delete a Textflow and all associated data structures.

`textflow` A Textflow handle returned by a call to `PDF_create_textflow()` or `PDF_add_textflow()`.

Details Textflows which have not been deleted with this method will be deleted automatically at the end of the enclosing *document* scope. However, failing to call `PDF_delete_textflow()` may significantly slow down the application if many Textflows are generated.

Scope any

5.3 Table Formatting

C++ Java C# *int add_table_cell(int table, int column, int row, string text, string optlist)*

Perl PHP *int add_table_cell(int table, int column, int row, string text, string optlist)*

C *int PDF_add_table_cell(PDF *p,
int table, int column, int row, const char *text, int len, const char *optlist)*

Add a cell to a new or existing table.

table A valid table handle retrieved with another call to *PDF_add_table_cell()*, or -1 (in PHP: 0) to start a new table. The table handle must not yet have been used in a call to *PDF_fit_table()*, i.e. all table contents must be defined before placing the table on the page.

column, row Number of the column and row containing the cell. If the cell spans multiple columns and/or rows the numbers of the leftmost column and the topmost row must be supplied. The first column/row has number 1.

text (Content string) Text for filling the cell. If *text* is not empty it will be used for filling the cell with *PDF_fit_textline()*.

len (C language binding only) Length of *text* (in bytes). If *len* = 0 a null-terminated string must be provided.

optlist An option list specifying table cell formatting details:

- ▶ General option: *errorpolicy* (see Table 1.5)
- ▶ Column and row definition options according to Table 5.16:
colwidth, colscalegroup, minrowheight, return, rowheight, rowjoingroup, rowscalegroup
- ▶ Cell property options according to Table 5.16:
avoidwordsplitting, colspan, margin, marginleft, marginbottom, marginright, margintop, rowspan
- ▶ Cell content formatting options according to Table 5.16.:
continuetextflow, repeatcontent
- ▶ Static cell contents according to Table 5.17:
fitgraphics, fitimage, fitpath, fitpdipage, fittextflow, fittextline, graphics, image, matchbox, path, pdipage, textflow
- ▶ Interactive cell contents according to Table 5.18 (only in *page* scope):
annotationtype, fieldname, fieldtype, fitannotation, fitfield
- ▶ Option for abbreviated structure element tagging according to Table 14.2: *tag*

Returns A table handle which can be used in subsequent table-related calls. If *errorpolicy=return* the caller must check for a return value of -1 (in PHP: 0) since it signals an error. In case of an error only the last cell definition will be discarded; no contents will be added to the table, but the table handle is still valid. The returned table handle can not be reused across multiple PDF output documents.

Details A table cell can be filled with images, graphics, imported PDF pages, path objects, form fields, annotations, Textflows, or Textlines. Multiple content types can be specified for a particular cell in a single method call.

See the PDFlib Tutorial for a description of the table formatting algorithm and width and height calculations.

PDF/UA Vector graphics and raster images supplied with the *path* or *image* options must be tagged as *Artifact* or *Figure* with the *tag* option.

Scope any except object

Table 5.16 Formatting options for `PDF_add_table_cell()`

option	description
avoidword-splitting	(Boolean; only relevant for Textflow cells) If true, the table formatter will check whether the Textflow requires at least one forced word splitting when fitting the text into the table cell. If so, the cell width will be increased in an attempt to avoid word splittings. Default: true
colscale-group¹	(String) Name of a column group to which the column will be added. All columns in a group will be scaled uniformly if one of the columns in the group must be enlarged to completely hold long text. If a cell spans multiple columns the affected columns form a scale group automatically.
colspan	(Integer) Number of columns spanned by the cell. Default: 1
colwidth¹	(Float or percentage) Width of the column specified in the column parameter. The width can be specified in user coordinates ² , or as a percentage of the width of the table's first fitbox (see <code>PDF_fit_table()</code>). User coordinates and percentages must not be mixed, i.e. either user coordinates or percentages must be used in all column width definitions of a table. The column width may be increased automatically if the column traverses cells containing text. Images, graphics and PDF pages in table cells don't have any influence on column widths. Default: see option <code>colwidthdefault</code> of <code>PDF_fit_table()</code>
continue-textflow	(Boolean; only relevant for Textflows) If true the contents of the Textflow specified in the <code>textflow</code> option can be continued in another cell provided that the other cell is filled with the same Textflow handle and <code>continuetextflow=true</code> as well. The parts of the Textflow are placed in the order in which the cells are added. PDFlib does not adjust the cell size to the whole Textflow, and the <code>avoidwordsplitting</code> option is ignored. Therefore, a suitable cell size should be defined. This option is useful for placing the Textflow in multiple table cells which don't form a row or column span. If false the Textflow is started from the beginning. Default: false
margin marginleft marginbottom marginright marginintop	(Float or percentage) Left/bottom/right/top cell margins in user coordinates (must be greater than or equal to 0) or as a percentage of the cell width or height (must be less than 100%). The specified margins define the inner cell box which serves as the fitbox for the cell contents. Default for <code>margin</code> : 0; Default for all others: <code>margin</code>
minrow-height¹	(Float or percentage; not recommended) This option indicates that a row which cannot completely be placed at the end of a table instance can be split into two fragments. The minimum height of the first fragment in the current table instance can be specified in user coordinates or as a percentage of the row height. The next table instance starts with the second fragment. This option should be used carefully and only if the row contains a Textflow which is very high compared to the height of the fitbox. The option value should be estimated such that at least one text line can be placed in the cell; otherwise an error will occur. Default: 100%, i.e. no splitting
repeatcontent	(Boolean) Specify whether the contents of a table cell will be repeated if a cell or row is split between several table instances. Default: true Splitting a cell: If the last rows spanned by a cell don't fit into the fitbox, the cell will be split. Except for Textflows (which will not be repeated), the cell contents will be repeated in the next table instance if <code>repeatcontent=true</code> . Otherwise it will not be repeated. Splitting a row: If the last body row doesn't fit into the fitbox, it will usually not be split but will completely be placed in the next table instance. You can decrease the <code>minrowheight</code> value to split the last body row with the given percentage of contents in the first instance, and place the remaining parts of that row in the next instance. Except for Textflows (which will not be repeated), the cell contents will be repeated in the next table instance if <code>repeatcontent=true</code> . Otherwise it will not be repeated.

Table 5.16 Formatting options for `PDF_add_table_cell()`

option	description
return ¹	(String) <code>PDF_fit_table()</code> will stop after placing the specified row, and will return the specified string. The string must not start with an underscore character '_'. If the specified row is part of a join group it must be the last row of the group; otherwise an error will occur.
rowheight ¹	(Float or percentage) Height of the row specified in the <code>row</code> parameter. The height can be specified in user coordinates ² or as a percentage of the height of the table's first fitbox (see <code>PDF_fit_table()</code>). User coordinates and percentages must not be mixed, i.e. either user coordinates or percentages must be used in all row height definitions of a table. The row height may be increased automatically if the row traverses cells containing text. Images, graphics and PDF pages in table cells don't have any influence on row heights. Default: see option <code>rowheightdefault</code> of <code>PDF_fit_table()</code>
rowscale-group ¹	(String) Name of a row group to which the row will be added. All rows in a group will be scaled uniformly if one of the rows in the group must be enlarged to completely hold long text. If a cell spans multiple rows the affected rows form a scale group automatically.
rowjoin-group ¹	(String) Name of a row group to which the row will be added. All rows in the group will be kept together in a table instance. The rows in a group must be numbered consecutively. If a cell spans multiple rows the affected rows do not automatically form a join group.
rowspan	(Integer) Number of rows spanned by the cell. Default: 1

1. The last specification of this option is dominant; earlier specifications for the same row or column are ignored.
2. More precisely, the coordinate system in effect when `PDF_fit_table()` is called for placing the first table instance.

Table 5.17 Options for static cell contents in `PDF_add_table_cell()` and suboptions for the `caption` option of `PDF_fit_table()`

option	description
fitgraphics	(Option list; only relevant for graphics) Option list for <code>PDF_fit_graphics()</code> . This option list will be applied to place the graphics supplied with the <code>graphics</code> option in the cell. The lower left corner of the fitbox will be used as reference point. Default: <code>fitmethod=meet position=center</code> . This option list is prepended to the user-specified options. ¹
fitimage	(Option list; only relevant for images and templates) Option list for <code>PDF_fit_image()</code> . This option list will be applied to place the image or template supplied with the <code>image</code> option in the cell. The lower left corner of the fitbox will be used as the reference point. Default: <code>fitmethod=meet position=center</code> . This option list is prepended to the user-specified options. ¹
fitpath	(Option list; only relevant for path objects) Option list for <code>PDF_draw_path()</code> . This option list will be applied to place the path object specified with the <code>path</code> option within its bounding box in the cell. The lower left corner of the fitbox will be used as reference point. Default: <code>fitmethod=meet position=center</code> . This option list is prepended to the user-specified options. ¹
fitpdipage	(Option list; only relevant for PDI pages; only if PDI is available) Option list for <code>PDF_fit_pdi_page()</code> . This option list will be applied to place the page supplied with the <code>pdipage</code> option in the cell. The lower left corner of the fitbox will be used as the reference point. Default: <code>fitmethod=meet position=center</code> . This option list is prepended to the user-specified options. ¹
fittextflow	(Option list; only relevant for Textflows) Option list for <code>PDF_fit_textflow()</code> . This option list will be applied to place the Textflow supplied in the <code>textflow</code> option in the cell. The fitbox will be used as fitbox. Default: <code>verticalalign=center lastlinedist=descender</code> . This option list will be prepended to the user-specified option list.

Table 5.17 Options for static cell contents in `PDF_add_table_cell()` and suboptions for the caption option of `PDF_fit_table()`

option	description
fittextline	(Option list; only relevant for Textlines) Option list for <code>PDF_fit_textline()</code> . This option list will be applied to fit the text supplied with the <code>text</code> parameter into the cell. The lower left corner of the fitbox will be used as the reference point. Options which have not been specified will be replaced with the respective defaults; the current text state is not taken into account. Default: <code>fitmethod=nofit position=center</code> . This option list is prepended to the user-specified options. ¹
graphics	(Graphics handle) The graphics associated with the handle will be placed in the fitbox.
image	(Image handle) The image or template associated with the handle will be placed in the fitbox.
matchbox	(Option list) Option list with matchbox details according to Table 6.4.
path	(Path handle) The path object within its bounding box will be placed in the fitbox according to the <code>fitpath</code> option.
pdipage	(Page handle) The imported PDF page associated with the handle will be placed in the fitbox.
text	(Content string) Text to be placed with <code>PDF_fit_textline()</code> according to the option <code>fittextline</code> . In <code>PDF_add_table_cell()</code> the value of this option can alternatively be provided via the method parameter <code>text</code> .
textflow	(Textflow handle) The Textflow associated with the handle will be placed in the fitbox. The <code>continue-textflow</code> option controls the behavior for a Textflow handle which is used in multiple cells. The Textflow handle must not be used outside the table.

1. The box size is calculated automatically; any `boxsize` option in the supplied option list is ignored.

Table 5.18 Options for interactive cell contents for `PDF_add_table_cell()` and suboptions for the caption option (only in page scope)

option	description
annotation-type	(String) Type of an annotation to be inserted in the table cell according to Table 12.2.
fieldname	(Hypertext string) Form field name for <code>fieldtype</code> .
fieldtype	(String) Type of a form field to be inserted in the table cell according to Table 12.5. Form field groups should be defined outside of tables.
fitannotation	(Option list) Annotation options for <code>annotationtype</code> according to Table 12.3.
fitfield	(Option list) Form field options for <code>fieldtype</code> according to Table 12.6.

C++ Java C# **`String fit_table(int table, double llx, double lly, double urx, double ury, String optlist)`**
 Perl PHP **`string fit_table(int table, float llx, float lly, float urx, float ury, string optlist)`**
 C **`const char *PDF_fit_table(PDF *p, int table, double llx, double lly, double urx, double ury, const char *optlist)`**

Fully or partially place a table on the page.

table A valid table handle retrieved with a call to `PDF_add_table_cell()`.

llx, lly, urx, ury Coordinates of the lower left and upper right corners of the target rectangle for the table instance (the fitbox) in user coordinates. The corners can also be specified in reverse order.

optlist An option list specifying filling details according to Table 5.19. The following options can be used:

- ▶ General option: *errorpolicy* (see Table 1.5)
- ▶ Fitting options according to Table 6.1: *fitmethod*, *position*, *showborder*
- ▶ General table options:
blind, *colwidthdefault*, *horshrinklimit*, *rewind*, *rowheightdefault*, *vertshrinklimit*
- ▶ Table contents: *caption*, *header*, *footer*
- ▶ Table decoration: *fill*, *firstdraw*, *gstate*, *round*, *stroke*
- ▶ Visualization aids for development and debugging: *debugshow*, *showcells*, *showgrid*
- ▶ Option for abbreviated structure element tagging according to Table 14.2 (only allowed in *page* scope): *tag*. This option can be used to trigger automatic table tagging (see PDFlib Tutorial for details).

Returns A string which specifies the reason for returning from the method:

- ▶ *_stop*: all rows in the table have been processed.
- ▶ *_boxfull*: there are still rows to be placed, but not enough space is available in the table's fitbox; another call to *PDF_fit_table()* is required for processing the remaining rows.
- ▶ *_error*: an error occurred; call *PDF_get_errmsg()* to obtain details about the problem and set *debugshow=true* to visualize the problem.
- ▶ Any other string: the string supplied to the *return* option in a call to *PDF_add_table_cell()*.

The error behavior can be changed with the *errorpolicy* option.

Details Place the table on the page. The table cells must have been filled with prior calls to *PDF_add_table_cell()*. If the full table doesn't fit in the fitbox, the first table instance will be placed; more table instances can be placed with subsequent calls to this method depending on the return value. The contents of a table cell are placed in the following order:

- ▶ Filling: the areas specified with the *fill* option are filled in the following order: *table*, *colother*, *colodd*, *coleven*, *col#*, *collast*, *rowother*, *rowodd*, *roweven*, *row#*, *rowlast*, *header*, *footer*.
- ▶ Matchbox filling: areas which are defined by a *matchbox* definition.
- ▶ Contents: the specified cell contents are placed in the following order: image, graphics, imported PDF page, graphics, path objects, Textflow, Textline, annotations, form fields.
- ▶ Matchbox ruling: areas which are defined by a *matchbox* definition.
- ▶ Ruling: the lines specified with the *stroke* option are stroked according to the *linecap* and *linejoin* suboptions of the *stroke* option in the following order: *other*, *horother*, *hor#*, *horlast*, *vertother*, *vert#*, *vertlast*, *frame* (the order of horizontal and vertical lines can be changed with the *firstdraw* option). Cells which span multiple rows or columns are not intersected by strokes. Similarly, lines are not stroked around cells with a *matchbox* which specifies border decoration (unless the matchbox uses the inner cell box). The table border lines *verto*, *horo*, *vertN*, and *horN* are suppressed if *frame* is specified.
- ▶ Named matchboxes: these can be filled with other elements like annotations, form fields, images, graphics etc. outside of the table methods.

Automatic table tagging: the option *tag={tagname=Table}* can be used to automatically create structure elements for table rows and cells (see PDFlib Tutorial).

PDF/UA Generally *page, pattern, template, glyph*; however, if the table contains form fields, annotations or tags only *page* scope is allowed.

Table 5.19 Options for `PDF_fit_table()`

option	description
blind	(Boolean) If true, all calculations will be performed, but no output will be created. The formatting results can be checked with <code>PDF_info_table()</code> . Default: false
caption	(Option list) Create a fit box for a caption relative to the calculated fit box and fill it with various content types. The following option can be supplied (default: no caption): fitbox (List of four floats or percentages with absolute or relative coordinates; required) Coordinates of two diagonal box corners in user coordinates. If a value is a percentage or a relative value it indicates the offset from the corresponding corner {llx lly urx ury} of the table instance. Percentages corresponding to llx or urx are percentages of the table instance width, percentages corresponding to lly or ury are percentages of the table instance height. The fitbox is not automatically adjusted to the size of its contents. The specified matchbox will describe the fitbox; this can be used to draw the caption fitbox or to retrieve the matchbox with <code>PDF_info_matchbox()</code> . Examples for using the fitbox option: Fit box at the top of the table instance with a height of 20: <code>fitbox={0r 100% 0r 20r}</code> Fit box to the right of the table instance with width 20 and offset 20% from the bottom: <code>fitbox={100% 20% 20r 0r}</code> In addition, the following options are supported: <ul style="list-style-type: none"> ▶ Options for static cell contents according to Table 5.17: <code>fitgraphics</code>, <code>fitimage</code>, <code>fitpath</code>, <code>fitpdipage</code>, <code>fittextflow</code>, <code>fittextline</code>, <code>graphics</code>, <code>image</code>, <code>matchbox</code>, <code>path</code>, <code>pdipage</code>, <code>text</code>, <code>textflow</code> ▶ Options for interactive cell contents according to Table 5.18 (only in page scope): <code>annotationtype</code>, <code>fieldname</code>, <code>fieldtype</code>, <code>fitannotation</code>, <code>fitfield</code> ▶ Option for abbreviated structure element tagging according to Table 14.2: <code>tag</code>. This can be used for inserting a parent element of the caption contents, or a grouping element as container for multiple elements which comprise the caption contents.
colwidth-default	(Float or keyword; only relevant in the first call to <code>PDF_fit_table()</code>) Default width for columns which do not contain any <code>Textline</code> or <code>Textflow</code> cells and for which the <code>colwidth</code> option of <code>PDF_add_table_cell()</code> was not specified. The default width can be specified as an absolute value or as a keyword. The value 0 (zero) is equivalent to the keyword <code>distribute</code> . The following keywords are supported (default: <code>auto</code>): auto Columns with unspecified width which contain only <code>Textline</code> cells have the width of the text. The remaining width of the fitbox is distributed among all columns with non- <code>Textline</code> cells with unspecified width. If such columns are present the table covers the full width of the fitbox. distribute The width of the fitbox is distributed among all columns with unspecified width which don't contain any <code>Textline</code> cells. If such columns are present the table covers the full width of the fitbox. minimum Columns with unspecified width which contain only <code>Textline</code> cells have the width of the text, i.e. the smallest possible width to hold the text. In order to create columns with minimal width you can supply a small value (e.g. 1). The width of all columns which contain <code>Textline</code> or <code>Textflow</code> cells is adjusted automatically (see <code>PDFlib Tutorial</code>).
debugshow	(Boolean) If true, all errors for tables which are too high or too wide, or where the cells get too small are suppressed and logged instead. The resulting table instance is created as a debugging aid although the table is damaged (i.e. the table may be incomplete or incorrect). Default: false

Table 5.19 Options for `PDF_fit_table()`

option	description
fill	<p>(List of option lists) This option can be used to fill rows or columns with color (the <code>matchbox</code> option can be used to fill single cells with color, see Section 6.2, »Matchboxes«, page 131):</p> <p>area (Keyword) Table area(s) to be filled:</p> <ul style="list-style-type: none"> col# column number # in the table collast last column coleven all columns with even numbers (according to <code>col</code> in <code>PDF_add_table_cell()</code>) colodd all columns with odd numbers colother all unspecified columns row# row number # in the table rowlast last body row in the table instance roweven all rows with even numbers (according to <code>row</code> in <code>PDF_add_table_cell()</code>) rowodd all rows with odd numbers header all rows in the header group footer all rows in the footer group rowother all unspecified body rows table complete table area (i.e. all rows in the table) <p>The following graphics appearance options according to Table 7.1 can also be used: fillcolor, shading Use <code>fillcolor=none</code> to suppress filling in a specified table area. Examples: fill all rows in the table with red: <code>fill = { {area=table fillcolor=red} }</code> fill odd-numbered rows with green and even-numbered rows with red: <code>fill = { {area=rowodd fillcolor=green} {area=roweven fillcolor=red} }</code></p>
firstdraw	<p>(Keyword) Order in which horizontal and vertical lines will be created (default: <code>vertlines</code>):</p> <ul style="list-style-type: none"> horlines Horizontal lines will be created first. vertlines Vertical lines will be created first.
footer	<p>(Integer) Number of final (footer) rows in the table definition which will be repeated at the bottom of the table instance. Default: 0 (no footer rows)</p>
gstate	<p>(Option list with options for <code>PDF_create_gstate()</code> or <code>gstate handle</code>) Graphics state options or handle for table decorations (but not cell contents). Default: no graphics state, i.e. current settings are used</p>
header	<p>(Integer) Number of initial (header) rows in the table definition which will be repeated at the top of the table instance. Default: 0 (no header rows)</p>
horshrinklimit	<p>(Float or percentage) Lower limit for the horizontal shrinking factor which will be used when the table is shrunk to fit in the table's fitbox (if a percentage is supplied) or the absolute difference between the table width and the width of the fitbox (if a float is supplied). Default: 50%</p>
rewind	<p>(Integer: -1, 0, or 1) The state of the table is reset to the state before some other call to <code>PDF_fit_table()</code> with the same table handle and then the table is placed as usual (default: 0):</p> <ul style="list-style-type: none"> 1 Rewind to the state before the first call to <code>PDF_fit_table()</code>. 0 Don't reset the table. -1 Rewind to the state before the previous call to <code>PDF_fit_table()</code>
round	<p>(Float) If this value is different from 0 the corners of the table rectangle for the <code>fill</code> and <code>stroke</code> options are rounded with a circular arc with the specified radius. If the radius is negative the arc segments are swept inwards. Default: 0, i.e. no rounding</p>

Table 5.19 Options for `PDF_fit_table()`

option	description
rowheight-default	<p>(Float or keyword; only relevant in the first call to <code>PDF_fit_table()</code>) Default height of rows for which the <code>rowheight</code> option of <code>PDF_add_table_cell()</code> was not specified. The default height can be specified as an absolute value or as a keyword. If a float value is specified it is used as default row height unless it is smaller than the textbox height. The value 0 (zero) is equivalent to the keyword <code>distribute</code>. The following keywords are supported (default: <code>auto</code>):</p> <p>auto Rows with unspecified height which contain only Textline cells have a height of two times the height of the textbox. The remaining height of the fitbox is distributed among all rows with non-Textline cells with unspecified height. If such rows are present the table covers the full height of the fitbox.</p> <p>distribute The height of the fitbox is distributed among all Textline and other rows with unspecified height. If such rows are present the table covers the full height of the fitbox.</p> <p>minimum Rows with unspecified height which contain only Textline cells have the height of the textbox, i.e. the smallest possible height to hold the text. Use the <code>boxsize</code> or <code>margin</code> options to increase the height of Textline cells.</p> <p>In order to create rows with minimal height you can supply a small positive value (e.g. 1). The height of all rows which contain Textline or Textflow cells is adjusted automatically (see PDFlib Tutorial).</p>
showcells	<p>(Boolean) If <code>true</code>, the border of each inner cell box will be stroked using the current graphics state. In page scope and if PDF/A is not active each cell is additionally decorated with an annotation with details describing the cell contents which may be helpful for analyzing table-related problems. Default: <code>false</code></p>
showgrid	<p>(Boolean) If <code>true</code>, the vertical and horizontal boundary of all columns and rows are stroked. Default: <code>false</code></p>
stroke	<p>(List of option lists) This option can be used to create stroked lines at the cell borders:</p> <p>line (Keyword) Table line(s) to be stroked:</p> <p>vert# vertical line at the right border of column number #; <code>vert0</code> is the left table border</p> <p>vertfirst first vertical line (equivalent to <code>vert0</code>)</p> <p>vertlast last vertical line</p> <p>vertother all unspecified vertical lines</p> <p>hor# horizontal line at the bottom of row number # in the table; <code>hor0</code> is the top border</p> <p>horfirst first horizontal line in the table instance</p> <p>horother all unspecified horizontal lines</p> <p>horlast last horizontal line in the table instance</p> <p>frame outer border of the table</p> <p>other all unspecified lines</p> <p>The following graphics appearance options according to Table 7.1 can also be used: dasharray, dashphase, linecap, linejoin, linewidth, strokecolor Use <code>strokecolor=none</code> or <code>linewidth=0</code> to suppress stroking in a specified table area.</p> <p>Examples: stroke all lines with black and <code>linewidth 1</code>: <code>stroke = {line=other}</code> stroke the outer border lines with <code>linewidth 0.5</code>: <code>stroke = { {line=frame linewidth=0.5} }</code> stroke the outer border lines with <code>linewidth 0.5</code>, and all other lines with <code>linewidth 0.1</code>: <code>stroke = { {line=frame linewidth=0.5} {line=other linewidth=0.1} }</code></p>
vertshrink-limit	<p>(Float or percentage) Lower limit for the vertical shrinking factor which will be used when the table is shrunk to fit the table's fitbox (if a percentage is supplied) or the absolute difference between the height of the table instance and the height of the fitbox (if a float is supplied). Default: <code>90%</code></p>

C++ Java C# **double** *info_table*(int table, String keyword)

Perl PHP **float** *info_table*(int table, string keyword)

C **double** *PDF_info_table*(PDF *p, int table, const char *keyword)

Retrieve table information related to the most recently placed table instance.

table A valid table handle retrieved with a call to *PDF_add_table_cell()*. The table handle must already have been used in at least one call to *PDF_fit_table()* since the returned values are meaningful only after placing a table instance on the page.

keyword A keyword specifying the requested information:

- ▶ Keywords for querying the results of object fitting according to Table 6.3:
boundingbox, fitscalex, fitscaley, height, objectheight, objectwidth, width, x1, y1, x2, y2, x3, y3, x4, y4
- ▶ Additional keywords according to Table 5.20:
firstbodyrow, horboxgap, horshrinking, lastbodyrow, returnreason, rowsplit, tableheight, tablewidth, vertboxgap, vertshrinking, xvertline#, yhorline#,

Returns The value of some table parameter as requested by *keyword*. This method returns correct geometry information even in blind mode. If the requested keyword produces text, a string index is returned, and the corresponding string must be retrieved with *PDF_get_string()*.

Scope any except object

Table 5.20 Keywords for *PDF_info_table()*

keyword	description
firstbodyrow	Number of the first body row in the most recently placed table instance
horboxgap	Difference between the width of the table instance and the width of the fitbox. If the table had to be shrunk the value will specify the deviation from the width of the fitbox (i.e. a negative value).
horshrinking	Horizontal shrinking factor as a percentage of the calculated table width. If the table had to be shrunk horizontally the value will specify the shrinking percentage, otherwise it will be 100.
lastbodyrow	Number of the last body row in the most recently placed table instance
returnreason	String index for the return reason
rowcount	Number of rows in the most recently placed table instance (including headers and footers)
rowsplit	1 if the last row had to be split, 0 otherwise
tableheight tablewidth	Width and height of the entire table
vertboxgap	Difference between the height of the most recently generated table instance and the height of the fitbox. If the table had to be shrunk, the value will specify the deviation from the height of the fitbox (i.e. a negative value).
vert-shrinking	Vertical shrinking factor as a percentage of the calculated table height. If the table had to be shrunk vertically the value will specify the shrinking percentage, otherwise it will be 100.
xvertline#	x coordinate of the vertical line with number #. xvertline0 is the left table border.
yhorline#	y coordinate of the horizontal line with number #. yhorline0 is the top table border.

C++ Java C# **void delete_table(int table, String optlist)**

Perl PHP **delete_table(int table, string optlist)**

C **void PDF_delete_table(PDF *p, int table, const char *optlist)**

Delete a table and all associated data structures.

table A valid table handle retrieved with a call to *PDF_add_table_cell()*.

optlist An option list specifying cleanup options according to Table 5.21.

Details Tables which have not been deleted with this method will be deleted automatically at the end of the enclosing *document* scope.

Scope any

Table 5.21 Option for *PDF_delete_table()*

option	description
keephandles	(Boolean) If false, all handles supplied to the textflow, image, graphics and pdipage options of <i>PDF_add_table_cell()</i> will automatically be deleted. Default: false

6 Object Fitting and Matchboxes

API method in this chapter:

- ▶ `PDF_info_matchbox()`

6.1 Object Fitting

PDFlib's fitting algorithm places a rectangular graphical object relative to a point, a horizontal or vertical line, or a rectangle. The fitting algorithm is implemented in several methods:

- ▶ `PDF_fit_textline()`, `PDF_info_textline()`
- ▶ `PDF_fit_image()`, `PDF_info_image()`
- ▶ `PDF_fit_graphics()`, `PDF_info_graphics()`
- ▶ `PDF_fit_pdi_page()`, `PDF_info_pdi_page()`
- ▶ `PDF_draw_path()`, `PDF_info_path()`
- ▶ `PDF_add_table_cell()` (via option lists for the `fitgraphics`, `fitimage`, `fitpdi`, `fitpath`, `fittextline` options)
- ▶ `PDF_fit_table()`
- ▶ `PDF_fill_*block()`

Note Since the fitting options for `Textflow` are slightly different they are not described here, but in Section 5.2, »Multi-Line Text with `Textflows`«, page 96.

Table 6.1 lists fitting options which can be supplied to the fitting methods. Not all options are available for all methods, and the behavior of some options may slightly change depending on the method; see Table 6.1 for details. The following options form the group of fitting options:

alignchar, boxsize, dpi, fitmethod, margin, matchbox, minfontsize, orientate, position, reappoint, rotate, scale, stamp, showborder, shrinklimit

Object box. The fitting algorithm calculates the smallest enclosing rectangle of the placed object. This rectangle is called the *object box*. It can be modified according to the type of object:

- ▶ Textlines (`PDF_fit/info_textline()`, single-line text Blocks, table cells): The width is the width of the text string (in horizontal writing mode) or the width of the widest glyph (in vertical writing mode). The default height of the text box is the *capheight* of the selected font. This can be changed with the *boxheight* suboption of the *matchbox* option. Character spacing will not be applied after the last glyph.
- ▶ Images and templates (`PDF_fit/info_image()`, image Blocks, table cells): the suboption *clipping* of the *matchbox* option can be used to define some part of the object as object box. For TIFF and JPEG images with a clipping path the smallest enclosing rectangle with edges parallel to the coordinate axes will be used as object box if the suboption *innerbox* of the *matchbox* option is set. If the *transform* option of `PDF_begin_template_ext()` has been supplied, the specified transformation is applied to the template.
- ▶ Graphics (`PDF_fit/info_graphics()`): the suboption *clipping* of the *matchbox* option can be used to define some part of the object as object box. The object box is defined by the width and height of the SVG graphics or by *forcedwidth* and *forcedheight*. If these values are 0 the following holds: if *fitmethod* is different from *nofit* or the *fitbox* is not defined, the size of the object box is defined by *fallbackwidth* and *fallbackheight*. If

fitmethod=nofit and the fitbox is defined, the size of the object box is defined by the fitbox.

- ▶ Imported PDF pages (*PDF_fit/info_pdi_page()*, PDF Blocks, table cells): the options used in *PDF_open_pdi_page()* are honored. If *cloneboxes=true* the visible box is used (i.e. the CropBox if present, else the MediaBox). If the *transform* option of *PDF_open_pdi_page()* has been supplied, the specified transformation is applied to the imported page. The suboption *clipping* of the *matchbox* option can be used to define some part of the object as object box.
- ▶ Path objects (*PDF_draw/info_path()*, table cells): the smallest rectangle with edges parallel to the coordinate axes which encloses the path will be used as object box. The object box will only be calculated if the *boxsize* and *position* options have values different from zero. The *linewidth* and *miterlimit* options are ignored.
- ▶ Table instances (*PDF_fit_table()*): the smallest rectangle with edges parallel to the coordinate axes which encloses the table instance will be used as object box.

Reference point. The *reference point* is used as an anchor for placing the object box. It is defined as follows:

- ▶ In *PDF_fit_**() and *PDF_draw_path()*: the *x* and *y* method parameters;
- ▶ In *PDF_info_**(): the point (*o*, *o*);
- ▶ *PDF_add_table_cell()*, *PDF_fit_table()*, and *PDF_fill_*block()*: the lower left corner of the table cell, table instance, or PDFlib Block;

The following methods additionally support the *repoint* option for overriding the reference point: *PDF_fit_graphics()*, *PDF_info_path()*, *PDF_fill_*block()*.

Fitbox and reference line segment. The rectangle in which the object box will be placed is called the *fitbox*. It has the reference point (*x*, *y*) as its lower left corner and its size is specified by the two values of the *boxsize* option:

lower left corner = (*x*, *y*)
upper right corner = (*x* + *boxsize*[0], *y* + *boxsize*[1]) (if *topdown=false*)
upper right corner = (*x* + *boxsize*[0], *y* - *boxsize*[1]) (if *topdown=true*)

In addition to the definition above the fitbox can be modified as follows:

- ▶ Textlines: the fitbox can be reduced with the *margin* option;
- ▶ table cells: the fitbox is defined by the inner cell box, i.e. the cell box as modified by the *margin** options;
- ▶ table instances: the fitbox is defined by the *llx/lly/urx/ury* parameters;
- ▶ PDFlib Blocks: the fitbox is by default defined by the Block's *Rect* property, but it can be modified with the *repoint* and/or *boxsize* options.

In the last three cases above the fitbox is always available; otherwise it is only available if the *boxsize* option was specified with two values different from zero.

If *boxsize*[0]=0 the box degenerates to a vertical line. The fitting algorithm will place the object box relative to this line segment. Similarly, if *boxsize*[1]=0 the box will be placed relative to the resulting horizontal line segment. The vertical or horizontal line segment is called the *reference line segment*.

Placing the object box. The object box can be placed in different ways:

- ▶ If no fitbox is available the object will be placed relative to the reference point (not for table cells, table instances, and PDFlib Blocks): the lower left corner of the object

box will coincide with the reference point. Using the *position* option other points within the object box can be selected. For example, *position=center* places the object box's center point at the reference point.

The option *scale* will be honored for images, graphics, templates, path objects, and imported PDF pages; the option *dpi* will be honored for images. The *fitmethod* option is ignored in this case.

Path objects: if *position={o o}* the bounding box will not be calculated and the origin of the path object will coincide with the reference point.

- ▶ Relative to a reference line segment (not for table cells, table instances, and PDFlib Blocks): this works similarly to placing an object relative to the reference point as described above. In addition, the *position* option also defines a point on the line segment which will serve as reference point.
- ▶ Relative to the fitbox: The *fitmethod* option specifies whether and how the object box will be forced to fit into the fit box. If *fitmethod=nofit* nothing will be done to restrict the result to the fitbox. Other values of *fitmethod* define details of the fitting algorithm according to Table 6.2.

In this case the options *scale* and *dpi* are ignored, and the options *margin*, *shrinklimit*, and *showborder* are honored.

The lower left corner of the object box will coincide with the lower left corner of the fitbox. Using the *position* option other points within the object box and simultaneously the corresponding point within the fitbox can be selected. For example, *position=center* places the object box's center point at the center point of the fitbox.

Table 6.1 Fitting options for various methods

option	description
align	(List of two floats; only for path objects) The coordinates of a direction vector in user coordinates which defines the rotation of the path object. The x direction of the path object's coordinate system will be aligned with the specified vector. The coordinates must not both be 0. The calculated rotation will be added to the rotation defined by the orientate option. Default: {1 0}, i.e. no additional rotation
alignchar	(Unichar < 0xFFFF or keyword; only for Textlines) If the specified character is found in the text, its lower left corner will be aligned at the reference point. For horizontal text with orientate=north or south the first value supplied in the position option defines the position. For horizontal text with orientate=west or east the second value supplied in the position option defines the position. If this option is present the formatted text may exceed beyond the fitbox. This option is ignored if the specified alignment character is not present in the text. If the specified character cannot be found in the font or encoding, an exception will be thrown if glyphcheck=error. For other values of glyphcheck the alignchar option will silently be ignored if the character is not available. The value 0 and the keyword none suppress alignment characters. The specified fitmethod will be applied, although the text cannot be placed within the fitbox because of the forced positioning of alignchar. Default: none
attachment-point	(String; only for path objects) Name of the attachment point. The path object will be placed so that the specified attachment point coincides with the reference point. If fitmethod is different from nofit the object will first be placed in the fitbox according to the specified method. Default: origin of the path object
blind	(Boolean; see Table 5.13 for Textflow) If true, no output will be generated, but all calculations will be performed and the formatting results can be checked with the appropriate info method PDF_info_*(). Default: false

Table 6.1 Fitting options for various methods

option	description
boxsize	<p>(List of two floats; not for tables) Width and height of the fitbox, relative to which the object (possibly rotated according to the rotate option) will be placed. The lower left corner of the fitbox coincides with the reference point (x, y). Placing the object is controlled by the position and fitmethod options. If width=0, only the height is considered; if height=0, only the width is considered. In these cases the fitmethod option is ignored and the object is placed relative to the vertical line from (x, y) to (x, y+height) (or (x, y-height) for topdown systems), or the horizontal line from (x, y) to (x+width, y) according to the position option.</p> <p>Default for PDF_fill_block(): width and height of the Block's Rect property. These methods accept absolute coordinates as well as relative coordinates which relate to the original width or height of the Block. Default for all other fitting methods: {0 0}</p>
dpi	<p>(List of two floats or keywords; only for images) One or two values specifying the desired image resolution in pixels per inch in horizontal and vertical direction. This option does not change the number of pixels in the image (downsampling). If a single value is supplied it is used for both dimensions. With the value zero the image's internal resolution is used if available, or 72 dpi otherwise. The keyword internal is equivalent to zero. The scaling resulting from this option is relative to the current user coordinate system; if the coordinate system has been scaled the resulting physical resolution is different from the supplied values. The scale option will be applied in addition to the dpi values.</p> <p>If the fitmethod option has been supplied with one of the keywords auto, meet, slice, or entire, the dpi values specify only the image's aspect ratio, but not its absolute size. Default: internal</p>
fitmethod	<p>(Keyword; see Table 5.13 for Textflow) Method used to fit the object into the specified fitbox. See Table 6.2 for supported keywords. Keywords other than nofit are ignored if no fitbox has been specified.</p> <p>Default: clip for tables; meet for path objects; and nofit otherwise</p>
margin	<p>(List of floats; only for Textlines) One or two float values describing additional horizontal and vertical reductions of the fitbox. Default: 0</p>
matchbox	<p>(Option list; not for path objects; see Table 5.13 for Textflow) Option list for creating a matchbox according to Table 6.4</p>
minfontsize	<p>(Float or percentage; see Table 5.13 for Textflow) Minimum allowed font size when text is scaled down to fit into the fitbox with fitmethod=auto when shrinklimit is exceeded. The limit is specified in user coordinates or as a percentage of the height of the fitbox. If the limit is reached the text will be created with the specified minfontsize as fontsize. Default: 0.1%</p>
orientate	<p>(Keyword or float; not for tables; see Table 5.13 for Textflow) Orientation of the object relative to the current coordinate system. Default: north.</p> <p>Arbitrary rotation angles (in degrees) can be specified for path objects, but not other object types. The bounding box of the path object will be calculated after rotating the path object. All methods support the following keywords (corresponding equivalent angles are shown in parentheses):</p> <ul style="list-style-type: none"> north upright (0) east pointing to the right (270) south upside down (180) west pointing to the left (90)

Table 6.1 Fitting options for various methods

option	description
position	<p>(List of floats or keywords) One or two values specifying the position of the object box relative to the reference point, the reference line segment, or the fitbox. The values specify a position within the object box. This position is defined horizontally as percentage of the box width (first value) and vertically as percentage of the box height (second value). This specified position coincides with the reference point, a point on the reference line segment or a point within the fitbox. Although the values designate percentages, they must be specified without any percent sign. Negative values are allowed. If both values are equal, it is sufficient to specify a single value.</p> <p>Default: {0 100} for tables, otherwise {0 0}. Examples:</p> <p>{0 0} The lower left corner of the object box coincides with the reference point, the start of the reference line segment, or the lower left corner of the fitbox.</p> <p>{100 100} The upper right corner of the object box coincides with the reference point, the end of the reference line segment, or the upper right corner of the fitbox.</p> <p>The keywords left, center, right (in x direction) or bottom, center, top (in y direction) can be used as equivalents for the values 0, 50, and 100. If only one keyword has been specified, the corresponding keyword for the other direction will be added. Examples:</p> <p>{left center} or {0 50} left-aligned {center} or {50 50} centered {right center} or {100 50} right-aligned</p> <p>Only for Textlines: the keyword auto can be used for the first value in the list. It indicates right if the writing direction of the text is from right to left (e.g. for Arabic and Hebrew text), and left otherwise</p>
refpoint	<p>(List of two floats; only for PDF_fit_graphics(), PDF_fill_block() and PDF_info_path()) Reference point in user coordinates for fitting the Block contents or path.</p> <p>Default for PDF_fill_block(): lower left corner of the rectangle defined by the Block's Rect property. These methods accept absolute coordinates as well as relative coordinates which relate to the lower left corner of the Block. Example with relative coordinates: refpoint={20r 50r}</p> <p>Default for PDF_info_path(): {0 0}</p> <p>PDF_fit_graphics(): the x and y parameters</p>
rotate	<p>(Float; not for tables and path objects; see Table 5.13 for Textflow) Rotate the coordinate system, using the reference point as center and the specified value as rotation angle in degrees. This results in the fitbox and the object being rotated. The rotation will be reset when the object has been placed. Default: 0</p> <p>Textline in table cells: if the rotate option was specified with a value different from 0, the table engine attempts to fit the bounding box of the rotated text into the cell box according to the fitmethod and position options. If fitmethod is different from auto, the cell will be enlarged appropriately if necessary.</p>
scale	<p>(List of floats; not for Textlines; ignored for fitmethod=meet) Scales the object in horizontal and vertical direction by the specified scaling factors (not percentages), using the reference point as center. If both factors are equal it is sufficient to specify a single value. Negative values result in mirroring. The absolute value of this option ignored if the fitmethod option has been supplied with one of the keywords auto, meet, slice, or entire. Default: {1 1}</p>
showborder	<p>(Boolean; see Table 5.13 for Textflow) If true, the border of the fitbox will be stroked using the current graphics state. If a stamp is created, the bounding box of the stamp will also be stroked. This may be useful for development and debugging. Default: false</p>

Table 6.1 Fitting options for various methods

option	description
shrinklimit	(Float or percentage; only for Textlines) The lower limit of the shrinkage factor which will be applied to fit text with fitmethod=auto. Default: 0.75
stamp	<p>(Keyword; only for Textlines; ignored if boxsize is not specified; see Table 5.13 for Textflow) This option can be used to create a diagonal stamp of maximal size in the rectangle specified with the boxsize option. More specifically, the text will be placed diagonally in the fitbox. The size of the text box will be chosen so that it covers the fitbox as much as possible while preserving the aspect ratio of the text box (i.e. the text comprising the stamp will be as large as possible). The options fontsize, fitmethod, and position are ignored. The options orientate=west and =east don't make any sense (only north and south). Supported keywords (default: none):</p> <p>llzur The stamp runs diagonally from the lower left corner to the upper right corner.</p> <p>ulzlr The stamp runs diagonally from the upper left corner to the lower right corner.</p> <p>none No stamp will be created.</p>

Table 6.2 Keywords for the fitmethod option of various methods; the illustrations demonstrate the typical effect of each keyword on a Textline, using the same value for the fontsize option in all examples.

keyword	description	
auto	<p>This method tries to fit the object box into the fitbox automatically: If the object fits into the fitbox the behavior is identical to the nofit method, i.e. the object is placed without any scaling. If the object is larger than the fitbox the object is proportionally reduced in size as follows:</p> <ul style="list-style-type: none"> ▶ Textlines: a scaling factor is calculated such that the text can be shrunk horizontally (distorted) to fit into the fitbox. If the calculated factor is smaller than the shrinklimit option, the meet method is applied by reducing the fontsize until the text can be fit or the fontsize 0.001 is reached. ▶ PDF_fit_table(): If the table box is narrower than the fitbox it is enlarged to the fitbox width. Otherwise the behavior is identical to the meet method. ▶ Other object types: the behavior is identical to the meet method. <p>In other words, with fitmethod=auto objects may be reduced in size, but never enlarged.</p>	
clip	<p>Position the object and graphically clip it at the edges of the fitbox. PDF_fit_table(): the calculated table box will be logically clipped at the bottom edge of the fitbox and can be continued in the next fitbox. Logical clipping is similar to PDF_fit_textflow(), but not graphical clipping as in PDF_fit_image() etc. The table box will be placed inside the fitbox according to the position option.</p>	
entire	<p>Scale the object box such that it entirely covers the fitbox. Generally this method will distort the object. The position option doesn't have any effect. PDF_fit_table(): similar to clip. If the table box is smaller than the fitbox, the cells of the table box (but not their contents) will be enlarged uniformly until the table box entirely covers the fitbox.</p>	
meet	<p>Position the object according to the position option and scale it such that it entirely fits into the fitbox while preserving its aspect ratio. Generally at least two edges of the object box meet the corresponding edges of the fitbox. PDF_fit_table(): similar to clip. If the table box is smaller than the fitbox, the cells of the table box (but not their contents) are enlarged uniformly until the horizontal or vertical table edge meets the fitbox.</p>	
nofit	<p>Position the object only. The scale option is applied to images and graphics, for images also the dpi option. PDF_fit_table(): The table will be calculated for a virtual fitbox with infinite height. The table box will be placed inside the fitbox according to the position option. The default sizes of columns and rows relate to the specified fitbox height. fitmethod=nofit is recommended to format the table in blind mode.</p>	
slice	<p>Position the object according to the position option and scale it such that it entirely covers the fitbox, while preserving the aspect ratio and making sure that at least one dimension of the object is fully contained in the fitbox. Generally parts of the object's other dimension will extend beyond the fitbox, and will therefore be clipped. PDF_fit_table(): similar to clip. If the table box is smaller than the fitbox the cells of the table box (but not their contents) will be enlarged uniformly until the fitbox is entirely covered by the table box while preserving its aspect ratio. The table box will be placed inside the fitbox according to the position option. The parts of the table box which exceed beyond the fitbox will be clipped graphically at the edges of the fitbox.</p>	

Common keywords for querying the results of object fitting. The results of object fitting can be queried without actually placing the object on the page. This can be used to make formatting decisions before actually creating page content. In order to query formatting results the fitting options for an object can be supplied to the respective `PDF_info_*` method. Table 6.3 lists keywords for querying fitting results. The fitting results for `PDF_info_path()` are expressed relative to the reference point.

Table 6.3 Common keywords for querying the results of object fitting with `PDF_info_image()`, `PDF_info_graphics()`, `PDF_info_path()`, `PDF_info_pdi_page()`, `PDF_info_table()`, `PDF_info_textline()`

keyword	description
boundingbox	Path handle for the object's bounding box
fitscalex, fitscaley	Scaling factors which resulted from fitting the object to a box.
height	Object height in user coordinates
objectheight, objectwidth	Raw size of the object after processing all options relevant for loading or creating the object. This size will be used by the fitting algorithm.
width	Object width in user coordinates
x1, y1, x2, y2, x3, y3, x4, y4	Position of the <i>i</i> -th rectangle corner (<i>i</i> =1, 2, 3, 4) of the object's bounding box in user coordinates according to the supplied options. x1, y1 correspond to the lower left, x2, y2 to the lower right, x3, y3 to the upper right and x4, y4 to the upper left corner.

6.2 Matchboxes

Matchboxes are rectangular areas which hold the result of a formatting process, e.g. placing text or an image. They can be used to add decoration like stroked border or filled interior to the area. You can also query the matchbox position and use it for other purposes such as creating annotations. Matchboxes can also be used for clipping an object.

6.2.1 Defining Matchboxes

Matchboxes are not defined with a dedicated API method, but with the *matchbox* option in the formatting method which creates the corresponding element:

- ▶ Textlines with `PDF_fit_textline()`, `PDF_fill_textblock()` with `textflow=false`: the matchbox describes the bounding box of the text line.
- ▶ Textflows with `PDF_add/create_textflow()`, `PDF_fit_textflow()`, `PDF_fill_textblock()` with `textflow=true`: the matchbox describes the bounding box of the generated text output. Matchbox specifications in `PDF_fill_textblock()` cannot be used as start for inline text decorations, but only for creating a matchbox for the whole text.
- ▶ imported PDF pages with `PDF_fit_pdi_page()`, `PDF_fill_pdfblock()`: the matchbox describes the bounding box of the placed page.
- ▶ images and templates with `PDF_fit_image()`, `PDF_fill_imageblock()`: the matchbox describes the bounding box of the placed image or template.
- ▶ graphics with `PDF_fit_graphics()`, `PDF_fill_graphicsblock()`: the matchbox describes the bounding box of the placed graphics.
- ▶ table cells: `PDF_add_table_cell()`: the matchbox describes the bounding box of the table cell.

Matchboxes are defined with the *matchbox* option of these methods and can be used until the end of the page where the matchbox is defined. A matchbox can be identified by its name and may consist of one or more rectangle(s). The *matchbox* option expects an option list which supports the following suboptions:

- ▶ Graphics appearance options according to Table 7.1:
borderwidth, *dasharray*, *dashphase*, *fillcolor*, *gstate*, *linecap*, *linejoin*, *shading*, *strokecolor*
- ▶ Matchbox control options according to Table 6.4;
- ▶ Option for abbreviated structure element tagging according to Table 14.2 (only allowed in *page* scope): *tag*

Note Matchboxes are not supported in blind mode, i.e. formatting with the blind option.

6.2.2 Using Matchboxes

Matchboxes can be used for the following purposes:

- ▶ Decoration: a rectangle defined by a matchbox is filled if the option(s) *fillcolor* or *shading* are specified. The border of a matchbox is stroked if the option *strokecolor* is specified and option *borderwidth* has a value > 0 . Matchbox decoration is especially useful for table cells.
- ▶ Clipping: the size of placed images, graphics or PDF pages can be modified with the *clipping* suboption of the *matchbox* option.
- ▶ Text wrapping: matchboxes can be used for Textflow wrapboxes with `PDF_fit_textflow()` and option *wrap*, suboption *usematchboxes*.
- ▶ Interactive elements: matchboxes can be used as target rectangle for annotations or form fields with `PDF_create_annotation()` and the option *usematchbox*.

- ▶ Fit path objects into a matchbox with `PDF_draw_path()` and the option `usematchbox`.
- ▶ Details of the rectangle(s) corresponding to a matchbox can be queried with `PDF_info_matchbox()`, e.g. to query the result of a formatting operation.

Table 6.4 Suboptions for the matchbox option of various methods

option	description
boxheight	(List with two elements, each being a positive float, a percentage of the fontsize, or a keyword; only for <code>Textline</code> and <code>Textflow</code>) Vertical extent of the text box. Two values can be specified numerically or via keywords for the extent above and below the baseline: none (no extent), <code>xheight</code> , <code>descender</code> , <code>capheight</code> , <code>ascender</code> , <code>fontsize</code> , <code>leading</code> , <code>textrise</code> With <code>Textflows</code> the values corresponding to the text at the beginning of the matchbox will be used. Default: { <code>capheight none</code> }
boxwidth	(Float or percentage; only for <code>Textflow</code>) Width of the matchbox specified in user coordinates or as a percentage of the width of the fitbox. If this option is supplied, horizontal space of the specified width is inserted between the matchbox option and the next text fragment or the matchbox end specification. This may be useful to reserve space for inserting an image, template, or PDF page in the <code>Textflow</code> . Note that with <code>alignment=justify</code> the box width may be compressed the same way as text (see option <code>shrink-limit</code>). Default: 0
clipping	(Rectangle or 4 percentages; only for images, graphics and imported PDF pages; ignored if the <code>innerbox</code> option has been specified) Coordinates of the lower left and upper right corner of a rectangle within the image, graphics or page specifying which part should be displayed. The specification depends on the type of object (default: {0% 0% 100% 100%}): <ul style="list-style-type: none"> ▶ For images the clipping rectangle can be specified in pixels or as a percentage of the width/height. ▶ For graphics the clipping rectangle can be specified in coordinates of the graphics or as a percentage of the width/height of the graphics' object box. ▶ For PDF pages the clipping rectangle can be specified in default units or as a percentage of the width/height of the page's crop box.
create-wrapbox	(Boolean; only for <code>Textflow</code>) If <code>true</code> , the rectangle(s) comprising the matchbox will be inserted as wrap areas in the <code>Textflow</code> after they have been calculated. The subsequent lines after the lines containing the matchbox will be wrapped around the rectangle(s). Default: <code>false</code>
doubleadapt	If <code>true</code> the start and end point of the second line will be adapted to the first line. Otherwise the second line will be shorter or longer by the amount of <code>doubleoffset</code> . Default: <code>true</code>
doubleoffset	(Float) If different from 0 the lines around the border of the inner matchbox rectangle will be doubled. The second line has the specified offset from the original line. If the offset is positive the line will be drawn outside the matchbox rectangle, and inside if the offset is negative. Default: 0 (i.e. single line)
drawleft drawbottom drawright drawtop	(Boolean) If <code>true</code> , the corresponding border of the rectangle will be drawn provided that the <code>borderwidth</code> is set to a value greater than 0. Default: <code>true</code>
end	(Boolean; only for <code>Textflow</code>) End of the matchbox. If <code>true</code> , all other suboptions for the current matchbox definition are ignored. Matchboxes in <code>Textflows</code> cannot be nested. The width of a <code>Textflow</code> matchbox is defined by the option <code>boxwidth</code> (if specified) and the extent of the text enclosed in the options <code>matchbox</code> and <code>matchbox=end</code> . If the <code>end</code> option has not been specified, the matchbox will end after the last character in the <code>Textflow</code> .
exceedlimit	(Float or percentage; only for <code>Textflow</code>) Upper limit for the part of the matchbox which is allowed to exceed beyond the bottom or right edge of the fitbox, specified in user coordinates or as a percentage of the matchbox height. If the specified limit would be exceeded <code>PDF_fit_textflow()</code> will return <code>_boxfull</code> ; the remaining text and the matchbox can be continued in the next fitbox. Default: 0, i.e. the matchbox must completely fit into the box.

Table 6.4 Suboptions for the matchbox option of various methods

option	description
innerbox	(Boolean; only for table cells, and TIFF and JPEG images) Table cells: if true, the cell box is reduced by the margins defined for the cell; otherwise the full cell box is used. TIFF and JPEG images: if true and the image contains a clipping path the bounding box of the clipping path is used instead of the full image. Default: false
margin	(Float or percentage) Additional margin for the matchbox rectangle, specified in user coordinates (must be greater than or equal to 0) or as a percentage of the rectangle width or height (must be less than 100%). This option is ignored for an edge for which offset* has been supplied. Default: 0
name	(Name string) Name of the matchbox. If the name has already been assigned to a matchbox, an additional rectangle for this matchbox is created. This means that a matchbox may consist of more than one rectangle. The name can be used in PDF_info_matchbox(). Various methods support the option usematchbox to reference one or more rectangles of a matchbox, e.g. to add an annotation with PDF_create_annotation(). Matchbox names can be used until the end of the current page. The name »*« (asterisk character) must not be used as matchbox name. Default: no name
offsetleft offsetbottom offsetright offsettop	(Float or percentage) User-defined offset from the left/right/bottom/top edge of the calculated rectangle and the desired box. The values are specified in user coordinates or as a percentage of the rectangle's width (for offsetleft/offsetright) or height (for offsetbottom/offsettop). Negative values are allowed, and can be used to extend the matchbox. Default of offsetleft/offsetbottom: margin; Default of offsetright/offsettop: -margin
openrect	(Boolean; only for Textflow and table cells) Textflow: If true and a matchbox rectangle has to be split (e.g. because of a font change or line break), the right border of the first rectangle and the left border of the second rectangle will not be drawn. Table cells: If true and a table row is split to the next table instance the bottom border of the first part and the top border of the second part will not be drawn. Default: false
round	(Float) Adjacent lines of a matchbox rectangle will be joined with a circular arc with the specified radius and the line segments as tangents. If the specified radius is negative the arc segments will be swept inwards, and the tangents will be perpendicular to the line segments of the box. Default: 0 (no rounding)

6.2.3 Querying Matchboxes

C++ Java C# **double info_matchbox(String boxname, int num, String keyword)**

Perl PHP **float info_matchbox(string boxname, int num, string keyword)**

C **double PDF_info_matchbox(PDF *p, const char *boxname, int len, int num, const char *keyword)**

Query information about a matchbox on the current page.

boxname (Name string) Name of a matchbox which has been created under this name on the current page. It must have been created with the *name* suboption of the *matchbox* option when the matchbox was defined. Alternatively, the name '*' (asterisk character) can be used to query information about all matchboxes on the page. An empty *boxname* can be used to query information about all matchbox rectangles on the current page.

len (C language binding only) Length of *name* in bytes. If *len* = 0 a null-terminated string must be provided.

num Positive number of a matchbox or rectangle (the first has number 1).

keyword A keyword specifying the requested information according to Table 6.5.

Returns The value of some matchbox parameter as requested by *keyword*. If a matchbox with the specified name or a matchbox rectangle with the specified number does not exist, the return value is -1 (in PHP: 0) for the keywords *boundingbox*, *name*, and *rectangle*, and 0 for all other keywords. If the requested keyword produces text, a string index is returned, and the corresponding string must be retrieved with *PDF_get_string()*.

Depending on the current scope, the method returns information about the matchboxes on the current page, pattern, template, or glyph description.

Details Named matchboxes within a Textflow can only be queried after calling *PDF_fit_textflow()*. Matchboxes created in blind mode cannot be queried.

Rectangles for the keywords *boundingbox*, *exists*, *height*, *name*, *rectangle*, *width*, *x1*, *y1*, ..., *x4*, *y4* are selected as follows:

- ▶ If *boxname* contains the name of a matchbox: select the *num*-th rectangle of the specified named matchbox on the current page.
- ▶ If *boxname*=*: select the first rectangle of the *num*-th named matchbox on the current page.
- ▶ If *boxname* is empty: select the *num*-th rectangle created by a named matchbox on the current page.

Scope any except *document and object*

Table 6.5 Keywords for *PDF_info_matchbox()*

keyword	description
activeitemid	Item id of the structure element if the matchbox has been created in a Tagged Textflow, otherwise -1.
boundingbox	Handle of a path object containing the bounding box of the selected rectangle in the current user coordinate system or -1 (in PHP: 0) if the specified rectangle doesn't exist. The bounding box is different from the rectangle if the matchbox was rotated.
count	(The <i>num</i> parameter is ignored) If <i>boxname</i> contains the name of a matchbox: Number of rectangles for this matchbox If <i>boxname</i> =*: number of matchboxes with at least one rectangle If <i>boxname</i> is empty: total number of rectangles created by named matchboxes
exists	1 if the selected rectangle exists, 0 otherwise.
height¹	Height of the selected rectangle in user coordinates
name	String index for the name of the matchbox for which the selected rectangle was created. The corresponding string can be retrieved via <i>PDF_get_string()</i>
rectangle	Handle of the path object containing the selected rectangle in user coordinates or -1 (in PHP: 0) if the rectangle couldn't be found
width¹	Width of the selected rectangle in user coordinates
x1, y1, ..., x4, y4¹	Position of the <i>i</i> -th corner (<i>i</i> =1, 2, 3, 4) of the selected rectangle in user coordinates. In the coordinate system of the respective fit element (image, text, etc.), <i>x1</i> , <i>y1</i> correspond to the lower left, <i>x2</i> , <i>y2</i> to the lower right, <i>x3</i> , <i>y3</i> to the upper right and <i>x4</i> , <i>y4</i> to the upper left corner.

1. This keyword is ignored if *boxname*=*

7 Graphics Methods

API methods in this chapter:

- ▶ `PDF_set_graphics_option()`
- ▶ `PDF_setlinewidth()`
- ▶ `PDF_save()`
- ▶ `PDF_restore()`
- ▶ `PDF_create_gstate()`
- ▶ `PDF_set_gstate()`
- ▶ `PDF_translate()`
- ▶ `PDF_scale()`
- ▶ `PDF_rotate()`
- ▶ `PDF_align()`
- ▶ `PDF_skew()`
- ▶ `PDF_concat()`
- ▶ `PDF_setmatrix()`
- ▶ `PDF_moveto()`
- ▶ `PDF_lineto()`
- ▶ `PDF_curveto()`
- ▶ `PDF_circle()`
- ▶ `PDF_arc()`
- ▶ `PDF_arcn()`
- ▶ `PDF_circular_arc()`
- ▶ `PDF_ellipse()`
- ▶ `PDF_elliptical_arc()`
- ▶ `PDF_rect()`
- ▶ `PDF_closepath()`
- ▶ `PDF_stroke()`
- ▶ `PDF_closepath_stroke()`
- ▶ `PDF_fill()`
- ▶ `PDF_fill_stroke()`
- ▶ `PDF_closepath_fill_stroke()`
- ▶ `PDF_clip()`
- ▶ `PDF_endpath()`
- ▶ `PDF_add_path_point()`
- ▶ `PDF_draw_path()`
- ▶ `PDF_info_path()`
- ▶ `PDF_delete_path()`

7.1 Graphics Appearance Options

Graphics appearance options. The graphics appearance options in Table 7.1 can be used with the following methods (note that not all methods support all options; see function descriptions for details):

- ▶ `PDF_set_graphics_option()`
- ▶ `PDF_create_gstate()` (only *dasharray*, *dashphase*, *flatness*, *linecap*, *linejoin*, *linewidth*, *miterlimit*)

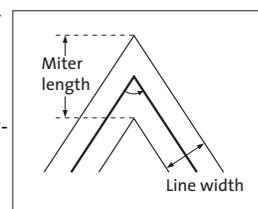
- ▶ `PDF_add_path_point()` and `PDF_draw_path()`
- ▶ The `fill` option of `PDF_fit_table()` (only `fillcolor`, `shading`) and the `stroke` option of `PDF_fit_table()` (only `dasharray`, `dashphase`, `linecap`, `linejoin`, `linewidth`, `strokecolor`)
- ▶ `PDF_shading_pattern()`
- ▶ The `matchbox` option of various methods

Table 7.1 Graphics appearance options

option	description	
<code>cliprule</code>	(Keyword) Clipping rule which determines the interior of areas for clipping; see <code>fillrule</code> for possible keywords. Default: value of the <code>fillrule</code> option	
<code>borderwidth</code>	(Float; only for matchboxes) Line width for the rectangle's border. If you set <code>borderwidth</code> to a value greater than 0 all rectangle borders will be stroked. To prevent the upper, lower, left, or right border from being stroked, set the corresponding <code>drawtop</code> , <code>drawbottom</code> , <code>drawleft</code> , or <code>drawright</code> option to false. Default: 0	
<code>dasharray</code>	(List of two non-negative floats or keyword) List of 2-12 alternating values for the lengths of dashes and gaps for stroked paths (measured in the user coordinate system). The array values must not be negative. They will be cyclically reused until the complete path is stroked. The keyword <code>none</code> can be used to create solid lines. Default: none	
<code>dashphase</code>	(Float) Distance into the dash pattern at which to start the dash. Default: 0	
<code>fillcolor</code> ¹	(Color) Fill color of the area. Default: generally {gray 0} (in PDF/A mode: {lab 0 0 0}), but none for matchboxes	
<code>fillrule</code>	(Keyword) Fill rule which determines the interior of areas for filling and clipping (default: winding):	
	<p>winding Use the nonzero winding number rule. For simple shapes, the result of filling matches intuitive expectations. For shapes consisting of multiple paths the direction of the paths is relevant.</p> <p>evenodd Use the even-odd rule, which yields the same results as winding for simple shapes, but produces different results for more complex shapes, especially self-intersecting paths.</p>	
<code>flatness</code> ²	(Float > 0) A positive number which describes the maximum distance (in device pixels) between a circular arc or a curve and an approximation constructed from straight line segments. Default: 1	
<code>gstate</code>	(Option list with options for <code>PDF_create_gstate()</code> or <code>gstate handle</code>) Graphics state options or handle. Default: no graphics state, i.e. current settings are used	
<code>initgraphics-state</code> ¹	(Boolean; only for <code>PDF_set_graphics_option()</code>) If true all graphics appearance options are initialized with the default values. The current clipping path is not affected. If false the current graphics state values are used. Default: false	
<code>linecap</code>	(Integer or keyword) Shape at the end of a path (default: projecting in <code>PDF_fit_table()</code> , otherwise butt):	
	<p>butt (Equivalent value: 0) Butt end caps: the stroke is squared off at the endpoint of the path.</p> <p>round (Equivalent value: 1) Round end caps: a semicircular arc with a diameter equal to the line width is drawn around the endpoint and filled in.</p> <p>projecting (Equivalent value: 2) Projecting square end caps: the stroke extends beyond the end of the line by a distance which is half the line width and is squared off.</p>	

Table 7.1 Graphics appearance options

option	description
linejoin	(Integer or keyword) Shape at the corners of paths (default: miter): <ul style="list-style-type: none"> miter (Equivalent value: 0) Miter joins: the outer edges of the strokes for the two segments are continued until they meet. If the extension projects too far, as determined by the miter limit, a bevel join is used instead. round (Equivalent value: 1) Round joins: a circular arc with a diameter equal to the line width is drawn around the point where the segments meet and filled in, producing a rounded corner. bevel (Equivalent value: 2) Bevel joins: the two path segments are drawn with butt end caps (see the discussion of <code>linecap</code>), and the resulting notch beyond the ends of the segments is filled in with a triangle.
linewidth	(Float > 0) Line width. Default: 1
miterlimit	(Float >= 1) Controls the spike produced by miter joins (default: 10; this corresponds to an angle of roughly 11.5 degrees). If the linejoin style is set to 0 (miter join), two line segments joining at a small angle will result in a sharp spike. This spike will be replaced by a straight end (i.e. the miter join will be changed to a bevel join) when the ratio of the miter length and the linewidth exceeds the miter limit.
shading	(Option list according to Table 8.4; only for matchboxes and tables) Specify a shading for the matchbox's rectangle(s) or table area The following options can be used: <code>antialias</code> , <code>domain</code> , <code>end</code> , <code>endcolor</code> , <code>N</code> , <code>start</code> , <code>startcolor</code> , <code>type</code>
strokecolor ¹	(Color) Stroke color of the path. Default: generally {gray 0} (in PDF/A mode: {lab 0 0 0}), but none for matchboxes



1. This option must not be used in glyph scope if the glyph's colored option is false.
2. This device-dependent graphics state option must not be used in pattern scope.

```
C++ Java C# void set_graphics_option(String optlist)
Perl PHP set_graphics_option(string optlist)
C void PDF_set_graphics_option(PDF *p, const char *optlist)
```

Set one or more graphics appearance options.

optlist An option list specifying graphics appearance options according to Table 7.1. The following options can be used:
`cliprule`, `dasharray`, `dashphase`, `fillcolor`, `fillrule`, `flatness`, `gstate`, `initgraphicsstate`, `linecap`, `linejoin`, `linewidth`, `miterlimit`, `strokecolor`

- Details** Graphics appearance options set the graphics state for the following methods:
- ▶ explicit drawing methods, e.g. `PDF_stroke()`, `PDF_fill()`
 - ▶ implicit drawing methods, e.g. the `showborder` option of `PDF_fit_textline()`, `PDF_fit_textflow()`
 - ▶ text output created with simple text output methods if no color has been set with text options, e.g. `PDF_show()`

All graphics appearance options are reset to their default values at the beginning of a page, pattern, template, or glyph description, and retain their values until the end of

the current *page*, *pattern*, *template*, or *glyph* scope. However, the graphics appearance options can also be reset with the *initgraphicsstate* option.

A subsequent call to *PDF_setcolor()* overrides the *fillcolor* and/or *strokecolor* values. A subsequent call to *PDF_setlinewidth()* overrides the *linewidth* value.

Scope *page*, *pattern*, *template*, *glyph* (the options *fillcolor*, *strokecolor* and *initgraphicsstate* can be used in *glyph* scope only if the *glyph*'s *colorized* option is *true*); the device-dependent option *flatness* must not be used in *pattern* scope.

7.2 Graphics State

C++ Java C# **void setlinewidth(double width)**

Perl PHP **setlinewidth(float width)**

C **void PDF_setlinewidth(PDF *p, double width)**

Set the current line width.

width The linewidth in units of the user coordinate system. It must be greater than 0.

Details This method sets the line width in the graphics state (see [PDF_set_graphics_option\(\)](#)) as well as the stroke width in the text state (see [PDF_set_text_option\(\)](#)). The *width* is reset to the default value of 1 at the beginning of each page.

Scope *page, pattern, template, glyph*

C++ Java C# **void save()**

Perl PHP **save()**

C **void PDF_save(PDF *p)**

Save the current graphics state to a stack.

Details The graphics state contains options that control all types of graphics objects. Saving the graphics state is not required by PDF; it is only necessary if the application wishes to return to some specific graphics state later (e.g. a custom coordinate system) without setting all relevant options explicitly again. The following items are subject to save/restore:

- ▶ graphics appearance options:
clipping path, coordinate system, current point, flatness tolerance, line cap style, dash pattern, line join style, line width, miter limit;
- ▶ color options: fill and stroke colors;
- ▶ graphics options which have been set with explicit graphics states in [PDF_set_gstate\(\)](#);
- ▶ text position and the following text appearance options:
charspacing, decorationabove, fakebold, font, fontsize, horizscaling, italicangle, leading, strokewidth, textrendering, textrise, underlineposition, underlinewidth, wordspacing.

Pairs of [PDF_save\(\)](#) and [PDF_restore\(\)](#) may be nested. Although the PDF specification doesn't limit the nesting level of save/restore pairs, applications should keep the nesting level below 26.

Most text options are affected by save/restore; see list above. The following text options are not subject to save/restore: *fillrule, kerning, underline, overline, strikeout*.

Scope *page, pattern, template, glyph*; must always be paired with a matching [PDF_restore\(\)](#) call. [PDF_save\(\)](#) and [PDF_restore\(\)](#) calls must be balanced on each page, pattern, template, and glyph description.

C++ Java C# **void restore()**

Perl PHP **restore()**

C **void PDF_restore(PDF *p)**

Restore the most recently saved graphics state from the stack.

Details The corresponding graphics state must have been saved on the same page, pattern, or template.

Scope *page, pattern, template, glyph*; must always be paired with a matching **PDF_save()** call. **PDF_save()** and **PDF_restore()** calls must be balanced on each page, pattern, template, and glyph description.

C++ Java C# **int create_gstate(String optlist)**

Perl PHP **int create_gstate(string optlist)**

C **int PDF_create_gstate(PDF *p, const char *optlist)**

Create a graphics state object subject to various options.

optlist An options list with graphics state options:

- ▶ Graphics appearance options according to Table 7.1:
dasharray, dashphase, linecap, linejoin, linewidth, miterlimit
- ▶ Graphics state options according to Table 7.2:
alphaishape, blendmode, halftoneorigin, opacityfill, opacitystroke¹, renderingintent, softmask¹, strokeadjust, textknockout, useblackptcomp
- ▶ Device-dependent graphics state options according to Table 7.3:
overprintfill, overprintmode, overprintstroke, smoothness
- ▶ Device-dependent graphics appearance option according to Table 7.1: *flatness*

Returns A graphics state handle that can be used in subsequent calls to **PDF_set_gstate()** during the enclosing *document* scope.

Details The option list may contain any number of graphics state options. Options which are not specified retain the previous value which may either be the default value or the value specified in another gstate which has been set earlier.

Scope any except *object*; the device-dependent graphics state options in Table 7.3 must not be used in pattern scope

Table 7.2 Options for **PDF_create_gstate()**

option	description
alphaishape	(Boolean) The current soft mask and alpha constant are treated as shape (true) or opacity (false). Default: false
blendmode¹	(Keyword; in PDF/A-1 and PDF/X-3 only Normal is allowed) Blend mode for transparency operations (default: None): Color, ColorDodge, ColorBurn, Darken, Difference, Exclusion, HardLight, Hue, Lighten, Luminosity, Multiply, None, Normal, Overlay, Saturation, Screen, SoftLight See PDFlib Tutorial for descriptions of these blend modes and examples for their use.
halftoneorigin	(List of two floats; PDF 2.0) Coordinates of the halftone origin in user coordinates

Table 7.2 Options for `PDF_create_gstate()`

option	description
<code>opacityfill</code> ¹	(Float or percentage; in PDF/A-1 and PDF/X-3 it must have the value 1) Opacity for fill operations in the range 0..1. The value 0 means fully transparent; 1 means fully opaque.
<code>opacitystroke</code> ¹	(Float or percentage; in PDF/A-1 and PDF/X-3 it must have the value 1) Opacity for stroke operations in the range 0..1. The value 0 means fully transparent; 1 means fully opaque.
<code>renderingintent</code>	(Keyword) Rendering intent for CIE-based color conversions: Auto, AbsoluteColorimetric, RelativeColorimetric, Saturation, Perceptual
<code>softmask</code> ¹	<p>(Option list or keyword; in PDF/A-1 and PDF/X-3 only none is allowed) Current soft mask for transparent imaging. A gstate soft mask can be overridden by an image soft mask since PDF doesn't support two soft masks at the same time (workaround: use the image option <code>template-options={transparencygroup={isolated}}</code>). Supported options and keyword (default: none):</p> <p>backdropcolor (Keyword or list with one, three, or four floats; only allowed for <code>type=luminosity</code>) Initial background of the soft mask (default: opaque):</p> <p>opaque The soft mask is initialized with black to make it opaque, i.e. all colors are blocked in areas where no light (white) content is drawn in the template.</p> <p>transparent The soft mask is initialized with white to make it transparent, i.e. the current color is used in areas where no dark (black) content is drawn in the template.</p> <p>As an alternative to the keywords color values can be supplied. The number of color components must match the <code>colorspace</code> suboption of the <code>transparencygroup</code> option used for creating the soft mask template (e.g. 3 for DeviceRGB).</p> <p>invert (Boolean) If true the effect of the soft mask is inverted, i.e. dark areas of the template are transparent while light areas block the color. This option also reverses the effect of the keywords for the <code>backdropcolor</code> suboption. Default: false</p> <p>none (Keyword) No soft mask at all; this is required to disable soft masks which may be in effect from a previous graphics state.</p> <p>template (Template handle; required) Transparency group template which has been created with <code>PDF_begin_template_ext()</code> and the <code>transparencygroup</code> option. If <code>type=luminosity</code> the template must have been created with the <code>colorspace</code> suboption and a value different from none.</p> <p>type (Keyword; required) Method for deriving mask values from the transparency group template:</p> <p>alpha The template's alpha values define the mask; the colors are ignored.</p> <p>luminosity The template colors are converted to luminosity values (perceived gray levels) which define the mask.</p>
<code>strokeadjust</code>	(Boolean) Whether or not to apply automatic stroke adjustment. Default: false
<code>textknockout</code>	(Boolean) With respect to compositing, glyphs in a text object are treated as separate objects (false) or as a single object (true). Default: true
<code>useblackptcomp</code>	(Keyword; PDF 2.0) Control black point compensation for CIE-based color conversions. Supported keywords: off, on and default. Default: default

1. A graphics state created with this transparency option must not be used in glyph scope if the glyph's colored option is false..

Table 7.3 Device-dependent options for `PDF_create_gstate()` (must not be used in *pattern* scope)

option	description
overprintfill	(Boolean) Overprint behavior for all non-stroking operations including image placement: if false, painting in any color space erases the corresponding areas of unspecified colorants; if true and the output device supports overprinting, previous markings in other colorants are left unchanged. Default: false
overprintmode	(Integer) Overprint behavior of CMYK component values <i>o</i> if <code>overprintfill</code> or <code>overprintstroke</code> is true. The overprint mode affects only text and vector elements, but not images or shadings. Allowed values (default: 0): <i>o</i> (Zero) Each color component replaces previously placed marks («foreground color wins»). 1 A color component of <i>o</i> leaves the corresponding component of previously painted color unchanged («foreground tint value <i>o</i> is ignored»). In other words, the value zero is treated as unspecified. PDF/A-2/3: <code>overprintmode=1</code> is not allowed if the current color space is ICC-based CMYK and <code>overprintfill</code> or <code>overprintstroke</code> is true.
overprintstroke	(Boolean) Overprint for stroke operations: if false, painting in any color space erases the corresponding areas of unspecified colorants; if true and the output device supports overprinting, previous markings in other colorants are left unchanged. Default: false
smoothness	(Float in the range 0..1) Maximum error of color interpolation for shadings

C++ Java C# **void set_gstate(int gstate)**

Perl PHP **set_gstate(int gstate)**

C **void PDF_set_gstate(PDF *p, int gstate)**

Activate a graphics state object.

gstate A handle for a graphics state object retrieved with `PDF_create_gstate()`.

Details All options contained in the graphics state object are set. Graphics state options accumulate when this method is called multiply. Options which are not specified in the `gstate` retain the previous value which may either be the default value or the value specified in another `gstate` which has been set earlier. All graphics state options are reset to their default values at the beginning of a page.

Scope *page, pattern, template, glyph*; the device-dependent graphics state options in Table 7.3 must not be used in *pattern* scope.

7.3 Coordinate System Transformations

All transformation methods (`PDF_translate()`, `PDF_scale()`, `PDF_rotate()`, `PDF_align()`, `PDF_skew()`, `PDF_concat()`, `PDF_setmatrix()`, and the `initgraphicsstate` option of `PDF_set_graphics_option()`) change the coordinate system used for drawing subsequent objects. They do not affect any existing objects on the page.

C++ Java C# **void translate(double tx, double ty)**

Perl PHP **translate(float tx, float ty)**

C **void PDF_translate(PDF *p, double tx, double ty)**

Translate the origin of the coordinate system.

tx, ty The new origin of the coordinate system is the point (tx, ty), measured in the old coordinate system.

Scope *page, pattern, template, glyph*

C++ Java C# **void scale(double sx, double sy)**

Perl PHP **scale(float sx, float sy)**

C **void PDF_scale(PDF *p, double sx, double sy)**

Scale the coordinate system.

sx, sy Scaling factors in x and y direction.

Details This method scales the coordinate system by *sx* and *sy*. It may also be used for achieving a reflection (mirroring) by using a negative scaling factor. One unit in the x direction in the new coordinate system equals *sx* units in the x direction in the old coordinate system; analogous for y coordinates.

Scope *page, pattern, template, glyph*

C++ Java C# **void rotate(double phi)**

Perl PHP **rotate(float phi)**

C **void PDF_rotate(PDF *p, double phi)**

Rotate the coordinate system.

phi The rotation angle in degrees.

Details Angles are measured counterclockwise from the positive x axis of the current coordinate system. The new coordinate axes result from rotating the old coordinate axes by *phi* degrees.

Scope *page, pattern, template, glyph*

C++ Java C# **void align(double dx, double dy)**

Perl PHP **align(float dx, float dy)**

C **void PDF_align(PDF *p, double dx, double dy)**

Align the coordinate system with a relative vector.

dx, dy Coordinates of a direction vector *dx* and *dy* must not both be 0.

Details Rotate the coordinate system such that the *x* axis of the new coordinate system is aligned with the vector (*dx*, *dy*), and the *y* axis is aligned with (-*dy*, *dx*). This is equivalent to `PDF_rotate()` with $\text{phi} = 180^\circ / \pi * \text{atan2}(\text{dy}/\text{dx})$.

Scope *page, pattern, template, glyph*

C++ Java C# **void skew(double alpha, double beta)**

Perl PHP **skew(float alpha, float beta)**

C **void PDF_skew(PDF *p, double alpha, double beta)**

Skew the coordinate system.

alpha, beta Skewing angles in *x* and *y* direction in degrees.

Details Skewing (or shearing) distorts the coordinate system by the given angles in *x* and *y* direction. *alpha* is measured counterclockwise from the positive *x* axis of the current coordinate system, *beta* is measured clockwise from the positive *y* axis. Both angles must not be odd multiples of 90°.

Scope *page, pattern, template, glyph*

C++ Java C# **void concat(double a, double b, double c, double d, double e, double f)**

Perl PHP **concat(float a, float b, float c, float d, float e, float f)**

C **void PDF_concat(PDF *p, double a, double b, double c, double d, double e, double f)**

Apply a transformation matrix to the current coordinate system.

a, b, c, d, e, f Elements of a transformation matrix. The six values make up a matrix in the same way as in PDF (see references). In order to avoid degenerate transformations, $a*d$ must not be equal to $b*c$.

Details This method allows for the most general form of transformations. Unless you are familiar with the use of transformation matrices, the use of `PDF_translate()`, `PDF_scale()`, `PDF_rotate()`, and `PDF_skew()` is suggested instead of this method. The coordinate system is reset to the default coordinate system (i.e. the current transformation matrix is the identity matrix $[1, 0, 0, 1, 0, 0]$) at the beginning of each page.

Scope *page, pattern, template, glyph*

C++ Java C# **void setmatrix(double a, double b, double c, double d, double e, double f)**

Perl PHP **setmatrix(float a, float b, float c, float d, float e, float f)**

C **void PDF_setmatrix(PDF *p, double a, double b, double c, double d, double e, double f)**

Explicitly set the current transformation matrix.

a, b, c, d, e, f See [PDF_concat\(\)](#).

Details This method is similar to [PDF_concat\(\)](#). However, it disposes of the current transformation matrix, and replaces it with the new matrix.

Scope *page, pattern, template, glyph*

7.4 Path Construction

Note Make sure to call one of the methods in Section 7.5, »Painting and Clipping«, page 150, after using the methods in this section, or the constructed path will have no effect, and subsequent operations may raise an exception.

PDF/UA Vector graphics must be tagged as *Artifact* or *Figure* with a call to `PDF_begin_item()`.

C++ Java C# **void moveto(double x, double y)**

Perl PHP **moveto(float x, float y)**

C **void PDF_moveto(PDF *p, double x, double y)**

Set the current point for graphics output.

x, y The coordinates of the new current point.

Details The current point is set to the default value of *undefined* at the beginning of each page. The current points for graphics and the current text position are maintained separately.

Scope *page, pattern, template, glyph, path*; this method starts *path* scope.

C++ Java C# **void lineto(double x, double y)**

Perl PHP **lineto(float x, float y)**

C **void PDF_lineto(PDF *p, double x, double y)**

Draw a line from the current point to another point.

x, y The coordinates of the second endpoint of the line.

Details This method adds a straight line from the current point to (x, y) to the current path. The current point must be set before using this method. The point (x, y) becomes the new current point.

The line will be centered around the »ideal« line, i.e. half of the linewidth (as determined by the value of the *linewidth* option) will be painted on each side of the line connecting both endpoints. The behavior at the endpoints is determined by the *linecap* option.

Scope *path*

C++ Java C# **void curveto(double x1, double y1, double x2, double y2, double x3, double y3)**

Perl PHP **curveto(float x1, float y1, float x2, float y2, float x3, float y3)**

C **void PDF_curveto(PDF *p, double x1, double y1, double x2, double y2, double x3, double y3)**

Draw a Bézier curve from the current point, using three more control points.

x1, y1, x2, y2, x3, y3 The coordinates of three control points.

Details A Bézier curve is added to the current path from the current point to $(x3, y3)$, using $(x1, y1)$ and $(x2, y2)$ as control points. The current point must be set before using this method. The endpoint of the curve becomes the new current point.

Scope *path*

C++ Java C# **void circle(double x, double y, double r)**

Perl PHP **circle(float x, float y, float r)**

C **void PDF_circle(PDF *p, double x, double y, double r)**

Draw a circle.

x, y The coordinates of the center of the circle.

r The radius of the circle.

Details This method adds a circle to the current path as a complete subpath. The point $(x + r, y)$ becomes the new current point. The resulting shape will be circular in user coordinates. If the coordinate system has been scaled differently in x and y directions, the resulting curve will be elliptical. The circle is created in counterclockwise direction.

Scope *page, pattern, template, glyph, path*; this method starts *path* scope.

C++ Java C# **void arc(double x, double y, double r, double alpha, double beta)**

Perl PHP **arc(float x, float y, float r, float alpha, float beta)**

C **void PDF_arc(PDF *p, double x, double y, double r, double alpha, double beta)**

Draw a counterclockwise circular arc segment.

x, y The coordinates of the center of the circular arc segment.

r The radius of the circular arc segment. r must be nonnegative.

alpha, beta The start and end angles of the circular arc segment in degrees.

Details This method adds a counterclockwise circular arc segment to the current path, extending from *alpha* to *beta* degrees. For both *PDF_arc()* and *PDF_arcn()*, angles are measured counterclockwise from the positive x axis of the current coordinate system. If there is a current point an additional straight line is drawn from the current point to the starting point of the arc. The endpoint of the arc becomes the new current point.

The arc segment will be circular in user coordinates. If the coordinate system has been scaled differently in x and y directions the resulting curve will be elliptical.

Scope *page, pattern, template, glyph, path*; this method starts *path* scope.

C++ Java C# **void arcn(double x, double y, double r, double alpha, double beta)**

Perl PHP **arcn(float x, float y, float r, float alpha, float beta)**

C **void PDF_arcn(PDF *p, double x, double y, double r, double alpha, double beta)**

Draw a clockwise circular arc segment.

Details Except for the drawing direction, this method behave exactly like *PDF_arc()*. In particular, the angles are still measured *counterclockwise* from the positive x axis.

C++ Java C# **void circular_arc(double x1, double y1, double x2, double y2)**

Perl PHP **circular_arc(float x1, float y1, float x2, float y2)**

C **void PDF_circular_arc(PDF *p, double x1, double y1, double x2, double y2)**

Draw a circular arc segment defined by three points.

x1, y1 The coordinates of an arbitrary point on the circular arc segment.

x2, y2 The coordinates of the end point of the circular arc segment.

Details This method adds a circular arc segment to the current path. The arc segment will start at the current point, pass through (x_1, y_1) , and end at (x_2, y_2) . The current point must be set before using this method. The endpoint of the curve becomes the new current point.

The arc segment will be circular in user coordinates. If the coordinate system has been scaled differently in x and y directions the resulting curve will be elliptical.

Scope *path*

C++ Java C# **void ellipse(double x, double y, double rx, double ry)**

Perl PHP **ellipse(float x, float y, double rx, double ry)**

C **void PDF_ellipse(PDF *p, double x, double y, double rx, double ry)**

Draw an ellipse.

x, y The coordinates of the center of the ellipse.

rx, ry The x and y radii of the ellipse.

Details This method adds an ellipse to the current path as a complete subpath. The point $(x + rx, y)$ becomes the new current point. The ellipse is created in counterclockwise direction.

Scope *page, pattern, template, glyph, path*; this method starts *path* scope.

C++ Java C# **void elliptical_arc(double x, double y, double rx, double ry, String optlist)**

Perl PHP **elliptical_arc(float x, float y, double rx, double ry, string optlist)**

C **void PDF_elliptical_arc(PDF *p, double x, double y, double rx, double ry, const char *optlist)**

Draw an elliptical arc segment from the current point.

x, y The coordinates of the end point of the elliptical arc segment.

rx, ry The x and y radii of the ellipse. At least one of these values must be larger than half the distance between the current point and (x, y) .

optlist An option list specifying construction options for the elliptical arc according to Table 7.4.

Details This method adds an elliptical arc segment to the current path. The arc segment will start at the current point and end at (x, y) . The current point must be set before using this method. The end point of the arc becomes the new current point. Two of the four possible arc segments represent an arc segment of $\leq 180^\circ$ (the small arc segments), while the other two represent an arc segment of $\geq 180^\circ$ (the large arc segments).

Scope *page, pattern, template, glyph, path*; this method starts *path* scope.

Table 7.4 Options for `PDF_elliptical_arc()`

option	description
clockwise	(Boolean) If <code>true</code> one of the clockwise arc segments will be created; otherwise one of the counterclockwise arc segments will be created. Default: <code>false</code>
largearc	(Boolean) If <code>true</code> one of the large arc segments will be created; otherwise one of the small arc segments will be created. Default: <code>false</code>
rectify	(Boolean) If <code>true</code> radii which are too small will be modified so that the elliptical arc can be constructed; otherwise an exception will be thrown. Default: <code>false</code>
xrotate	(Float) Rotation angle for the ellipse, i.e. the angle of the ellipse x axis relative to the current coordinate system x axis in degrees. The start and end point of the arc segment remain fixed. Default: <code>0</code>

C++ Java C# **`void rect(double x, double y, double width, double height)`**

Perl PHP **`rect(float x, float y, float width, float height)`**

C **`void PDF_rect(PDF *p, double x, double y, double width, double height)`**

Draw a rectangle.

`x, y` The coordinates of the lower left corner of the rectangle.

`width, height` The size of the rectangle.

Details This method adds a rectangle to the current path as a complete subpath. Setting the current point is not required before using this method. The point (x, y) becomes the new current point. The lines are centered around the »ideal« line, i.e. half of the linewidth (as determined by the value of the *linewidth* option) is painted on each side of the line connecting the respective endpoints. The rectangle is created in counterclockwise orientation. This method implicitly closes the path.

Scope *page, pattern, template, glyph, path*; this method starts *path* scope.

C++ Java C# **`void closepath()`**

Perl PHP **`closepath()`**

C **`void PDF_closepath(PDF *p)`**

Close the current path.

Details This method closes the current subpath, i.e. adds a line from the current point to the starting point of the subpath.

Scope *path*

7.5 Painting and Clipping

Note Most methods in this section clear the path, and leave the current point undefined. Subsequent drawing operations must therefore explicitly set the current point (e.g. using `PDF_moveto()`) after one of these methods has been called.

C++ Java C# **void stroke()**

Perl PHP **stroke()**

C **void PDF_stroke(PDF *p)**

Stroke the path with the current line width and current stroke color, and clear it.

Scope *path*; this method terminates *path* scope.

C++ Java C# **void closepath_stroke()**

Perl PHP **closepath_stroke()**

C **void PDF_closepath_stroke(PDF *p)**

Close the path, and stroke it.

Details This method closes the current subpath (adds a straight line segment from the current point to the starting point of the path), and strokes the complete current path with the current line width and the current stroke color.

Scope *path*; this method terminates *path* scope.

C++ Java C# **void fill()**

Perl PHP **fill()**

C **void PDF_fill(PDF *p)**

Fill the interior of the path with the current fill color.

Details This method fills the interior of the current path with the current fill color. The interior of the path is determined by one of two algorithms (see the *fillrule* option). Open paths are implicitly closed before being filled.

Scope *path*; this method terminates *path* scope.

C++ Java C# **void fill_stroke()**

Perl PHP **fill_stroke()**

C **void PDF_fill_stroke(PDF *p)**

Fill and stroke the path with the current fill and stroke color.

Scope *path*; this method terminates *path* scope.

C++ Java C# **void closepath_fill_stroke()**

Perl PHP **closepath_fill_stroke()**

C **void PDF_closepath_fill_stroke(PDF *p)**

Close the path, fill, and stroke it.

Details This method closes the current subpath (adds a straight line segment from the current point to the starting point of the path), and fills and strokes the complete current path.

Scope *path*; this method terminates *path* scope.

C++ Java C# **void clip()**

Perl PHP **clip()**

C **void PDF_clip(PDF *p)**

Use the current path as clipping path, and terminate the path.

Details This method uses the intersection of the current path and the current clipping path as the clipping path for subsequent operations. The clipping path is set to the default value of the page size at the beginning of each page. The clipping path is subject to [PDF_save\(\)/PDF_restore\(\)](#). It can only be enlarged by means of [PDF_save\(\)/PDF_restore\(\)](#). The clipping area is determined according to the algorithm selected with the *cliprule* option.

Scope *path*; this method terminates *path* scope.

C++ Java C# **void endpath()**

Perl PHP **endpath()**

C **void PDF_endpath(PDF *p)**

End the current path without filling or stroking it.

Details This method doesn't have any visible effect on the page. It generates an invisible path on the page.

Scope *path*; this method terminates *path* scope.

7.6 Path Objects

C++ Java C# `int add_path_point(int path, double x, double y, String type, String optlist)`

Perl PHP `int add_path_point(int path, float x, float y, string type, string optlist)`

C `int PDF_add_path_point(PDF *p, int path, double x, double y, const char *type, const char *optlist)`

Add a point or path to a new or existing path object.

path A valid path handle returned by another call to `PDF_add_path_point()` or -1 (in PHP: 0) to create a new path.

x, y Coordinates of the new current point. If `polar=false` the two numbers designate the Cartesian coordinates (x, y) of the point. If `polar=true` the two numbers designate the radius r and angle ϕ (in degrees or radians depending on the option `radians`) of the point. This point will become the new current point for `type=circle, circular, elliptical, ellipse, move, line, curve, rect`.

type The type of the point according to Table 7.5.

Table 7.5 Types of points for `PDF_add_path_point()`

type	description
addpath	Add the path definition specified in the <code>svgpath</code> option as a complete subpath, using (x, y) as origin.
circle	Add a circle to the path as a complete subpath, using (x, y) as the center and radius for the size. ¹
circular	Add a circular arc from the current point to (x, y) with the previously defined control point as third circular arc point which is required. If the new point is identical with the current point a circle with diameter between the current point and the control point will be created. ²
control	Control point for a Bézier curve or a circular arc. ²
curve	Add a Bézier curve from the current point to the new point with the previously defined control points. At least one control point must be provided. If only one control point is available, it will be used as the second control point for the curve, and the first control point will be constructed as the reflection of the second control point at the endpoint of the previous Bézier curve. ²
ellipse	Add an ellipse to the path as a complete subpath, using (x, y) as the center and the values in the <code>radius</code> option for the size. ¹ The ellipse can be rotated with the <code>xrotate</code> option.
elliptical	Add an elliptical arc from the current point to (x, y). The size and orientation of the ellipse are defined by the <code>radius, xrotate, largearc, and clockwise</code> options. If only a single value is provided as <code>radius</code> a circular arc will be created. In this case an appropriate circular arc point will be created automatically. If two values are provided in the <code>radius</code> option a set of Bézier curves will be created. ²
line	Add a line segment from the current point to (x, y). ²
move	Start a new subpath. Subpaths will be numbered consecutively (1, 2, ...). The first subpath starts at the origin.
pathref	Add a reference to the path specified in the <code>path</code> option as a complete subpath, using (x, y) as origin. Since the path is referenced (and not copied) subsequent changes to path will be reflected when drawing the path.
rect	Add a rectangle to the path as a complete subpath, using (x, y) as the center of the rectangle and width and height for the size. ¹ The corners of the rectangle can be rounded with the <code>round</code> option. Alternatively, the corners can be rounded with elliptical arcs with the <code>radius</code> option.

1. A new point with `type=move` and the same coordinates and graphics appearance options is created automatically after the path.

2. Graphics appearance options and path operation options are not allowed for these types.

optlist Option list specifying path construction options:

- ▶ Path calculation and naming options for a point according to Table 7.6:
name, polar, radians, relative
- ▶ Path operation options according to Table 7.6:
close, fill, round, stroke
- ▶ Options for adding path definitions according to Table 7.6:
path, svgpath
- ▶ Options for constructing path elements according to Table 7.6:
clockwise, height, largearc, radius, rectify, width, xrotate
- ▶ Graphics appearance options according to Table 7.1 (only for *type=addpath, circle, ellipse, move, rect, or pathref*):
dasharray, dashphase, fillcolor, fillrule, flatness, gstate, linecap, linejoin, linewidth, miterlimit, strokecolor

Returns A path handle which can be used until it is deleted with `PDF_delete_path()`.

Details A path object serves as a container for vector graphics. The path object can be populated with paths and subpaths incrementally, where new path elements can be created by specifying individual path nodes or by adding path definitions specified via a path handle or an SVG path description. The generated path can later be used with `PDF_draw_path()` and other methods.

A path object can hold any number of paths. Each path in turn may contain one or more subpaths which can be selected for drawing in the *subpaths* option of `PDF_draw_path()`. All paths will be closed, filled, stroked, and rounded separately according to the specified options.

An operation with any of the types *addpath, circle, ellipse, move, rect, or pathref* starts a new subpath. Graphics appearance options and path operation options (e.g. *stroke, fill*) can only be changed for *type=addpath, circle, ellipse, move, rect, or pathref*. In this situation a new path within the path object will be started automatically. Shapes of type *circle, ellipse, elliptical, and rect* are created in counterclockwise direction by default, but this can be changed with the option *clockwise*.

Scope any

Table 7.6 Options for `PDF_add_path_point()`

option	description
clockwise	(Boolean; only for <i>type=circle, ellipse, elliptical, rect</i>) If true the shape is created in clockwise direction, otherwise counterclockwise. Default: false
close	(Boolean; only for <i>type=move</i>) If true, the subpath will be closed with a straight line. Default: see footnote ¹
fill	(Boolean; only for <i>type=move</i>) If true the subpath will be closed and filled. Default: see footnote ¹
height	(Float; only for <i>type=rect</i> ; required in this case) Height of the rectangle
largearc	(Boolean; only for <i>type=elliptical</i>) If true one of the large elliptical arc segments will be created; otherwise one of the small elliptical arc segments will be created. Default: false
name	(String) Name of the point. Default: <code>p<i></code> (e.g. <code>p1</code>) where <i>i</i> is the consecutive number of supplied points.
path	(Path handle; only for <i>type=pathref</i>) The specified path will be added to the current path by reference. The coordinates of the added path refer to the current point as origin. Graphics appearance options and the name option are ignored.

Table 7.6 Options for `PDF_add_path_point()`

option	description
polar	(Boolean) If true, the (x, y) parameters are polar coordinates specifying radius r and angle phi, otherwise Cartesian coordinates specifying x and y values. Default: false
radians	(Boolean) If true, angles for polar coordinates are specified in radians, otherwise in degrees. Default: false
radius	(One or two floats; required for type=circle, ellipse and elliptical; also allowed for type=rect) The first value specifies the radius of the circle or the x radius of the ellipse. The second float value, if present, specifies the y radius of the ellipse. The first value will be used as default for the second value. For type=rect the values specify the x and y radii of the elliptical arcs in the rectangle corners. The elliptical arcs will be created immediately. Default: 0
rectify	(Boolean; only for type=ellipse and elliptical) If true radii which are too small are modified so that the elliptical arc can be constructed; otherwise an exception is thrown. Default: false
relative	(Boolean) If true, (x, y) are relative to the current point, otherwise to the current origin. Default: see footnote ¹
round	(Float; only for type=move and rect) Adjacent line vertices in the subpath will be rounded in their joining point by a circular arc with the line segments as its tangents and with the specified radius. If the radius is negative the arc will be grooved so that the corners are circularly grooved. If close=true and no line from the last to the first point was explicitly specified, the first line and the closing line will also be rounded. If round=0 no rounding will be done. The circular arcs will be created when the path is drawn. Default: see footnote ¹
stroke	(Boolean; only for type=move) If true the subpath will be stroked. Default: see footnote ¹
svgpath	(String; only for type=addpath) String containing a path description in SVG syntax (as provided to the d attribute of the path SVG element, see www.w3.org/TR/SVG11/paths.html#PathData). The specified path is added to the current path. The coordinates of the SVG path refer to the current point as origin. Graphics appearance options can be specified for the SVG path. If the path stems from an SVG file with a topdown coordinate system it must be translated and mirrored (even if PDFlib operates in topdown mode); see PDFlib Tutorial for details.
width	(Float; only for type=rect; required in this case) Width of the rectangle
xrotate	(Float; only for type=ellipse and elliptical) Rotation angle for the ellipse in current units (see option radians), i.e. the angle of the ellipse x axis relative to the current coordinate system x axis in degrees. The start and end point of the arc segment remain fixed. This option is ignored if only a single value was supplied as radius. Default: 0

1. The default is specified in `PDF_draw_path()`, `PDF_info_path()`, the `textpath` option of `PDF_fit_textline()`, the `wrap` option of `PDF_fit_textflow()`, or the `fitpath` option of `PDF_add_table_cell()`.

C++ Java C# **void draw_path(int path, double x, double y, String optlist)**

Perl PHP **draw_path(int path, float x, float y, string optlist)**

C **void PDF_draw_path(PDF *p, int path, double x, double y, const char *optlist)**

Draw a path object.

path A valid path handle returned by a call to `PDF_add_path_point()` or another method which returns a path handle (e.g. `PDF_info_image()` with the `boundingbox` keyword).

x, y Coordinates of the reference point in user coordinates. The reference point is used by various options, and specifies the position of the origin of the path object in the current user system. This implies a translation of the path object.

If the `usematchbox` option has been specified, x and y are ignored.

If the *boxsize* option is specified, (x, y) is the lower left corner of the fitbox (see Table 6.1) into which the path object will be fit.

optlist Option list specifying path drawing options:

- ▶ Fitting options according to Table 6.1:
align, attachmentpoint, boxsize, fitmethod, orientate, position, scale
- ▶ Path operation and subpath selection options according to Table 7.7:
clip, close, fill, round, stroke, subpaths
- ▶ Box options according to Table 7.7:
bboxexpand, boundingbox, usematchbox
- ▶ Graphics appearance options for the *fill* and *stroke* options according to Table 7.1:
dasharray, dashphase, fillcolor, flatness, gstate, linecap, linejoin, linewidth, miterlimit, strokecolor
- ▶ Graphics appearance options according to Table 7.1 for the *clip* option according to Table 7.1: *cliprule, fillrule*
- ▶ Option for abbreviated structure element tagging according to Table 14.2 (only allowed in *page* scope): *tag*

Details The path(s) will be placed at the reference point (x, y) and then be stroked, filled, or used as a clipping path according to the specified options. This method does not modify the current graphics state unless the *clip* option is used. The appearance and operation options override the default settings, but they do not override any appearance option which may have been specified for a subpath in `PDF_add_path_point()`.

PDF/UA All path objects must be tagged as *Artifact* or *Figure*, either with the *tag* option or with a preceding call to `PDF_begin_item()`.

Scope *page, pattern, template, glyph*

Table 7.7 Path operation options for `PDF_draw_path()` for controlling all subpaths in a path object

option	description
bboxexpand	(List of floats; ignored if the <i>boundingbox</i> option is specified) One or two floats which indicate the expansion of the automatically calculated bounding box (the smallest enclosing rectangle of the path object). Default: {0 0}
bounding-box	(Rectangle) A rectangle in the coordinate system of the path object which will be used as bounding box for fitting the path object into the fitbox. Default: the smallest enclosing rectangle of the path object, possibly expanded according to the <i>bboxexpand</i> option
clip	(Boolean) If true the path will be closed and used as clipping path. Default: false
close	(Boolean) If true, each subpath will be closed with a straight line. Default: the value specified when the path was constructed, or false if no value was specified
fill	(Boolean; overrides clip) If true each path will be filled. Default: the value specified when the path was constructed, or false if no value was specified
round	(Float) For each subpath, adjacent line vertices will be rounded in their joining point by a circular arc with the line segments as its tangents and with the specified radius. If the radius is negative the arc will be grooved so that the corners are circular grooved. If <i>close=true</i> and no line from the last to the first point was explicitly specified, the first line and the closing line will also be rounded. If <i>round=0</i> no rounding will be done. Default: the value specified when the path was constructed, or 0 if no value was specified
stroke	(Boolean; overrides clip) If true the path will be stroked. Default: false

Table 7.7 Path operation options for `PDF_draw_path()` for controlling all subpaths in a path object

option	description
subpaths	(List of integers or single keyword) List with the numbers of subpaths to be drawn; the first subpath has number 1. The keyword <code>all</code> specifies all subpaths. Default: <code>all</code>
usematchbox	(List of strings; only allowed in page scope) Draw the path in all rectangles of the named matchbox. The first element in the option list is a name string which specifies a matchbox. The second element is either an integer specifying the number of the desired rectangle (starting with 1) or the keyword <code>all</code> to specify all rectangles of the matchbox. If the second element is missing, it defaults to <code>all</code> . The supplied path is drawn in the specified rectangle(s) of the matchbox. The lower left corner of the rectangle is used as reference point instead of the <code>x/y</code> parameters, and the rectangle width and height are used as <code>boxsize</code> . The parameters <code>x/y</code> and the option <code>boxsize</code> are ignored if <code>usematchbox</code> is supplied. If the matchbox or the specified rectangle does not exist on the current page the method silently returns without drawing anything.

C++ Java C# **double** `info_path(int path, String keyword, String optlist)`

Perl PHP **float** `info_path(int path, string keyword, string optlist)`

C **double** `PDF_info_path(PDF *p, int path, const char *keyword, const char *optlist)`

Query the results of drawing a path object without actually drawing it.

path A valid path handle returned by a call to `PDF_add_path_point()` or another method which returns a path handle (e.g. `PDF_info_image()` with the `boundingbox` keyword).

keyword A keyword specifying the requested information:

- ▶ Keywords for querying the results of object fitting according to Table 6.3:
`boundingbox`, `fitscalex`, `fitscaley`, `height`, `objectheight`, `objectwidth`, `width`, `x1`, `y1`, `x2`, `y2`, `x3`, `y3`, `x4`, `y4`
- ▶ Additional keywords according to Table 7.8:
`bboxwidth`, `bboxheight`, `numpoints`, `pathlength`, `px`, `py`

Table 7.8 Keywords for `PDF_info_path()`

keyword	description										
bboxwidth , bboxheight	Width and height of the bounding box for the path										
numpoints	Number of supplied points. The option <code>subpaths</code> is ignored.										
px , py	The x or y coordinate (in user coordinates) of the path point specified in the name option. The option <code>subpaths</code> is ignored.										
pathlength	Length of the path contour.										
type	Numerical indicator of the type of a point which has been specified with the option name. Path components of type <code>elliptical</code> , <code>circle</code> , and <code>ellipse</code> are already converted to Bézier curves, <code>type rect</code> is converted to lines: <table border="0" style="margin-left: 20px;"> <tr> <td>0</td> <td><code>move</code></td> </tr> <tr> <td>1</td> <td><code>line</code></td> </tr> <tr> <td>2</td> <td><code>control</code></td> </tr> <tr> <td>3</td> <td><code>curve</code></td> </tr> <tr> <td>4</td> <td><code>circular</code></td> </tr> </table>	0	<code>move</code>	1	<code>line</code>	2	<code>control</code>	3	<code>curve</code>	4	<code>circular</code>
0	<code>move</code>										
1	<code>line</code>										
2	<code>control</code>										
3	<code>curve</code>										
4	<code>circular</code>										

optlist Option list specifying path drawing options:

- ▶ All options of `PDF_draw_path()` according to Table 7.7
- ▶ Additional fitting option according to Table 6.1: `refpoint`
- ▶ Additional option according to Table 7.9: `name`

Returns The value of some path property as requested by keyword.

Details This method performs the same calculations as `PDF_draw_path()`, but does not create any visible output on the page.

Scope any

Table 7.9 Options for `PDF_info_path()`

option	description
<code>name</code>	Name of a path point for the keywords <code>px</code> , <code>py</code> or <code>type</code> . A default name (e.g. <code>p1</code>) can be used even if an explicit name has been specified in <code>PDF_add_path_point()</code> .

C++ Java C# **void delete_path(int path)**

Perl PHP **delete_path(int path)**

C **void PDF_delete_path(PDF *p, int path)**

Delete a path object.

path A valid path handle returned by a call to `PDF_add_path_point()` or another method which returns a path handle (e.g. `PDF_info_image()` with the `boundingbox` keyword).

Details Delete the path object and all associated internal data structures. Path objects are not automatically deleted in `PDF_end_document()`.

Scope any



8 Color Methods

API methods in this chapter:

- ▶ `PDF_set_color()`
- ▶ `PDF_load_iccprofile()`
- ▶ `PDF_makespotcolor()`
- ▶ `PDF_create_devicen()`
- ▶ `PDF_shading()`
- ▶ `PDF_shading_pattern()`
- ▶ `PDF_shfill()`
- ▶ `PDF_begin_pattern_ext()`
- ▶ `PDF_end_pattern()`

8.1 Setting Color

Fill and stroke colors can be set with `PDF_setcolor()` as well as with the options `fillcolor`, `strokecolor` and others. Using options is recommended because this method is more general. Section 1.1.4, »Color Data Type«, page 13, contains a discussion of color options.

```
C++ Java C# void setcolor(String fstype, String colorspace, double c1, double c2, double c3, double c4)
Perl PHP setcolor(string fstype, string colorspace, float c1, float c2, float c3, float c4)
C void PDF_setcolor(PDF *p,
    const char *fstype, const char *colorspace, double c1, double c2, double c3, double c4)
```

Set the current color space and color for the graphics and text state.

fstype One of *fill*, *stroke*, or *fillstroke* to specify that the color is set for filling, stroking, or both.

colorspace The color space to be used for the supplied color values or an RGB color value which is specified by name or hexadecimal values:

- ▶ First form: one of *gray*, *rgb*, *cmymk*, *spot*, *devicen*, *pattern*, *iccbasedgray*, *iccbasedrgb*, *iccbasedcmymk*, or *lab* to specify the color space.
- ▶ Second form: an RGB color name (e.g. *pink*) or a hash character followed by six hexadecimal digits (e.g. *#FFCoCB*). The parameters *c1*, *c2*, *c3*, and *c4* are ignored in this case.

c1, c2, c3, c4 Color components for the chosen color space. The interpretation of these values depends on the *colorspace* parameter (see PDFlib Tutorial for a full discussion of color spaces and values):

- ▶ *gray*: *c1* specifies a gray value;
- ▶ *rgb*: *c1*, *c2*, *c3* specify red, green, and blue values.
- ▶ *cmymk*: *c1*, *c2*, *c3*, *c4* specify cyan, magenta, yellow, and black values;
- ▶ *iccbasedgray*: *c1* specifies a gray value;
- ▶ *iccbasedrgb*: *c1*, *c2*, *c3* specify red, green, and blue values;
- ▶ *iccbasedcmymk*: *c1*, *c2*, *c3*, *c4* specify cyan, magenta, yellow, and black values;
- ▶ *spot*: *c1* specifies a spot color handle returned by `PDF_makespotcolor()`, and *c2* specifies a tint value between 0 and 1;

- ▶ *devicen*: *c1* specifies a DeviceN color space handle returned by `PDF_create_devicen()`; *c2*, *c3*, and *c4* specify up to three tint values between 0 and 1. DeviceN colors with $N > 3$ cannot be specified with this method.
- ▶ *lab*: *c1*, *c2*, and *c3* specify color values in the CIE $L^*a^*b^*$ color space. *c1* specifies the L^* (luminance) value in the range 0 to 100, and *c2*, *c3* specify the a^* , b^* (chrominance) values in the range -128 to 127.
- ▶ *pattern*: *c1* specifies a tiling pattern handle returned by `PDF_begin_pattern_ext()` or a shading pattern handle returned by `PDF_shading_pattern()`. If a tiling pattern has been created with the option `painttype=uncolored` in `PDF_begin_pattern_ext()`, the current fill or stroke color is applied when the pattern is used for filling or stroking. In this situation the current color space must not be another pattern color space.

Details All color values for the *gray*, *rgb*, and *cmymk* color spaces and the *tint* value for the *spot* color space must be numbers in the inclusive range 0–1. Unused parameters should be set to 0. More information about color spaces and color values can be found in the PDFlib Tutorial.

The fill and stroke color values for the *gray*, *rgb*, and *cmymk* color spaces are set to a default value of black at the beginning of each page. There are no defaults for spot and pattern colors.

If the *iccbasedgray/rgb/cmymk* color spaces are used, a suitable ICC profile must have been set before using one of the *iccprofilegray/rgb/cmymk* options.

This method is equivalent to `PDF_set_text_option()` and `PDF_set_graphics_option()` with the *fillcolor* and/or *strokecolor* options. `PDF_setcolor()` overrides the values of these options.

Colors in an *iccbased* color space and DeviceN colors with $N > 3$ cannot be specified with this method, but only with color options.

PDF/A *colorspace=gray* requires an output intent (any type) or the *defaultgray* option.
colorspace=rgb requires an RGB output intent or the *defaultrgb* option.
colorspace=cmymk requires a CMYK output intent or the *defaultcmymk* option.

PDF/UA Information should not be conveyed by color or contrast alone.

PDF/X-3: Using *iccbasedgray/rgb/cmymk* and *lab* color requires an ICC profile in the output intent (a standard name is not sufficient in this case).

PDF/X-3/4/5p: *colorspace=gray* requires a grayscale or CMYK device output intent or the *defaultgray* option.

colorspace=rgb requires an RGB output intent or the *defaultrgb* option

colorspace=cmymk requires a CMYK output intent or the *defaultcmymk* option.

PDF/X-5n: *colorspace=gray* can only be used if the output intent contains the colorant *Black* or the *defaultgray* option has been set.

colorspace=rgb can only be used if the *defaultrgb* option has been set.

colorspace=cmymk can only be used if the output intent contains all of the colorants *Cyan*, *Magenta*, *Yellow* and *Black* or the *defaultcmymk* option has been set.

Scope *page*, *pattern* (only if `painttype=colored`), *template*, *glyph* (only if the glyph's *colored* option is *true*), *document*; a pattern color can not be used within its own definition. Setting the color in *document* scope may be useful for defining spot colors with `PDF_makespotcolor()`.

8.2 ICC Profiles

C++ Java C# *int load_iccprofile(String profilename, String optlist)*

Perl PHP *int load_iccprofile(string profilename, string optlist)*

C *int PDF_load_iccprofile(PDF *p, const char *profilename, int len, const char *optlist)*

Search for an ICC profile and prepare it for later use.

profilename (Name string) The name of an *ICCProfile* resource, or a disk-based or virtual file name.

len (C language binding only) Length of *profilename* (in bytes). If *len* = 0 a null-terminated string must be provided.

optlist An option list describing aspects of profile handling:

- ▶ General option: *errorpolicy* (see Table 1.5)
- ▶ Profile handling options according to Table 8.1:
description, embedprofile, outputintenttype, urls, usage

Table 8.1 Options for *PDF_load_iccprofile()*

option	description
description	(String; only for usage=outputintent) Human-readable description of the ICC profile which will be used along with the output intent.
embedprofile	(Boolean; only for usage=outputintent and usage=pageoutputintent) Control embedding or attaching of the output intent ICC profile: PDF/X-5n: if true, attach the ICC profile to the document as embedded file stream. Default: true PDF 2.0: If true, embed the ICC profile; otherwise an external reference is created. Default: true PDF/X-4p: the option is forced to false, i.e. the profile is referenced externally. All other cases including PDF/X-3/4: the option is forced to true, i.e. the profile is embedded. The sRGB profile is always embedded.
outputintent-type	(Name string; PDF 2.0; not for PDF/A and PDF/X; only for usage=outputintent or pageoutputintent) Custom usage type of the output intent. The type must not start with GTS_ or Plib_. Default: Plib_OutputIntent.
urls	(List of one or more strings; only for PDF 2.0 and PDF/X-4p; required for PDF/X-4p) The ICC profile is not embedded in the generated PDF, but a reference to one or more external profiles is created instead. The list contains URLs which indicate where a referenced output intent ICC profile can be obtained. Sender and receiver should arrange reasonable URL entries. The strings can freely be chosen, but must contain valid URL syntax.
usage	(Keyword) Intended use of the ICC profile. Supported keywords (default: iccbased): iccbased The ICC profile can be used as ICC-based color space for text or graphics, applied to an image, used as default color space or as blending color space for a transparency group. outputintent (Only for PDF/A, PDF/X or PDF 2.0) The ICC profile implicitly specifies the document-wide output intent. The profile can also be used as page-specific output intent in PDF 2.0. pageoutputintent (PDF 2.0) The ICC profile specifies an output intent for one or more pages. The returned ICC profile handle can be supplied to <i>PDF_begin_page_ext()</i> .

- Returns** If *usage=iccbased* the returned ICC profile handle can be used in subsequent calls to `PDF_load_image()` or for setting profile-related options.
 If *usage=pageoutputintent* the returned output intent handle can be used with the *outputintents* option of `PDF_begin_page_ext()`.
 If *errorpolicy=return* the caller must check for a return value of -1 (in PHP: 0) since it signals an error. The returned handle cannot be reused across multiple PDF documents. If the method call fails you can request the reason of the failure with `PDF_get_errmsg()`.
- Details** The named profile is searched according to the profile search strategy. Depending on the intended usage ICC profiles must satisfy the conditions listed in the PDFlib Tutorial. The *sRGB* profile (also available as *srgb*) is always available internally and doesn't have to be configured.
- PDF/A** The document-level output intent can be set using this method or by copying an imported document's output intent using `PDF_process_pdi()`. If only device-independent colors are used in the document no output intent is required.
- PDF/X** The document-level output intent must be set either using this method or by copying an imported document's output intent using `PDF_process_pdi()`.
 PDF/X-4: a document-level output intent ICC profile must be specified and will be embedded.
 PDF/X-4/5: a CMYK output intent profile (i.e. loaded with *usage=outputintent*) cannot be used for an ICCBased color space (i.e. loaded with *usage=iccbased*) in the same document. This requirement is mandated by the PDF/X standard, and applies only to CMYK profiles, but not to grayscale or RGB profiles. If you have a requirement to use the same CMYK ICC profile as in the output intent also as ICCBased color (e.g. for tagging an image), you can simply omit the ICC profile since PDF/X implies that the output intent profile will be used anyway.
 PDF/X-4p: The profile is not embedded, but a reference to an external profile is created. The profile must be available when generating the PDF, and it must also be available to the PDF consumer when viewing or printing the document.
 PDF/X-5n: an n-colorant ICC profile (also called *xCLR* profile) must be supplied for *usage=outputintent*. The externally referenced output intent ICC profile can optionally be included in the document as attachment subject to the *embedprofile* option.
- Scope** If *usage=outputintent* the only allowed scope is *document*; the document-level output intent should be set immediately after `PDF_begin_document()`.
 If *usage=pageoutputintent* the following scopes are allowed: *document*, *page*, *pattern*, *template*, *glyph*.
 If *usage=iccbased* the following scopes are allowed: *document*, *page*, *pattern*, *template*, *glyph*.

8.3 Spot Colors

C++ Java C# *int makespotcolor(String spotname)*

Perl PHP *int makespotcolor(string spotname)*

C *int PDF_makespotcolor(PDF *p, const char *spotname, int reserved)*

Find a built-in spot color name, or make a named spot color from the current fill color.

spotname The name of a built-in spot color, or an arbitrary name for a custom spot color to be defined. This name is restricted to a maximum of 63 Unicode characters.

The special spot color name *All* can be used to apply color to all color separations, which may be useful for painting registration marks. The spot color name *None* produces no visible output on any color separation. The colorant names *Cyan*, *Magenta*, *Yellow* and *Black* always refer to CMYK process colors.

reserved (C language binding only) Reserved, must be 0.

Returns A spot color handle which can be used in subsequent calls to *PDF_setcolor()* or the *fillcolor* and *strokecolor* options of *PDF_set_graphics_option()* and other methods. Spot color handles can be reused across all pages, but not across documents.

Details If *spotname* is known in the internal color tables of PANTONE and HKS colors, and the global *spotcolorlookup* option is *true* (which is default), the specified spot color name and corresponding internal *Lab* alternate color values are used. Otherwise the color values of the current fill color are used to define the appearance of a new spot color.

If *spotname* has already been used in a previous call to *PDF_makespotcolor()*, the return value is the same as in the earlier call and does not use the current fill color as alternate color.

This method can usually be avoided by supplying options of type *Color* directly. This way definition and use of a spot color can be achieved in a single option list without the need for passing spot color handles (see Table 1.2 for examples).

Table 8.2 lists relevant global options which affect the behavior of this method (see Section 2.1, »Global Options«, page 25).

Scope *page, pattern, template, glyph, document*; the current fill color must not be a spot color, DeviceN color, or pattern if a custom spot color is defined.

Table 8.2 Spot color related global options for *PDF_set_option()*

option	description
<i>spotcolorlookup</i>	(Boolean) If false, PDFlib does not use its internal database of spot color names. This can be used to provide custom definitions of known spot colors, which may be required as a workaround to match the definitions used by other applications. This feature should be used with care and is not recommended. Default: true

8.4 DeviceN Colors

C++ Java C# **int create_devicen(String optlist)**

Perl PHP **int create_devicen(string optlist)**

C **int PDF_create_devicen(PDF *p, const char *optlist)**

Create a DeviceN color space with an arbitrary number of color components.

optlist Option list specifying DeviceN color creation options

- ▶ General option: *errorpolicy* (see Table 1.5)
- ▶ DeviceN options according to Table 8.1: *alternate*, *names*, *process*, *subtype*, *transform*

Returns A DeviceN color space handle which can be used with the *fillcolor* and *strokecolor* options of *PDF_set_graphics_option()* and other methods. The DeviceN color space handle can be used until the end of the enclosing document scope.

By default, this method throws an exception in case of an error. However, this behavior can be changed to an error return value of -1 (in PHP: 0) with the *errorpolicy* option.

PDF/A PDF/A-1/2/3: The color space supplied in the *alternate* option must adhere to the following restrictions:

alternate=devicegray requires a grayscale, RGB or CMYK output intent or the *defaultgray* option.

alternate=devicergb requires an RGB output intent or the *defaultrgb* option.

alternate=devicecmymk requires a CMYK output intent or the *defaultcmymk* option.

PDF/A-2/3: *PDF_makespotcolor()* must be called before *PDF_create_devicen()* for all custom spot colors used in the DeviceN color space.

PDF/X The color space supplied in the *alternate* option must adhere to the following restrictions:

PDF/X-3: *alternate=iccbased* and *lab* require a grayscale, RGB or CMYK output intent.

PDF/X-3/4/5n: *alternate=devicegray* requires a grayscale or CMYK output intent or the *defaultgray* option.

alternate=devicergb requires an RGB output intent or the *defaultrgb* option.

alternate=devicecmymk requires a CMYK output intent or the *defaultcmymk* option.

PDF/X-4: the *colorspace* suboption of the *process* option must match the PDF/X output intent.

PDF/X-4 and PDF/X-5n: *PDF_makespotcolor()* must be called before *PDF_create_devicen()* for all custom spot colors used in the DeviceN color space. In PDF/X-5n spot colors found in the colorant list of the output intent are excluded from this requirement.

PDF/X-5n: The options *subtype=nchannel* and *process* are not allowed.

Scope any except *object*

Table 8.3 Options for `PDF_create_devicen()`

option	description
alternate	<p>(Keyword or option list; restrictions apply to PDF/A and PDF/X; required) Alternate color space for the DeviceN color space. The following keywords can be supplied: <code>devicegray</code>, <code>devicergb</code>, <code>devicecmyk</code>, <code>lab</code>. Alternatively, an option list with the following suboption can be supplied:</p> <p>iccbased (Keyword or ICC profile handle) ICC profile specified by a handle or the keyword <code>srgb</code>. The ICC profile must have been loaded with <code>usage=iccbased</code>.</p>
names	<p>(List of name strings; required) List containing up to 32 colorant names (PDF 1.4: up to 8 colorant names). All names must be different from each other except <code>None</code> which may appear more than once. The colorant name <code>All</code> is not allowed. The colorant names <code>Cyan</code>, <code>Magenta</code>, <code>Yellow</code> and <code>Black</code> always refer to CMYK process colors.</p> <p>If <code>subtype=nchannel</code> the colorant name <code>None</code> is not allowed and the names of all spot colors used in the DeviceN color space must be known to PDFlib, i.e. they must be included in the internal spot color database or <code>PDF_makespotcolor()</code> must have been called earlier.</p>
process	<p>(Option list; PDF 1.6 or above; required if <code>subtype=nchannel</code> and the <code>names</code> option includes one or more process colors; not allowed for PDF/X-5n):</p> <p>colorspace (Keyword; required) Process color space: <code>devicegray</code>, <code>devicergb</code>, or <code>devicecmyk</code>. PDF/X-4: the color space must match the PDF/X output intent.</p> <p>components (List of strings; required) Names of all components of the process color space specified in the <code>colorspace</code> suboption. The list of component names must not include the component name <code>None</code> or built-in or custom spot color names. If the <code>names</code> option includes one or more process colors these must match the names supplied here, but the names may differ from the usual names (e.g. <code>Process Cyan</code> may be used instead of <code>Cyan</code> for <code>colorspace=devicecmyk</code>)</p>
subtype	<p>(Keyword; PDF 1.6) Preferred treatment of the <code>colorspace</code> (default: <code>devicen</code>):</p> <p>devicen The color space is treated as plain DeviceN color space.</p> <p>nchannel (Not allowed for PDF/X-5n) The color space is treated as NChannel color space. In this case the <code>names</code> option is subject to certain restrictions.</p>
transform	<p>(String; required) PostScript code for the transform method of the DeviceN color space. The transform method must convert <code>N</code> tint values in the range 0..1 to values in the alternate color space. The number of output values must correspond to the number of components of the alternate color space. Since the PostScript code must be enclosed with brace characters <code>{</code> and <code>}</code>, and the option list string also requires surrounding brace characters if it contains whitespace, the option value generally is enclosed by two pairs of brace characters.</p>

8.5 Shadings and Shading Patterns

```
C++ Java C# int shading(String type, double xo, double yo, double x1, double y1,  
double c1, double c2, double c3, double c4, String optlist)
```

```
Perl PHP int shading(string type, float xo, float yo, float x1, float y1,  
float c1, float c2, float c3, float c4, string optlist)
```

```
C int PDF_shading(PDF *p, const char *type, double xo, double yo, double x1, double y1,  
double c1, double c2, double c3, double c4, const char *optlist)
```

Define a color shading (color gradient) between two or more colors.

type The type of the shading; must be *axial* for linear shadings or *radial* for circle-like shadings.

xo, yo Starting point (for *type=axial*), or center of the starting circle (for *type=radial*). The values are interpreted in user coordinates.

x1, y1 Ending point (for *type=axial*), or center of the ending circle (for *type=radial*). The values are interpreted in user coordinates.

c1, c2, c3, c4 Color values of the shading's end color, interpreted in the color space of the shading's start color. If the current fill color space is a spot color space *c1* is ignored, and *c2* contains the tint value. The parameters *c1*, *c2*, *c3*, and *c4* are ignored if the *endcolor* or *stopcolors* option is supplied. Also, they cannot be used for shadings in a DeviceN color space with $N > 4$.

optlist Option list describing aspects of the shading according to Table 8.4. The following options can be used:

antialias, boundingbox, end, endcolor, extendo, extend1, N, ro, r1, startcolor, stopcolors, type

Returns A shading handle that can be used in subsequent calls to *PDF_shading_pattern()* and *PDF_shfill()* during the enclosing *document* scope.

Details This method creates a shading between the colors supplied in the *startcolor* and *endcolor* options. If the option *stopcolors* is supplied, a shading is created between all entries in the specified color list. All colors must be from the same color space and must not use pattern color space. If different spot colors are used all of the following conditions must be satisfied:

- ▶ all stop colors are either spot colors known to PDFlib internally, custom spot colors with Lab alternate values, or direct Lab colors; if a direct Lab color is supplied PDFlib creates an artificial spot color from it.
- ▶ the number of different spot colors in the list of stop colors does not exceed 19 (8 for PDF 1.4);
- ▶ all spot color names are different from *All*.

If the spot colors supplied as stop colors violate one of the conditions above an exception is thrown.

As an alternative to the *startcolor* option the current fill color can be used. As an alternative to the *endcolor* option the values *c1*, *c2*, *c3*, and *c4* can be used. As an alternative to the combination of the *startcolor* and *endcolor* options the option *stopcolors* can be used. In this case at least two colors must be specified.

Scope any except object

Table 8.4 Options for `PDF_shading()` and suboptions for the shading graphics appearance option

option	description
antialias	(Boolean) Specifies whether to activate antialiasing for the shading. Default: false
boundingbox	(Rectangle) A rectangle defining the shading's bounding box in user coordinates. The bounding box will be applied as a temporary clipping path when the shading is painted (in addition to the current clipping path which may be in effect). This option may be useful to clip the shading without applying <code>PDF_clip()</code> . Default for <code>PDF_shading()</code> : no clipping Default for the shading graphics appearance option: the matchbox or table cell rectangle
end	(List of 2 floats or percentages; not for <code>PDF_shading()</code>) Coordinates of the ending point (for type=axial) or a point on the ending circle (for type=radial), specified as percentages of the rectangle's width and height or in user coordinates. Default: {100% 100%}
endcolor	(Color; ignored if stopcolors is supplied; one of the options endcolor and stopcolors is required for the shading graphics appearance option) The shading's end color. Default for <code>PDF_shading()</code> : the last color provided in the stopcolors option if specified, otherwise the values supplied in the parameters c1, c2, c3, and c4
extendo	(Boolean) Specifies whether to extend the shading beyond the starting point or circle. Default: false
extendi	(Boolean) Specifies whether to extend the shading beyond the ending point or circle. Default: false
N	(Positive Float) Exponent for the color transition method. Default: 1
ro	(Float; only for type=radial) Radius of the starting circle in user coordinates. Default: 0
ri	(Float; only for type=radial) Radius of the ending circle in user coordinates. Default for <code>PDF_shading()</code> : 0. Default for the shading graphics appearance option: distance between start and end
start	(List of 2 floats or percentages; not for <code>PDF_shading()</code>) Coordinates of the starting point (type=axial) or the center of the starting circle (type=radial), specified as percentages of the rectangle's width and height or in user coordinates. Default: {0% 0%}
startcolor	(Color; ignored if stopcolors is supplied) The shading's start color. Default: the first color provided in the stopcolors option if specified, otherwise the color specified in the option fillcolor for the matchbox or table cell
stopcolors	(List of pairs; one of the options endcolor and stopcolors is required for the shading graphics appearance option) List of two or more colors for the shading. Each pair contains a float value or a percentage for the position of an intermediate shading color in the inclusive interval 0...1 and the corresponding color value. All color values must use the same color space except for spot colors with Lab alternate colors as described in the Details section above. The positions must be sorted in ascending order, but adjacent position values may be identical. If no value for position 0 is present the first list entry is used; if no value for position 1 is present the last list entry is used. If this option is supplied the options startcolor and endcolor, the parameters c1, . . . , c4 and the current fill color are ignored. Examples: stopcolors={0 red 0.4 magenta 0.75 green 1 black} stopcolors={0% {cmyk 1 0 0 0} 33% {cmyk 0 0.4 0.3 0} 100% {cmyk 0 0 0.2 0.8}} stopcolors={0% {spotname {PANTONE 123 U} 1} 100% {spotname {PANTONE 289 U} 1} }
type	(Keyword) Shading type: axial for a linear shading or radial for a circle-like shading. Default: value of the parameter type if present (only for <code>PDF_shading()</code>), otherwise: axial

C++ Java C# **int shading_pattern(int shading, String optlist)**

Perl PHP **int shading_pattern(int shading, string optlist)**

C **int PDF_shading_pattern(PDF *p, int shading, const char *optlist)**

Define a shading pattern using a shading object.

shading Shading handle returned by [PDF_shading\(\)](#).

optlist Option list describing details of the shading pattern:

- ▶ Graphics appearance option according to Table 7.1: *gstate*
- ▶ Transformation option according to Table 8.5: *transform*

Returns A pattern handle that can be used in subsequent calls to [PDF_setcolor\(\)](#) and for the options *fillcolor* and *strokecolor* during the enclosing *document* scope.

Details The following steps are required to apply a shading to an object:

- ▶ A shading handle must be retrieved using [PDF_shading\(\)](#).
- ▶ A shading pattern must be defined based on this shading using [PDF_shading_pattern\(\)](#).
- ▶ The pattern handle can be supplied to [PDF_setcolor\(\)](#) or the options *fillcolor* and *strokecolor* to set the current color to the shading pattern.

Scope any except *object*

Table 8.5 Additional option for [PDF_shading_pattern\(\)](#)

option	description
transform	<p>(Option list) A list which defines the transformation that maps the shading pattern coordinate system to the default coordinate system of the target page, template or glyph description where the shading pattern is used. The concatenation of the shading pattern matrix with that of the target page, template or glyph description establishes the coordinate system within which the shading pattern is interpreted.</p> <p>The list contains pairs of a keyword and a float list according to Table 9.12 where each pair defines a transformation.</p>

C++ Java C# **void shfill(int shading)**

Perl PHP **shfill(int shading)**

C **void PDF_shfill(PDF *p, int shading)**

Fill an area with a shading.

shading Shading handle returned by [PDF_shading\(\)](#).

Details This method allows shadings to be used without involving [PDF_shading_pattern\(\)](#) and [PDF_setcolor\(\)](#) or the options *fillcolor* and *strokecolor*. However, it works only for simple shapes where the geometry of the object to be filled is the same as that of the shading itself. Since the current clip area will be shaded (subject to the *extendo* and *extend1* options of the shading) this method will generally be used in combination with [PDF_clip\(\)](#).

Scope *page*, *pattern* (only if *painttype=colored*), *template*, *glyph* (only if the glyph's *colorized* option is *true*)

8.6 Tiling Patterns

C++ Java C# `int begin_pattern_ext(double width, double height, string optlist)`

Perl PHP `int begin_pattern_ext(float width, float height, string optlist)`

C `int PDF_begin_pattern_ext(PDF *p, double width, double height, const char *optlist)`

Start the definition of a tiling pattern with options.

width, height The dimensions of the pattern's bounding box in the pattern coordinate system.

optlist Option list describing pattern details:

- ▶ Pattern-specific options according to Table 8.6:
painttype, tilingtype, xstep, ystep
- ▶ Template options according to Table 9.9: *boundingbox, topdown*
- ▶ Common XObject options according to Table 9.11 (although no XObject is involved):
defaultcmyk, defaultgray, defaultrgb, transform

Returns A pattern handle that can be used in subsequent calls to `PDF_setcolor()` and for the options `fillcolor` and `strokecolor` during the enclosing *document* scope.

Details This method starts the definition of a tiling pattern. It resets all text, graphics, and color state parameters to their default values. The *transform* option defines the mapping of the pattern coordinate system to the coordinate system of the page, template or glyph description where the pattern is used.

Scope any except *object*; this method starts *pattern* scope, and must always be paired with a matching `PDF_end_pattern()` call.

Table 8.6 Options for `PDF_begin_pattern_ext()`; some options can also be used with `PDF_begin_template_ext()`.

option	description
painttype	(Keyword) Indicates whether the pattern contains color specifications on its own or is used as a stencil which will be colored with the current fill or stroke color when the pattern is used for filling or stroking (default: <code>colored</code>):
colored	The pattern is colored with one or more calls to <code>PDF_setcolor()</code> or the options <code>fillcolor</code> / <code>strokecolor</code> . The pattern description may place images, PDF pages or graphics.
uncolored	The pattern does not contain any color specification. Instead, the current fill or stroke color is applied when the pattern is used for filling or stroking. Image masks may be used, but not any images, placed PDF pages or graphics. Before using the pattern, <code>PDF_setcolor()</code> or the options <code>fillcolor</code> / <code>strokecolor</code> must be called to set the current color with a color space which is not based on a pattern.

Table 8.6 Options for `PDF_begin_pattern_ext()`; some options can also be used with `PDF_begin_template_ext()`.

option	description
tilingtype	(Keyword) Controls adjustments to the spacing of pattern tiles (default: <code>constantspacing</code>): constantspacing <i>Pattern cells are spaced consistently, i.e. by a multiple of a device pixel. The PDF consumer may need to distort the pattern cell slightly by making small adjustments to <code>xstep</code>, <code>ystep</code>, and the transformation matrix.</i> nodistortion <i>Pattern cells are not distorted, but the spacing between pattern cells may vary by as much as one device pixel, both horizontally and vertically, when the pattern is painted. This achieves the requested spacing on average but not necessarily for each individual pattern cell.</i> fastertiling <i>Pattern cells are spaced consistently as with <code>constanttiling</code>, but with additional distortion permitted to enable a more efficient implementation.</i>
xstep	(Float different from 0; values close to 0 should be avoided) Horizontal spacing between pattern cells in pattern coordinates. Default: <code>width</code>
ystep	(Float different from 0; values close to 0 should be avoided) Vertical spacing between pattern cells in pattern coordinates. Default: <code>height</code>

C++ Java C# **`void end_pattern()`**

Perl PHP **`end_pattern()`**

C **`void PDF_end_pattern(PDF *p)`**

Finish the definition of a tiling pattern.

Scope `pattern`; this method terminates `pattern` scope, and must always be paired with a matching `PDF_begin_pattern_ext()` call.

9 Image, SVG and Template Methods

API methods in this chapter:

- ▶ `PDF_load_image()`
- ▶ `PDF_close_image()`
- ▶ `PDF_fit_image()`
- ▶ `PDF_info_image()`
- ▶ `PDF_load_graphics()`
- ▶ `PDF_fit_graphics()`
- ▶ `PDF_info_graphics()`
- ▶ `PDF_begin_template_ext()`
- ▶ `PDF_end_template_ext()`

9.1 Images

C++ Java C# `int load_image(String imagetype, String filename, String optlist)`

Perl PHP `int load_image(string imagetype, string filename, string optlist)`

C `int PDF_load_image(PDF *p,
const char *imagetype, const char *filename, int len, const char *optlist)`

Open a disk-based or virtual image file subject to various options.

imagetype The string *auto* instructs PDFlib to automatically detect the image file type (not possible for raw images which must always be loaded with the proper image type). Explicitly specifying the image format with one of the strings *bmp*, *jb1g2*, *jpeg*, *jpeg2000* (PDF 1.5 and above), *png*, *raw*, or *tiff* offers slight performance advantages. Details of the image formats are discussed in the PDFlib Tutorial.

filename (Name string; will be interpreted according to the global *filenamehandling* option, see Table 2.1) Name of the image file to be opened. This must be the name of a disk-based or virtual file; PDFlib will not pull image data from URLs.

If a file with the specified file name cannot be found and *imagetype=auto* PDFlib tries to determine the appropriate file name suffix automatically; it appends all suffixes from the following list (in both lowercase and uppercase) to the specified *filename* and tries to locate a file with that name in the directories specified in the search path:

.bmp, .jb1g2, .jb2, .jpg, .jpeg, .jpx, .jp2, .jpf, .j2k, .png, .raw, .tif, .tiff

len (C language binding only) Length of *filename* (in bytes). If *len = 0* a null-terminated string must be provided.

optlist An option list specifying image-related properties according to Table 9.1. The following options can be used:

- ▶ General options: *errorpolicy* (see Table 1.5)
- ▶ Color-related options:
chromakey, *colorize*, *decode*, *honoriccprofile*, *iccprofile*, *invert*, *renderingintent*
- ▶ Clipping, masking, and transparency options:
alphachannelname, *clippingpathname*, *honorclippingpath*, *ignoremask*, *mask*, *masked*

- ▶ Special PDF features: *interpolate*, *templateoptions*
- ▶ Common XObject options can be applied to the generated Image XObject (see Table 9.11):
associatedfiles, *georeference*, *layer*, *metadata*, *pdfvt*
- ▶ Option for analyzing the image without writing PDF output: *infomode*
- ▶ Options for processing the image data:
ignoreorientation, *page*, *passthrough*
- ▶ Options for JBIG2 and raw images according to Table 9.2:
bpc, *components*, *copyglobals*, *height*, *imagehandle*, *inline*, *width*
- ▶ Encoding option for C and Perl, PHP, Ruby with *stringformat=legacy*:
hypertextencoding (see Table 2.1)

Returns An image handle (or template handle if *templateoptions* is supplied) which can be used in subsequent image-related calls. If *errorpolicy=return* the caller must check for a return value of -1 (in PHP: 0) since it signals an error. The returned image handle can not be re-used across multiple PDF documents. If the method call fails you can request the reason of the failure with `PDF_get_errmsg()`.

Details This method opens and analyzes a raster graphics file in one of the supported formats as determined by the *imagetype* parameter, and copies the relevant image data to the output document. This method will not have any visible effect on the output. In order to actually place the imported image somewhere in the generated output document, `PDF_fit_image()` must be used. Opening the same image more than once per generated document is not recommended because the actual image data will be copied to the output document more than once. If the application cannot prevent this situation you can remove redundant image data with the *optimize* option of `PDF_begin_document()`.

PDFlib opens the image file with the provided *filename*, processes the contents, and closes the file before returning from this call. Although images can be placed multiply within a document with `PDF_fit_image()`, the actual image file will not be kept open after this call.

PDF/A Some options are restricted.

Grayscale images require an output intent (any type) or the *defaultgray* option.

CMYK images require an ICC profile, a CMYK output intent or the *defaultcmyk* option.

PDF/A-2/3: JPEG 2000 images must satisfy certain conditions; see PDFlib Tutorial for details.

PDF/VT This call may fail if the *usestransparency=false* option was specified in `PDF_begin_document()`, but the imported image contains transparency nevertheless.

PDF/X Some options are restricted.

PDF/X-3: JBIG2 images are not allowed.

PDF/X-3/4/5p/5pg: Grayscale images require a grayscale or CMYK output intent or the *defaultgray* option.

CMYK images require an ICC profile, a CMYK output intent or the *defaultcmyk* option.

JPEG 2000 images must satisfy certain conditions; see PDFlib Tutorial for details.

PDF/X-5n: Grayscale images can only be used if the output intent contains the colorant *Black* or the *defaultgray* option has been set.

CMYK color can only be used if the output intent contains all of the colorants *Cyan*, *Magenta*, *Yellow* and *Black* or the *defaultcmyk* option has been set.

Scope any except *object*; if *inline=true* is supplied, this method can only be called in page, pattern, template or glyph scope. It should be paired with a matching call to *PDF_close_image()*.

Table 9.1 Options for *PDF_load_image()*

option	description
alphachannel-name	(Name string; only for TIFF images; ignored if <i>ignoremask=true</i>) Read the alpha channel with the specified name from the image file and apply it as a soft mask to the image. The named channel must be present in the image file. Default: the first alpha channel in the image
chromakey	(List of <i>n</i> or $2 \times n$ integers in the range 0 to $2^{\text{bitspercomponent}} - 1$, where <i>n</i> is the number of components in the image color space; forces <i>ignoremask=true</i> ; not allowed if <i>mask</i> is specified; not for <i>imagetype=jpeg</i> and <i>jpeg2000</i> ; ignored for Lab images) Activate chroma key masking (also called color key masking) for a single color or a range of colors. Each pair in the list contains the inclusive lower and upper bounds of a color component range. Pixels where all color components (before applying any decode values or color inversion with the <i>invert</i> option) fall in the specified ranges are treated as transparent, i.e. they are not painted but allow the background to shine through. If <i>n</i> values instead of <i>n</i> pairs are supplied each component range contains only a single color value, i.e. each list value describes both lower and upper bound for a color component. This option overrides chroma key values found in PNG images.
clipping-pathname	(String; only for <i>imagetype=tiff</i> and <i>jpeg</i> ; ignored if <i>honorclippingpath=false</i>) Read the path with the specified name from the image file and use it as clipping path. The named path must be present in the image file. The special name <i>Work</i> Path can be used to address a temporary path created in Photo-shop. Default: name of the path which is provided as clipping path in the image file
colorize	(Spot or DeviceN color handle, or color option defining a spot color; this option is ignored if the <i>iccprofile</i> option is provided) Colorize the image with a spot or DeviceN color. Colorizing an image with a spot color works only with black and white or grayscale images. Colorizing an image with a spot color reverses the polarity interpretation of the color values, i.e. color value zero results in the maximum amount of spot color. You can compensate this effect with the <i>invert</i> option. Colorizing an image with a DeviceN color works only with raw images. The number of image components must match the number <i>N</i> of color components in the DeviceN color space. DeviceN color spaces are always subtractive, i.e. color value 0 results in white.
decode	(List of $2 \times n$ float or percentage values, where <i>n</i> is the number of components in the image color space; not allowed for JPEG 2000 images or if <i>mask</i> is specified) List of <i>n</i> pairs of numbers which map image sample values to values in the image's color space. Each pair describes the target color values to which the lowest and highest component values 0 and $(2^{\text{bitspercomponent}}) - 1$ are mapped; intermediate values are interpolated linearly. For palette images a single decode pair is required, not multiple pairs for the base color space (e.g. a single pair for images with an RGB palette). Since the decode values apply to the palette index and not the palette entries this option should not be applied to palette images.
honor-clippingpath	(Boolean; only for <i>imagetype=tiff</i> and <i>jpeg</i> ; forced to false for inline images) Read the clipping path from the image file if available, and apply it to the image. Default: true
honor-iccprofile	(Boolean; only for <i>imagetype=jpeg</i> , <i>jpeg2000</i> , <i>png</i> , and <i>tiff</i> ; forced to false if the <i>colorize</i> option is specified or if the embedded profile is identical to the PDF/X-4/5 CMYK output intent) Honor an ICC profile which may be available in the image, either directly by embedding or indirectly (e.g. via a reference in the Exif marker) and apply it to the image. Default: true
iccprofile	(ICC handle or keyword) ICC profile which is applied to the image. The following keywords are supported: none No ICC profile is applied. srgb The sRGB profile is applied. Default: an embedded profile (or equivalent Exif information in the image file) if present in the image and <i>honoriccprofile=true</i> ; if no embedded or user-specified ICC profile is available the sRGB profile is applied to RGB images in BMP, JPG, PNG, or TIFF format.

Table 9.1 Options for `PDF_load_image()`

option	description
ignoremask	(Boolean; must be set to <code>true</code> in PDF/X-3 and PDF/A-1 for images with an alpha channel) Ignore transparency information and alpha channels in the image. Default: <code>false</code>
ignore-orientation	(Boolean; only for <code>imagetype=tiff</code> and <code>jpeg</code>) Ignores any orientation information in the image. This may be useful for compensating incorrect orientation info in the image data. Default: <code>false</code>
infomode	(Boolean) If <code>true</code> the image is loaded, but no pixel data is written to the output. Image properties can be queried with <code>PDF_info_image()</code> , but the image cannot be placed on a page with <code>PDF_fit_image()</code> or other methods. This option may be useful to check images without any side effects on the PDF output. If <code>false</code> the pixel data is written to the PDF output immediately. Default: <code>false</code>
interpolate	(Boolean; must be <code>false</code> for PDF/A) Enables image interpolation to improve the appearance on screen and paper. This is useful for bitmap images for glyph descriptions in Type 3 fonts. Default: <code>false</code>
invert	(Boolean; not for <code>imagetype=jpeg2000</code> unless <code>mask=true</code>) Invert the image (swap light and dark colors). Default: <code>false</code>
mask	(Boolean; only for bitmap images, i.e. one bit per pixel; forces <code>ignoremask=true</code>). Prepare the image for use as a stencil mask. If the image itself is placed on the page black pixels are painted with the current fill color while white pixels are ignored, i.e. the background remains unchanged. If the image is used to mask another base image with the <code>masked</code> option the contents of the base image are shown where the mask is black and ignored where the mask is white.
masked	(Image handle; PDF/A-1 and PDF/X-3: only allowed with 1-bit masks; ignored if the image contains an alpha channel and <code>ignoremask=false</code>) Handle for a grayscale image which is applied as alpha channel to the current image. In PDF/A-1 and PDF/X-1/3 bitmap masks must have been loaded with the <code>mask</code> option. If the mask image has been loaded with the <code>decode</code> option the <code>decode</code> array must be <code>{0 1}</code> or <code>{1 0}</code> .
page	(Integer; only for <code>imagetype=jbig2</code> and <code>tiff</code> ; must be 1 if used with other formats) Extract the image with the given number from a multi-page image file. The first image has the number 1. The call fails if the requested page cannot be found in the image file. Default: 1
passthrough	(Boolean; only for <code>imagetype=tiff</code> or <code>jpeg</code>) Controls handling of image data. TIFF images (default: <code>true</code>): If <code>true</code> , compressed TIFF image data is directly passed through to the PDF output if possible. Setting this option to <code>false</code> may help in cases where a TIFF image contains damaged or incomplete data. JPEG images (default: <code>false</code>): If <code>false</code> , PDFlib transcodes JPEG image data (without uncompressing the pixel data) for compatibility with PDF. If <code>true</code> , JPEG image data is directly copied to the PDF output. This option is ignored for multiscan and certain CMYK JPEG images. Setting this option to <code>true</code> may speed up processing, but certain rare JPEG flavors won't display correctly in Acrobat.
rendering-intent	(Keyword) Color rendering intent (default: <code>Auto</code>): Auto, AbsoluteColorimetric, RelativeColorimetric, Saturation, Perceptual
template-options	(Option list) Create a template (PDF Form XObject) according to the supplied option list (which may be empty). This may be useful for creating luminosity soft masks or templates for form field icons which consist of an image only. A handle for the generated template is returned. The option list contains common XObject options according to Table 9.11.

Table 9.2 Additional options for `PDF_load_image()` with `imagetype=jbig2`, `jpeg` or `raw`

option	description
bpc	(Integer; only for <code>imagetype=raw</code> ; required in this case) Number of bits per component; must be 1, 2, 4, 8 or 16 (in PDF 1.4 the value 16 is not allowed)
components	(Integer; only for <code>imagetype=raw</code> ; required in this case unless the <code>colorize</code> option is provided) Number of image components (channels); must be 1, 3, or 4. This option is ignored if the <code>colorize</code> option is supplied since the number of components is determined from the color space.
copyglobals	(Keyword; only for <code>imagetype=jbig2</code>) Specify which global segments in a JBIG2 stream will be copied to the PDF. If the JBIG2 stream doesn't contain any global segments this option will not have any effect (default: <code>current</code>): <ul style="list-style-type: none"> all Copy the global segments for all pages in the JBIG2 stream to the PDF. This should be used if more than one page from the same JBIG2 stream will be imported. The <code>imagehandle</code> option should be used if more pages from the same JBIG2 stream will be imported later. current Copy only the global segments required for the current page (i.e. the page specified in the <code>page</code> option) in the JBIG2 stream to the PDF. This should be used if no more pages from the same JBIG2 stream will be imported.
height	(Integer; only for <code>imagetype=raw</code> ; required in this case) Image height in pixels.
imagehandle	(Image handle; only for <code>imagetype=jbig2</code>) Add a reference to an existing global segment attached to another image created from the same JBIG2 stream which must have been loaded earlier with the <code>copyglobals=all</code> option. It is an error to refer to an image which has been created from a different file than the current JBIG2 stream. The specified image handle must not have been closed. Default: no image handle, i.e. a new PDF object will be created with all required global segments for the current page only
inline	(Boolean; only for <code>imagetype=jpeg</code> and <code>raw</code> ; not allowed if <code>templateoptions</code> is supplied) If true, the image is written directly into the content stream of a page, pattern, template, or glyph description. This option implicitly calls <code>PDF_fit_image()</code> and <code>PDF_close_image()</code> (see PDFlib Tutorial). This option is recommended only for bitmap glyphs of Type 3 fonts and should not be used in other situations. If this option is supplied <code>PDF_close_image()</code> must not be called. Default: <code>false</code>
width	(Integer; only for <code>imagetype=raw</code> ; required in this case) Image width in pixels

C++ Java C# **void close_image(int image)**

Perl PHP **close_image(int image)**

C **void PDF_close_image(PDF *p, int image)**

Close an image or template.

image A valid image or template handle retrieved with `PDF_load_image()` or `PDF_begin_template_ext()`.

Details This method only affects PDFlib's associated internal image structure. If the image has been opened from file, the actual image file is not affected by this call since it has already been closed at the end of the corresponding `PDF_load_image()` call. An image handle cannot be used any more after it has been closed with this method.

Scope any except *object*; must always be paired with a matching call to `PDF_load_image()` (unless the `inline` option has been used) or or `PDF_begin_template_ext()`.

C++ Java C# **void fit_image(int image, double x, double y, String optlist)**

Perl PHP **fit_image(int image, float x, float y, string optlist)**

C **void PDF_fit_image(PDF *p, int image, double x, double y, const char *optlist)**

Place an image or template on the page, subject to various options.

image A valid image or template handle retrieved with `PDF_load_image()` or `PDF_begin_template_ext()`. The image must not have been loaded with `infomode=true`. Template handles can only be used if their definition has been finalized with `PDF_end_template_ext()`.

x, y The coordinates of the reference point in the user coordinate system where the image or template will be located, subject to various options.

optlist An option list specifying image fitting and processing options. The following options are supported:

- ▶ Fitting options according to Table 6.1:
boxsize, blind, dpi, fitmethod, matchbox, orientate, position, rotate, scale, showborder
- ▶ Options for image processing according to Table 9.3:
adjustpage, gstate, ignoreclippingpath, ignoreorientation
- ▶ Option for abbreviated structure element tagging according to Table 14.2 (only allowed in *page* scope): *tag*

Details The image or template (collectively referred to as an object below) will be placed relative to the reference point (*x, y*). By default, the lower left corner of the object will be placed at the reference point. However, the *orientate, boxsize, position,* and *fitmethod* options can modify this behavior. By default, an image will be scaled according to its resolution value(s). This behavior can be modified with the *scale* and *fitmethod* options.

PDF/UA Images must be tagged as *Artifact* or *Figure*

Scope *page, pattern* (only if the pattern's *painttype* is *colored* or the image is a mask), *template, glyph* (only if the glyph's *colorized* option is *true* or if the image is a mask); this method can be called an arbitrary number of times on arbitrary pages, as long as the image handle has not been closed with `PDF_close_image()`.

Table 9.3 Options for image, graphics, PDF page and template processing with `PDF_fit_image()`, `PDF_fit_graphics()`, `PDF_fit_pdi_page()`, `PDF_fill_imageblock()`, `PDF_fill_graphicsblock()` and `PDF_fill_pdfblock()`

option	description
adjustpage	(Boolean; only effective in page scope; not allowed if the <i>topdown</i> option has been supplied in <code>PDF_begin_page_ext()</code> ; not for <code>PDF_fill_*block()</code>) Adjust the dimensions of the current page to the object such that the upper right corner of the page coincides with the upper right corner of the object plus (<i>x, y</i>) with the method parameters <i>x</i> and <i>y</i> . The <i>MediaBox</i> will be adjusted, and all other box entries will be reset to their defaults. With the value <i>o</i> for the <i>position</i> option the following useful cases shall be noted: <i>x</i> >= <i>o</i> and <i>y</i> >= <i>o</i> The object is surrounded by a white margin. This margin has thickness <i>y</i> in horizontal direction and thickness <i>x</i> in vertical direction. <i>x</i> < <i>o</i> and <i>y</i> < <i>o</i> Horizontal and vertical strips will be cropped from the image. Default: false

Table 9.3 Options for image, graphics, PDF page and template processing with `PDF_fit_image()`, `PDF_fit_graphics()`, `PDF_fit_pdi_page()`, `PDF_fill_imageblock()`, `PDF_fill_graphicsblock()` and `PDF_fill_pdfblock()`

option	description
gstate	(Option list with options for <code>PDF_create_gstate()</code> or <code>gstate handle</code>) Graphics state options or handle. The graphics state affects all graphical elements created with this method. Default: no graphics state, i.e. current settings are used
ignore-clippingpath	(Boolean; only for TIFF and JPEG images) A clipping path which may be present in the image file is ignored. Default: false, i.e. the clipping path is applied
ignore-orientation	(Boolean; only for TIFF and JPEG images) Ignore any orientation information in the image. This may be useful for compensating wrong orientation info. Default: the value of the <code>ignoreorientation</code> option in <code>PDF_load_image()</code>

C++ Java C# **double info_image(int image, String keyword, String optlist)**

Perl PHP **float info_image(int image, string keyword, string optlist)**

C **double PDF_info_image(PDF *p, int image, const char *keyword, const char *optlist)**

Format an image or template and query metrics and other properties.

image A valid image or template handle retrieved with `PDF_load_image()` or `PDF_begin_template_ext()`.

keyword A keyword specifying the requested information:

- ▶ Keywords for querying the results of object fitting according to Table 6.3: `boundingbox`, `fitscalex`, `fitscaley`, `height`, `objectheight`, `objectwidth`, `width`, `x1`, `y1`, `x2`, `y2`, `x3`, `y3`, `x4`, `y4`
- ▶ Additional keywords according to Table 9.4: `clippingpath`, `checkcolorspace`, `filename`, `iccprofile`, `imageheight`, `imagemask`, `imagetype`, `imagewidth`, `infomode`, `mirroringx`, `mirroringy`, `orientation`, `resx`, `resy`, `strips`, `transparent`, `xid`

optlist The following options are supported:

- ▶ Options for `PDF_fit_image()`. Options which are not relevant for determining the value of the requested keyword are ignored.
- ▶ Option for switching between underlying image and template: `useembeddedimage`

Returns The value of some image property as requested by `keyword`. If the requested property is not available in the image file, the method returns 0. If an object handle is requested (e.g. `clippingpath`) this method returns a handle to the object, or -1 (in PHP: 0) if the object is not available. If the requested keyword produces text, a string index is returned, and the corresponding string must be retrieved with `PDF_get_string()`.

Details This method performs all calculations required for placing the image according to the supplied options, but will not actually create any output on the page. The image reference point is assumed to be {0 0}.

Scope any except object

Table 9.4 Keywords for `PDF_info_image()`

keyword	description
<code>clippingpath</code>	Path handle of the image's clipping path, or -1 (in PHP: 0) if no clipping path is present
<code>checkcolorspace</code>	1 if the image or template can safely be placed on the current page without risking a color-related violation of PDF/A or PDF/X; 0 otherwise. This check takes into account the page's default color space which is not checked when loading the image.
<code>filename</code>	String index for the name of the image file (including a searchpath directory if applicable), or -1 for templates
<code>iccprofile</code>	Handle for the ICC profile embedded in the image or -1 (in PHP: 0) if no profile is present
<code>imageheight</code>	Images: height in pixels Templates: user-supplied height
<code>imagemask</code>	Image handle of the mask associated with the image, or -1 (in PHP: 0) if no mask is attached
<code>imagetype</code>	String index for the type (format) of the image: bmp, jbig2, jpeg, jpeg2000, png, raw, tiff for raster images. If the object related to the supplied handle has been created with <code>PDF_begin_template_ext()</code> , the string <code>template</code> will be returned.
<code>imagewidth</code>	Images: width in pixels Templates: user-supplied width
<code>infomode</code>	1 if the image has been loaded with the <code>infomode</code> option, 0 otherwise
<code>mirroringx, mirroringy</code>	Horizontal or vertical mirroring of the image (expressed as 1 or -1) according to the supplied options
<code>orientation</code>	Orientation value of the image. For images containing an orientation tag the value of this tag is returned; in all other cases 1 is returned. PDFlib automatically compensates orientation values different from 1.
<code>resx, resy</code>	Horizontal or vertical resolution of the image. Positive values represent the image resolution in pixels per inch (dpi). The value zero means that the resolution is unknown. Negative values can be used together to determine the aspect ratio of non-square pixels, but don't have any absolute meaning.
<code>strips</code>	Number of image strips (can be different from 1 only for certain multi-strip TIFF images)
<code>transparent</code>	1 if the image contains transparency (alpha channel with > 1 bit), otherwise 0. Transparency is assumed to be present if an alpha channel was read from the original image file, or the image has been loaded with the mask option.
<code>xid</code>	(Only for PDF/VT) String index for the GTS_XID entry of the image or template, or -1 if no GTS_XID value has been assigned. The GTS_XID string can be used in the CIP4/Summary/Content/Referenced metadata property for DPM.

Table 9.5 Option for `PDF_info_image()`

option	description
<code>useembedded-image</code>	(Boolean; only relevant if <code>templateoptions</code> is supplied) If true the information of the image embedded in the template is queried, otherwise the information of the template. If <code>useembeddedimage=true</code> some keywords are relevant only for the original image, but not for the generated template. In particular, the values should not be used for fitting the template created for the image. Default: false

9.2 SVG Graphics

C++ Java C# *int load_graphics(String type, String filename, String optlist)*

Perl PHP *int load_graphics(string type, string filename, string optlist)*

C *int PDF_load_graphics(PDF *p, const char *type, const char *filename, int len, const char *optlist)*

Open a disk-based or virtual vector graphics file subject to various options.

type Type of vector graphics file. The keyword *auto* automatically determines the file type. It is equivalent to *svg* which specifies SVG graphics.

filename (Name string; will be interpreted according to the global *filenamehandling* global option, see Table 2.1) Name of the graphics file to be opened. This must be the name of a disk-based or virtual file; PDFlib will not pull graphics from URLs.

If a file with the specified file name cannot be found PDFlib will try to determine the appropriate file name suffix automatically; it will append all suffixes from the following list (in both lowercase and uppercase) to the specified *filename* and try to locate a file with that name in the directories specified in the searchpath:

.svg, .svgz

len (C language binding only) Length of *filename* (in bytes). If *len = 0* a null-terminated string must be provided.

optlist An option list specifying graphics-related properties. The following options can be used:

- ▶ General option: *errorpolicy* (see Table 1.5)
- ▶ Font-related options according to Table 9.6: *defaultfontfamily, defaultfontoptions, fallbackfontfamily, fallbackfontoptions*
- ▶ Size options according to Table 9.6: *bboxexpand, fallbackheight, fallbackwidth, forcedheight, forcedwidth*
- ▶ Image-related options according to Table 9.6: *defaultimageoptions, fallbackimage*
- ▶ Other SVG processing options according to Table 9.6: *downloadlifetime, errorconditions, externalrefs, lang*
- ▶ Color control options according to Table 9.6: *devicencolors, forcesrgb, honoriccprofile, iccprofilecmyk, iccprofilegray, iccprofilergb*
- ▶ Template-related options according to Table 9.6: *inline, templateoptions*
- ▶ Encoding option for C and Perl, PHP, Ruby with *stringformat=legacy: hypertextencoding* (see Table 2.1)

Returns A graphics handle which can be used in subsequent graphics-related calls. If *errorpolicy=return* the caller must check for a return value of -1 (in PHP: 0) since it signals an error. If the method was called in *object* scope, the graphics handle can be reused across multiple PDF documents. If the method call fails you can request the reason of the failure with *PDF.get_errmsg()*.

If this method is called in *object* scope the graphics handle can be reused across multiple PDF documents. Otherwise the graphics handle is closed automatically at the end of the current document if necessary.

Details This method opens and analyzes a vector graphics file in one of the supported formats as determined by the *type* parameter. The graphics data will be stored in memory until

the graphics is closed with `PDF_close_graphics()` or at the end of the PDFlib object's lifetime. This method does not have any visible effect on the PDF output. In order to actually place the imported graphics somewhere in the generated document, `PDF_fit_graphics()` must be used. Opening the same graphics more than once per generated document is not recommended because the graphics data will be copied to the output document multiply.

PDFlib opens the graphics file with the provided *filename*, processes the contents, and by default closes the file before returning from this call. However, if the *template-options* option has been supplied, the file contents are required until the first call to `PDF_fit_graphics()` with the returned graphics handle.

Font embedding (especially relevant for PDF/A, PDF/X, and PDF/UA): font files for all fonts used in the graphics (or suitable default fonts) must be configured.

- PDF/A** All fonts must be embedded. If colors are specified as *device-gray*, *device-rgb*, or *device-cmyk* these are subject to the PDF/A color space requirements (see `PDF_set_color()`).
PDF/A-1: graphics with transparency are not allowed.
PDF/A-2a/3a: if the graphics contain text with PUA characters, the *tag* option with the *ActualText* suboption must be provided.
- PDF/UA** All fonts must be embedded (see above). If the graphics contain text with PUA characters, the *tag* option with the *ActualText* suboption must be provided.
- PDF/VT** This call may fail if the *usestransparency=false* option was specified in `PDF_begin_document()`, but the imported graphics contains transparency nevertheless.
- PDF/X** All fonts must be embedded. If colors are specified as *device-gray*, *device-rgb*, or *device-cmyk* these are subject to the PDF/X color space requirements (see `PDF_set_color()`).
PDF/X-3: graphics with transparency are not allowed.

Scope any

Table 9.6 Options for `PDF_load_graphics()`

option	description
<i>bbboxexpand</i>	(List of two floats or keywords; affects only SVG graphics without absolute size) The calculated bounding box may be too small because line widths and oversized glyphs are not taken into account. This option can be used to expand the calculated bounding box by numerical values (in default coordinates) in x and y direction on both sides. As an alternative the following keyword is supported (default: {offset offset}): <i>offset</i> Use the horizontal or vertical distance of the bounding box from the origin of the SVG coordinate system.
<i>defaultfont-family</i>	(Name string) Name of the font family which is used if a font for some text in the graphics file is either not specified or not available. Default: Arial Unicode MS if available, otherwise Helvetica
<i>defaultfont-options</i>	(Option list) Font loading options according to Table 4.1. When a font is required for text in a graphics file and this font has not already been loaded earlier, the options specified here are supplied to <code>PDF_load_font()</code> . Default: empty list (the option <i>keepfont</i> is added in all cases if the method is called in object scope)
<i>defaultimage-options</i>	(Option list) Image loading options according to Table 9.1. When an embedded or external image is processed, the options specified here are supplied to <code>PDF_load_image()</code> . Default: { }
<i>devicencolors</i>	(List of DeviceN color handles) Handles to DeviceN color spaces for use in the SVG graphics. The specified DeviceN color spaces are applied to device-nchannel colors in the SVG graphics according to the number of colorants.

Table 9.6 Options for `PDF_load_graphics()`

option	description
download-lifetime	(Keyword) Lifetime of automatically created PVF files containing downloaded network resources (default: <code>graphics</code>): graphics PVF files are released in <code>PDF_close_graphics()</code> with the exception of font resources which are released in <code>PDF_end_document()</code> . document PVF files are released in <code>PDF_end_document()</code> . If the graphics file is loaded in object scope, this keyword is mapped to the keyword <code>graphics</code> . object PVF files are released at the end of the enclosing object scope.
error-conditions	(Option list) List of conditions which trigger an error (default: empty): attributes (List of strings) An error occurs if one of the specified SVG attributes is present, but is not supported in PDFlib (see PDFlib Tutorial for a list of unsupported attributes). By default, unsupported attributes are ignored. elements (List of strings) An error occurs if one of the specified SVG elements is present, but is not supported in PDFlib (see PDFlib Tutorial for a list of unsupported elements). By default, unsupported elements are ignored. references (List of keywords) An error occurs if one of the following types of reference cannot be resolved or executed (by default all types except <code>image</code> are silently ignored and a warning is emitted): image reference to an image or graphics file; see option <code>fallbackimage</code> for default behavior internal internal reference to an SVG element external reference to files other than image or graphics fontfamily reference to a font family font reference to a full font name specified via font family, weight and style
externalrefs	(Boolean; only relevant if networking is enabled via the network option list) If false, URL references to external resources are not resolved, i.e. no network activity is initiated. This option affects only resource references with a leading protocol identifier. If an external reference is encountered in the SVG, but <code>externalrefs=false</code> the behavior is controlled by the option <code>errorconditions</code> . Default: true
fallback-fontfamily	(Name string) Name of the font family which is used to create a fallback font for each font, in addition to the fallback fonts which may have been specified in the graphics file. Default: empty
fallback-fontoptions	(Option list) Options which will be applied to the fallback fonts created via the <code>fallbackfontfamily</code> option. The following options according to Table 4.2 can be used: <code>fontsize</code> , <code>forcechars</code> , <code>textrise</code> . Default: empty
fallbackheight, fallbackwidth	(Float; affects only SVG graphics without absolute size; ignored if <code>forcedheight/forcedwidth</code> is supplied) Height/width of the SVG graphics (in default coordinates) for the fitting process. If one of the values is zero PDFlib calculates the bounding box. Default: values in the SVG graphics if present, otherwise 0

Table 9.6 Options for `PDF_load_graphics()`

option	description
fallback-image	(Option list) Specify how to visualize the space reserved for the fitbox of a graphics or image element if the element is not available. If this option is supplied without any suboptions a gray semi-transparent checkerboard is painted (default: no fallback display)
fillcolor	(sRGB color or keyword) Color used to fill the area with a checkerboard pattern (if <code>gridsize > 0</code>) where half of the squares are painted with the specified color and the other half of the squares is transparent, or color used to fill the area with solid color if <code>gridsize=0</code> . The keyword <code>none</code> means that the area will not be filled. Default: <code>LightGrey</code>
gridsize	(Float or percentage) Width of a square in the checkerboard pattern in default coordinates or as a percentage of the width of the fitbox. Percentages are rounded so that an integer number of squares fits into the area. <code>gridsize=0</code> means that the area is filled with solid color instead of a checkerboard pattern. Default: <code>10</code>
image	(Image or template handle) Image or template which will be placed with <code>fitmethod=entire</code> into the fitbox. Default: no image or template
opacity	(Float in the range 0..1 or percentage) Opacity of the checkerboard squares or the interior of the area. Default: <code>0.5</code>
strokecolor	(sRGB color or keyword) Color used to stroke the border area and a cross inside the area. The keyword <code>none</code> means that the border and cross is not stroked. Default: <code>Red</code>
forcedheight forcedwidth	(Float ≥ 1) The height/width of the SVG graphics are ignored and the specified values (in default coordinates) are applied instead. Default: height/width attributes of the graphics
forcesrgb	(Boolean) If true, all non-sRGB color specifications are ignored and the sRGB fallback color is used instead. <code>honoriccprofile=false</code> . This option does not affect referenced images. Default: <code>false</code>
honor-iccprofile	(Boolean) If true, honor all ICC profiles in <code>icc-color</code> specifications or implicitly in sRGB colors; otherwise all explicit ICC colors and implicit sRGB colors are interpreted as device-gray/device-rgb/device-cmyk colors. In PDF/A and PDF/X-3/4/5 the option <code>honoriccprofile=false</code> requires the <code>defaultrgb</code> option, an RGB output intent or the <code>iccprofilergb</code> option. Default: <code>true</code>
iccprofilecmyk, iccprofilegray, iccprofilergb	(ICC handle or keyword <code>srgb</code> ; overrides <code>forcesrgb</code> for <code>icc-color</code> (<code>gray/rgb/cmyk</code>); not allowed in object scope) ICC profiles to be used instead of the explicit profiles in <code>icc-color</code> specifications or the implicit sRGB profile. Default: ICC profiles in <code>icc-color</code> specifications and sRGB for default color specifications
inline	(Boolean; ignored if <code>templateoptions</code> is supplied) If true, create output inline in the content stream; otherwise a template (Form XObject) is created. The inline method suffers from disadvantages related to transparency handling and file size (see PDFlib Tutorial for details). On the other hand, this option is required if active links in the graphics should be converted (option <code>convertlinks</code> of <code>PDF_fit_graphics()</code>). Default: <code>false</code>
lang	(String) Natural language for the graphics file which can be used e.g. in an SVG switch element. The format of the language specification is identical to the <code>lang</code> option of <code>PDF_begin_document()</code> (see Table 3.3). Default: the language identifier found in the LANG environment variable.
template-options	(Option list) Create a template (Form XObject) according to the supplied option list. The supplied option list (which may be empty) is used for <code>PDF_begin_template_ext()</code> . The following common XObject options can be used (see Table 9.11): <code>associatedfiles</code> , <code>defaultgray</code> , <code>defaultrgb</code> , <code>defaultcmyk</code> , <code>iconname</code> , <code>layer</code> , <code>metadata</code> , <code>pdfvt</code> , <code>transparencygroup</code> Default: <code>templateoptions={ }</code> , i.e. a template with default options is created

C++ Java C# **void close_graphics(int graphics)**

Perl PHP **close_graphics(int graphics)**

C **void PDF_close_graphics(PDF *p, int graphics)**

Close vector graphics.

graphics A valid graphics handle retrieved with [PDF_load_graphics\(\)](#).

Details Delete the internal structure associated with the graphics. If the graphic has been opened from file, the actual graphics file is not affected by this call since it has already been closed at the end of the corresponding [PDF_load_graphics\(\)](#) call. A graphics handle cannot be used any more after it has been closed.

Scope any; *object* scope is not allowed if *templateoptions* was specified in the corresponding call to [PDF_load_graphics\(\)](#) and the graphics was placed at least once; must always be paired with a matching call to [PDF_load_graphics\(\)](#).

C++ Java C# **void fit_graphics(int graphics, double x, double y, String optlist)**

Perl PHP **fit_graphics(int graphics, float x, float y, string optlist)**

C **void PDF_fit_graphics(PDF *p, int graphics, double x, double y, const char *optlist)**

Place vector graphics on a content stream, subject to various options.

graphics A valid graphics handle retrieved with [PDF_load_graphics\(\)](#).

x, y The coordinates of the reference point in the user coordinate system where the graphic will be placed.

optlist An option list specifying graphics fitting and processing options. The following options are supported:

- ▶ Fitting options according to Table 6.1:
boxsize, fitmethod, matchbox, orientate, position, retpoint, rotate, scale, showborder
- ▶ Options for graphics processing according to Table 9.3:
adjustpage, gstate
- ▶ Option for processing interactive links in the graphics according to Table 9.7:
convertlinks
- ▶ Option for abbreviated structure element tagging according to Table 14.2 (only allowed in *page* scope): *tag*

Details The graphics will be placed relative to the reference point (*x, y*). By default, the lower left corner of the object will be placed at the reference point. However, the *orientate*, *boxsize*, *position*, and *fitmethod* options can modify this behavior. By default, a graphic will be scaled according to its internally specified size. This behavior can be modified with the *scale* and *fitmethod* options.

It is recommended to use [PDF_info_graphics\(\)](#) with the *fittingpossible* keyword before calling [PDF_fit_graphics\(\)](#) to check whether [PDF_fit_graphics\(\)](#) will succeed (and avoid an exception in case of a failure).

PDF/UA Graphics containing vector graphics or raster images must be tagged as *Figure* or *Artifact*.

Scope *page, pattern* (only if the pattern's *painttype* is 1), *template, glyph* (only if the glyph's *colorized* option is *true*); this method can be called an arbitrary number of times on arbitrary pages as long as the graphics handle has not been closed with `PDF_close_graphics()`.

Table 9.7 Additional option for `PDF_fit_graphics()`

option	description
convertlinks	(Boolean; only relevant for graphics loaded with the <code>inline</code> option) If <code>true</code> , interactive links in the graphics file are converted to interactive Link annotations in PDF. Regardless of this setting links are not created in the following situations (default: <code>true</code>): <ul style="list-style-type: none"> ▶ this method is called in a scope other than <code>page</code> ▶ the <code>templateoptions</code> option was supplied in <code>PDF_load_graphics()</code> ▶ Tagged PDF mode: the currently active structure item is an Artifact; regular Artifacts in PDF 2.0 (i.e. Artifacts which have been created with <code>direct=false</code>) are not affected by this restriction. ▶ PDF/X: the link annotation is located inside the <code>BleedBox</code> (or <code>TrimBox/ArtBox</code> if no <code>BleedBox</code> is present).

C++ Java C# **double** `info_graphics(int graphics, String keyword, String optlist)`

Perl PHP **float** `info_graphics(int graphics, string keyword, string optlist)`

C **double** `PDF_info_graphics(PDF *p, int graphics, const char *keyword, const char *optlist)`

Format vector graphics and query metrics and other properties.

graphics A valid graphics or template handle retrieved with `PDF_load_graphics()`.

keyword A keyword specifying the requested information:

- ▶ Keywords for querying the results of object fitting according to Table 6.3: *boundingbox, fitscalex, fitscaley, height, objectheight, objectwidth, width, x1, y1, x2, y2, x3, y3, x4, y4*
- ▶ Additional keywords according to Table 9.8: *description, filename, fittingpossible, graphicswidth, graphicsheight, istemplate, metadata, title, type, xid*

optlist An option list specifying options for `PDF_fit_graphics()`. Options which are not relevant for determining the value of the requested keyword are ignored.

Returns The value of some graphics property as requested by *keyword*. If a geometrical property is requested outside of a page, this method returns -1 (in PHP: 0). If an object handle is requested this method returns a handle to the object, or -1 (in PHP: 0) if the object is not available. If the requested keyword produces text, a string index is returned, and the corresponding string must be retrieved with `PDF_get_string()`.

Details This method performs all calculations required for placing the graphics according to the supplied options, but will not actually create any output on the page. The graphics reference point is assumed to be `{0 0}`.

Scope any

Table 9.8 Keywords for `PDF_info_graphics()`

keyword	description
description	String index for the contents of the desc element of the outermost svg element if present, or of the outermost g element if present, or -1 otherwise. The string may contain markup.
filename	String index for the name of the graphics file (including a searchpath directory if applicable)
fittingpossible	<p>Check whether the graphics can be placed with <code>PDF_fit_graphics()</code> in the current context. The value 1 is returned if the graphics can be placed. The value 0 is returned if fitting would fail (i.e. <code>PDF_fit_graphics()</code> would throw an exception) for one of the following reasons:</p> <ul style="list-style-type: none"> ▶ An internal problem in the graphics file. ▶ A conflict with current standards requirements (e.g. transparency not allowed, font must be embedded but no font file available) <p>If 0 is returned the nature of the problem can be queried with <code>PDF_get_errmsg()</code>. Since the result is valid only for the current context this check should be applied immediately before attempting to place the graphics.</p>
graphicswidth, graphicsheight	Width and height of the graphic in the default coordinate system according to the information in the graphics file. If no values are available in the graphics file, 0 will be returned.
istemplate	1 if the <code>templateoptions</code> option has been supplied, 0 otherwise
metadata	String index for the contents of the metadata element of the outermost svg element, or -1 if this element is not present. The string may contain markup.
title	String index for the contents of the title element of the outermost svg element if present, or of the outermost g element if present, or -1 otherwise. The string may contain markup.
type	String index for the type (format) of the graphics: always svg
xid	(Only for PDF/VT) String index for the GTS_XID entry of the template created for graphic, or -1 if no template was created or no GTS_XID value has been assigned to the template. The GTS_XID string can be used in the CIP4/Summary/Content/Referenced metadata property for DPM.

9.3 Templates (Form XObjects)

Note We use the term »template« as a synonym for PDF Form XObjects. The template methods described in this section are unrelated to variable data processing with PDFlib Blocks. Use `PDF_fill_block()` to fill Blocks prepared with the PDFlib Block Plugin (see Chapter 11, »Block Filling Methods (PPS)«, page 211).

C++ Java C# `int begin_template_ext(double width, double height, String optlist)`

Perl PHP `int begin_template_ext(float width, float height, string optlist)`

C `int PDF_begin_template_ext(PDF *p, double width, double height, const char *optlist)`

Start a template definition.

width, height The dimensions of the template's bounding box in points. The *width* and *height* parameters can be 0. In this case they must be supplied in `PDF_end_template_ext()` or with the *boundingbox* option. Ultimately both values must be different from 0 unless the *watermark* option is specified.

If the *width* or *height* option has been specified, the corresponding *width* or *height* API parameter is ignored.

optlist Option list specifying template-related properties.

- ▶ Common XObject options according to Table 9.11:
associatedfiles, defaultcmyk, defaultgray, defaultrgb, iconname, layer, metadata, pdfvt, transform, transparencygroup
- ▶ Template options according to Table 9.9:
boundingbox, height, topdown, watermark, width

Returns A template handle which can be used in subsequent calls to `PDF_fit_image()`, `PDF_info_image()`, and various options of other *methods*, or -1 (in PHP: 0) in case of an error.

Details This method resets all text, graphics, and color state parameters to their default values, and establishes a coordinate system according to the *topdown* option.

Template size: in the simplest case width and height are supplied in `PDF_begin_template_ext()`. However, if they are not yet known they can also be specified as zero. In this case they must be supplied in the corresponding call to `PDF_end_template_ext()`.

Editable watermarks. If the deprecated *watermark* option is supplied, the template defines a watermark which can be edited or removed in Acrobat. In this case only one of the following content placement methods can be used, as well as methods which are allowed in *document* or *object* scope:

- ▶ `PDF_fit_textline()`: The bounding box of the Textline defines the width and height of the template; the *width* and *height* parameters and the options *boundingbox* and *transform* are ignored. The bounding box of a watermark's Textline is calculated from its rotation angle and size, especially the option *boxheight*. A Textline used as watermark has a default *boxheight* of {*ascender descender*} instead of {*capheight 0*} (see Table 6.4, page 132). The parameters *x* and *y* of `PDF_fit_textline()` are ignored.
- ▶ `PDF_fit_textflow()`: The supplied fitbox defines the width and height of the template; the position of the fitbox is not relevant. The default value of the option *firstlinedist* is *ascender*; the default value of *lastlinedist* is *descender*. The first occurrences of the

Textflow options *fillcolor*, *font* (or *fontname*), *fontsize*, and *underline* determine the corresponding settings in Acrobat's Watermark menu.

- ▶ *PDF_fit_image()*
- ▶ *PDF_fit_pdi_page()*

The watermark template is automatically placed on all pages specified by the suboptions *startpage/endpage/pagesubset*. In Tagged PDF mode it is marked as Artifact. The following rules must be obeyed for creating and using the watermark template:

- ▶ The watermark template must be created before the first page where it is used.
- ▶ The *watermark* option can be used only once per document.
- ▶ The watermark template's handle must not be used in any call to *PDF_fit_image()* or any option list with an image handle (e.g. in *PDF_add_table_cell()*) since the watermark is placed automatically on the page range specified via the options *startpage* and *endpage*.

Only the following color spaces can be used for the watermark contents: *gray*, *iccbasedgray*, *iccbased* with a grayscale or RGB profile, *rgb*, *iccbasedrgb*

Scope any except *object*; this method starts *template* scope, and must always be paired with a matching *PDF_end_template_ext()* call.

Table 9.9 Option for *PDF_begin_template_ext()*

option	description
boundingbox	(Rectangle) Coordinates of the left, bottom, right, and top edges of the bounding box of the pattern cell or template. The bounding box can be used to clip the pattern cell or template, or to create white space around the pattern cell or template. Default: {0 0 width height}
height	(Float; ignored if boundingbox is supplied) The dimension of the template's bounding box in points. The width and height API parameters and options can be 0 in <i>PDF_begin_template_ext()</i> , but in this case they must be supplied in <i>PDF_end_template_ext()</i> or with the boundingbox option. Ultimately both values must be different from 0 unless the watermark option is specified. The width and height options override the API parameters with the same name. The following symbolic ISO page size names can be used as keywords by appending .width or .height (e.g. a4.width, a4.height): a0, a1, a2, a3, a4, a5, a6, b5, letter, legal, ledger, 11x17
topdown	(Boolean) If true, the origin of the coordinate system at the beginning of the page is located in the top left corner of the pattern cell or template, and y coordinates increase downwards; otherwise the default coordinate system is used. If topdown=true, the template height cannot be postponed to <i>PDF_end_template_ext()</i> , but must be supplied as parameter height or option height to <i>PDF_begin_template_ext()</i> . Default: false
watermark	(Option list; can only be used once per document; deprecated; not allowed in PDF 2.0, PDF/A, PDF/UA, PDF/X and PDF/VT) Create an editable watermark according to the options in Table 9.10.
width	(Float) See height.

Table 9.10 Suboptions for the deprecated watermark option of *PDF_begin_template_ext()*

option	description
endpage	(Integer or keyword) Number of the last page with a watermark, or the keyword last for the final page. Default: last
fixedprint	(Boolean) Keep position and size of the watermark constant when printing on different page sizes. Default: false

Table 9.10 Suboptions for the deprecated watermark option of `PDF_begin_template_ext()`

option	description
horizontalign	(Keyword) One of keywords <code>left</code> / <code>center</code> / <code>right</code> to specify the relative horizontal position of the watermark's bounding box on the page ¹ . Default: <code>center</code>
horizshift	(Float) Shift from the relative horizontal position in points. Default: <code>0</code>
location	(Keyword) Specify whether the watermark appears behind or on top of other page contents (default: <code>ontop</code>): behind The watermark appears behind the page contents. If <code>location=behind</code> is supplied all future calls to <code>PDF_begin_page_ext()</code> in the same document must supply a value different from <code>0</code> for the <code>width</code> and <code>height</code> parameters. ontop The watermark appears on top of the page contents.
onprint	(Boolean; PDF 1.5; the value <code>false</code> is not allowed in PDF/A-2/3, PDF/X-4/5, and PDF/UA-1) If <code>true</code> , the watermark is shown when printing the page. Default: <code>true</code>
onscreen	(Boolean; PDF 1.5; the value <code>false</code> is not allowed in PDF/A-2/3, PDF/X-4/5, and PDF/UA-1) If <code>true</code> , the watermark is shown when displaying the page on screen. Default: <code>true</code>
opacity	(Float or percentage; must have the value <code>1</code> in PDF/A-1 and PDF/X-3) Opacity value for painting the watermark on the page. Default: <code>1</code>
pagesubset	(Keyword) Select a subset of the pages in the range specified by <code>startpage</code> and <code>endpage</code> (default: <code>all</code>): all Select all pages in the range. even Select all pages in the range with an even number. odd Select all pages in the range with an odd number.
scale	(Float, percentage, or keyword) Scaling factor of the watermark's bounding box relative to the target page. Scaling preserves the aspect ratio of the Textline's bounding box (default: <code>1</code>). Supported keyword: none No scaling
startpage	(Integer) Number of the first page with a watermark (the first page has number <code>1</code>). If the next page, i.e. the page created with the next call to <code>PDF_begin_page_ext()</code> , has a higher number than the one specified in <code>startpage</code> , the value is increased to the next page. Default: the next page
verticalign	(Keyword) One of keywords <code>bottom</code> / <code>center</code> / <code>top</code> to specify the relative vertical position of the watermark's bounding box on the page. Default: <code>center</code>
vertshift	(Float) Shift from the relative vertical position in points. Default: <code>0</code>

¹ The page size is defined by the `CropBox` if present, otherwise the `MediaBox`.

C++ Java C# **void end_template_ext(double width, double height)**

Perl PHP **end_template_ext(float width, float height)**

C **void PDF_end_template_ext(PDF *p, double width, double height)**

Finish a template definition.

width, height The dimensions of the template's bounding box in points. If `width` or `height` is `0`, the value supplied in `PDF_begin_template_ext()` are used. Otherwise the values supplied in the `boundingbox` option and the `width` and `height` parameters of `PDF_begin_template_ext()` are overridden. However, if the `watermark` option has been specified in the corresponding call to `PDF_begin_template_ext()`, the values supplied to `PDF_end_template_ext()` are ignored.

Scope *template*; this method terminates *template* scope, and must always be paired with a matching *PDF_begin_template_ext()* call.

9.4 Common XObject Options

The options listed in this section apply to the following methods which create Form XObjects:

- ▶ `PDF_load_image()` with the `templateoptions` option
- ▶ `PDF_open_pdi_page()`
- ▶ `PDF_load_graphics()` (unless the `inline` option has been supplied)
- ▶ `PDF_begin_template_ext()`

The following XObject options according to Table 9.11 are available (not all options apply to all methods above):

`associatedfiles`, `defaultcmyk`, `defaultgray`, `defaultrgb`, `georeference`, `iconname`, `layer`, `metadata`, `pdfvt`, `transform`, `transparencygroup`

PDF/A Some options are restricted.

PDF/VT The `pdfvt` option is relevant for PDF/VT. The generated XObject is marked as encapsulated if certain conditions are met (see PDFlib Tutorial for details).

PDF/X Some options are restricted.

Table 9.11 Common XObject options for `PDF_load_image()`, `PDF_open_pdi_page()`, `PDF_load_graphics()` and `PDF_begin_template_ext()`

option	description
associatedfiles	(List of asset handles; only for PDF 2.0 and PDF/A-3) Asset handles for files associated with the XObject. The files must have been loaded with <code>PDF_load_asset()</code> and <code>type=attachment</code> .
defaultgray defaultrgb defaultcmyk	(ICC handle or keyword; in <code>PDF_load_image()</code> only with <code>templateoptions</code> ; not for <code>PDF_open_pdi_page()</code>) Set a default gray, RGB, or CMYK color space for the template according to the supplied ICC profile handle. The option <code>defaultrgb</code> also supports the keyword <code>srgb</code> . The specified color space is used to map device-dependent gray, RGB, or CMYK colors on the template (but not on nested templates).
georeference	(Option list; PDF 1.7ext3; only for <code>PDF_load_image()</code> without <code>templateoptions</code>) Description of an earth-based coordinate system associated with the XObject to use for geospatial measuring; see Section 13.4, »Geospatial Features«, page 260, for details.
iconname	(Hypertext string; in <code>PDF_load_image()</code> only with <code>templateoptions</code>) Attach a name to a Form XObject so that it can be referenced via JavaScript, e.g. to use the XObject as an icon for form fields.
layer	(Layer handle; PDF 1.5) Layer to which the XObject will belong unless another layer has been activated with <code>PDF_begin_layer()</code> prior to placing the object. Calling <code>PDF_begin_layer()</code> to activate a layer before placing the XObject overrides the object's <code>layer</code> option. Call <code>PDF_end_layer()</code> before placing the object to make sure that the object's <code>layer</code> option will not be overridden.
metadata	(Option list) Metadata for the XObject (see Section 15.2, »XMP Metadata«, page 277).
pdfvt	(Option list; only for PDF/VT) PDF/VT suboptions for the XObject according to Table 9.13

Table 9.11 Common XObject options for `PDF_load_image()`, `PDF_open_pdi_page()`, `PDF_load_graphics()` and `PDF_begin_template_ext()`

option	description
transform	<p>(Transformation list according to Table 9.12; ignored if the <code>cloneboxes</code> option of <code>PDF_open_pdi_page()</code> has been supplied for the page; not for <code>PDF_load_image()</code>) A list which defines the transformation that maps the page, pattern, or template coordinate system to the default coordinate system of the target page, template or glyph description where the page, pattern or template is used. The concatenation of the page, pattern or template matrix with that of the target page, template or glyph description establishes the coordinate system within which all graphics objects in the page, pattern or template are interpreted.</p> <p>The list contains pairs of a keyword and a float list according to Table 9.12 where each pair defines a transformation. The interpretation and length of the float list depends on the transformation. Transformations are applied in the specified order. The elements within a pair may be separated with equals signs '='. Default: no transformation.</p> <p>Example: <code>transform={rotate=45 translate={100 0}}</code></p>
transparency group	<p>(Option list or keyword; not for PDF/A-1 and PDF/X-3; <code>PDF_open_pdi_page()</code> and <code>PDF_load_graphics()</code> enforce auto in PDF/A-2/3 and PDF/X-4/5 modes) Attach a transparency group to the Form XObject. The following keywords are supported (default: auto):</p> <p>auto <code>PDF_open_pdi_page()</code> and <code>PDF_load_graphics()</code>: unless the document option <code>usestransparency=false</code> is set, a transparency group is created which ensures that the placed contents are rendered the same way as before placing them on the output page. <code>PDF_load_image()</code> with <code>templateoptions</code> and <code>PDF_begin_template_ext()</code>: same as none</p> <p>none Don't create a transparency group for the Form XObject.</p> <p>The following suboptions can be used to explicitly create a transparency group:</p> <p>colorspace (Keyword or ICC profile handle; required with a value different from none if the template is used for the <code>softmask</code> option of <code>PDF_create_gstate()</code> with <code>type=luminosity</code>) Blending color space; see Table 3.9 (option <code>transparencygroup</code>) for supported keywords and restrictions (default: none).</p> <p>isolated (Boolean) Specifies whether the transparency group is isolated. All objects in an isolated group are composited against a transparent initial background and the result is composited against the existing page contents. In effect, compositing the group objects is independent of the background: the group is »isolated«. Objects in a non-isolated group are composited individually against each other and the existing page contents. Default: false</p> <p>knockout (Boolean) Specifies whether the transparency group is a knockout group. All objects in a knockout group are composited directly against the page background (or a transparent background if the group is isolated) instead of with the preceding elements in the group. In effect, there is no interaction among objects in the group: each object knocks out earlier overlapping objects in the group, and the topmost object is composited against the page background. Default: false</p>

Table 9.12 Keywords and float lists for the `transform` option of `PDF_begin_pattern_ext()`, `PDF_begin_template_ext()`, `PDF_shading_pattern()` and `PDF_open_pdi_page()`

keyword	description
align	Rotate by the direction vector {dx dy}. The vector {0 0} implies no rotation.
matrix	Specify a non-degenerate transformation matrix with six values {a b c d e f}. The keyword <code>current</code> creates six values comprising the current transformation matrix. This may be useful for creating a transformation which matches the current user coordinate system.

Table 9.12 Keywords and float lists for the transform option of `PDF_begin_pattern_ext()`, `PDF_begin_template_ext()`, `PDF_shading_pattern()` and `PDF_open_pdi_page()`

keyword	description
rotate	Rotate by {phi}, where the angle phi is measured in degrees counterclockwise from the positive x axis of the pattern, page or template coordinate system.
scale	Scale by {sx sy}. If sy is not provided it is assumed to be equal to sx.
skew	Skew (shear) by {alpha beta}, where alpha is measured in degrees counterclockwise from the positive x axis of the pattern coordinate system and beta is measured clockwise from the positive y axis. Both angles must not be odd multiples of 90°.
translate	Translate by {tx ty}. Since translations are compensated when placing the page, this translation doesn't have any visible effect for imported PDF pages.

Table 9.13 Suboptions for the pdfvt option in `PDF_load_image()`, `PDF_open_pdi_page()`, `PDF_begin_template_ext()` as well as `PDF_load_graphics()` with the templateoptions option

option	description
environment	(Hypertext string; required if scope=stream or scope=global) PDF/VT environment context, i.e. an identifier that can be used by a PDF/VT processor to provide a management interface for managing related XObjects. For example, the customer name or job name could be used to identify the environment.
scope	(Keyword) PDF/VT scope (not related to PDFlib method scope) of the XObject (default: unknown): <ul style="list-style-type: none"> unknown The scope of the XObject is not known. singleuse The XObject is referenced only once in the PDF/VT file. record (Only allowed if the recordlevel option for <code>PDF_begin_document()</code> has been specified) The XObject is referenced more than once in the pages belonging to a single record, but is not referenced within other records. file The XObject is referenced more than once in the PDF/VT file. If the recordlevel option has been supplied, scope=file should only be used if the XObject is used in more than one record (and scope=record otherwise). global (Requires the environment option) The XObject or an equivalent XObject is referenced in more than one PDF/VT file.
xid	(String; only for <code>PDF_begin_template_ext()</code> since identifiers are automatically created for all other types of XObjects) Unique identifier for the Form XObject created for the template. It is strongly recommended to provide this identifier in the format suggested in ISO 16612-2:2010, section 6.7.2, i.e. a URI with the uuid scheme and 128-bit number according to RFC 4122. The identifiers should be identical for template definitions which create equivalent PDF Form XObjects according to PDF/VT (i.e. templates which create the same visual output). Templates which are not equivalent must have different identifiers or no identifier at all. It is strongly recommended to supply the xid option for templates with scope=stream or scope=global to allow caching of Form XObjects across documents. Example xid in the recommended format: uuid:1228c416-48f2-e817-ad69-8206e41dca2d

10 PDF Import (PDI) and pCOS Methods

API methods in this chapter:

- ▶ `PDF_open_pdi_document()`
- ▶ `PDF_close_pdi_document()`
- ▶ `PDF_open_pdi_page()`
- ▶ `PDF_close_pdi_page()`
- ▶ `PDF_fit_pdi_page()`
- ▶ `PDF_info_pdi_page()`
- ▶ `PDF_process_pdi()`
- ▶ `PDF_pcos_get_number()`
- ▶ `PDF_pcos_get_string()`
- ▶ `PDF_pcos_get_stream()`

Note All methods described in this chapter require the PDF import library (PDI) which is included in PDFlib+PDI and PDFlib Personalization Server (PPS), but not in the base PDFlib product. Please visit our Web site for more information on obtaining PDI.

10.1 PDF Document Import Methods

C++ Java C# `int open_pdi_document(String filename, String optlist)`

Perl PHP `int open_pdi_document(string filename, string optlist)`

C `int PDF_open_pdi_document(PDF *p, const char *filename, int len, const char *optlist)`

Open a disk-based or virtual PDF document and prepare it for later use.

filename (Name string; will be interpreted according to the global `filenamehandling` option, see Table 2.1) Name of the PDF file.

optlist An option list specifying PDF open options:

- ▶ General option: `errorpolicy` (see Table 1.5)
- ▶ PDF document options according to Table 10.1:
`acceptdynamicxfa, checkoutintentprofile, infomode, inmemory, password pcosengines repair, requiredmode, shrug, useactions, usejavascript, userenditions`
- ▶ Tagged PDF processing options according to Table 10.1: `checktags, usetags`
- ▶ Layer processing options according to Table 10.1:
`parentlayer, parenttitle, uselayers`

len (C language binding only) Length of `filename` (in bytes). If `len = 0` a null-terminated string must be provided.

Returns A PDI document handle which can be used for processing individual pages of the document or for querying document properties. A return value of -1 (in PHP: 0) indicates that the PDF document couldn't be opened. An arbitrary number of PDF documents can be opened simultaneously. If the method call fails you can request the reason of the failure with `PDF_get_errmsg()`.

The error behavior can be changed with the `errorpolicy` option.

Details By default, the document is rejected if at least one of the following conditions is true:

- ▶ The document is damaged and couldn't be repaired (or *repair=none* was specified).
- ▶ The document is encrypted, but the corresponding master password has not been supplied in the *password* option. The *shrug* option can be used to enable page import from protected documents under certain conditions (see PDFlib Tutorial).

In the second case, the option *requiredmode=minimum* or *requiredmode=restricted* can be used to open the document nevertheless. This may be useful to query information about the PDF using the *PDF_pcos_get_**() methods, such as encryption, document info fields, etc.

In order to get more detailed information about the nature of a PDF import-related problem (wrong PDF file name, bad PDF data, etc.), use *PDF_get_errmsg*() to receive a more detailed error message.

Scope any

Table 10.1 Options for *PDF_open_pdi_document*()

option	description
accept-dynamicxfa	(Boolean) If true, dynamic XFA forms can successfully be opened. Querying pCOS paths is the only reasonable activity. Calling <i>PDF_open_pdi_page</i> () will fail since no pages can be imported. Default: false
checktags	(Keyword) Specifies whether the structure element nesting rules (see PDFlib Tutorial) are checked in <i>PDF_open_pdi_page</i> () for imported structure elements. Supported keywords (default: none): none Tag nesting rules are not enforced. This setting is default since many real-world documents violate structure element nesting rules, and couldn't otherwise be imported. relaxed Similar to strict except that a few rules are not enforced (see PDFlib Tutorial). strict If an imported tag violates the nesting rules the call to <i>PDF_open_pdi_page</i> () will fail.
checkoutprofile	(Boolean, only relevant for PDF/A and PDF/X) If true, the number of color components of an output intent is checked against the number of components in the associated ICC profile. This guards against inconsistent input documents. Setting this option to false reduces memory requirements, but should be used only if the input documents are known to contain consistent output intent profiles. Default: true
infomode	(Boolean) If true, the document will be opened such that information can be queried with the pCOS interface, but the pages can not be imported into the current output document with <i>PDF_open_pdi_page</i> (). The following documents can be opened when <i>infomode=true</i> : encrypted PDFs where the password is not known (exception: PDF 1.6 and above documents created with the Distiller setting »Object Level Compression: Maximum«). Default: false if <i>requiredmode=full</i> , otherwise true
inmemory	(Boolean) If true, PDI loads the complete file into memory and process it from there. This can result in a tremendous performance gain on some systems (especially z/OS) at the expense of memory usage. If false, individual parts of the document are read from disk as needed. Default: false
parentlayer	(Layer handle; ignored if the input document doesn't contain any layers or <i>uselayers=false</i>) Insert all layer definitions imported from the document as children of the specified layer. If the specified layer has been activated anywhere in the output document it is used as parent; otherwise it is used as title (separator) only. Default: no parent layer
parenttitle	(Hypertext string; ignored if the input document doesn't contain any layers or <i>uselayers=false</i>) Add a title layer which does not directly control the visibility of page contents, but serves as a hierarchical separator for the imported layer definitions. Default: no parent title
password	(String) Master password required to open a protected PDF document for import. If no password has been supplied at all for an encrypted document the document handle can only be used to query its encryption status. The <i>shrug</i> option can be used to import pages from protected documents under certain conditions (see PDFlib Tutorial).

Table 10.1 Options for `PDF_open_pdi_document()`

option	description
pcosengines	(Option list) Enable or disable pCOS engines for document-wide resource collection. Turning off an engine means that the corresponding per-document and per-page pseudo object arrays will be empty. Supported keywords (default: all pCOS engines are active): colorspace, extgstate, font, image, pattern, property, shading, template
repair	(Keyword) Specifies how to treat damaged PDF input documents. Repairing a document takes more time than normal parsing, but may allow processing of certain damaged PDFs. Note that some documents may be damaged beyond repair. Supported keywords (default: auto): auto Repair the document only if problems are detected while opening the PDF. force Unconditionally try to repair the document, regardless of whether or not it has problems. none No attempt will be made at repairing the document. If there are problems in the PDF the call will fail.
requiredmode	(Keyword) The lowest pCOS mode (minimum/restricted/full) which is acceptable when opening the document. The call fails if the resulting pCOS mode would be lower than the required mode. If the call succeeds it is guaranteed that the resulting pCOS mode is at least the one specified in this option. However, it may be higher; e.g. <code>requiredmode=minimum</code> for an unencrypted document results in full mode. Default: full
shrug	(Boolean) If true, the shrug feature is activated to enable page import from protected documents under certain conditions (see PDFlib Tutorial). By using the shrug option you assert that you honor the PDF document author's rights. Default: false
useactions	(Boolean; not for PDF/A) If true, all document actions of the imported document are copied to the generated output document(s). If actions for the same trigger events are imported from multiple PDI documents it is undefined which action is written to the output. The options <code>action</code> and <code>destination</code> of <code>PDF_begin/end_document()</code> have priority over actions imported from PDI documents. Default: false
usejavascript	(Boolean) Import all document Javascripts (not to be confused with JavaScript for document actions such as open or close). Such document-level JavaScript may be required for field or annotation actions such as calculation or validation scripts. It is therefore recommended to import JavaScript if pages are imported with form fields (option <code>usefields</code> in <code>PDF_fit_pdi_page()</code>). If the same script name is imported from multiple documents only the first script is used. Default: false
userenditions	(Boolean) If true, import all renditions in the document's name tree. Such renditions are accessible to JavaScript by name. Default: false
uselayers	(Boolean; only relevant if the input contains layers) If true, all layer definitions used on any of the imported pages are imported. This option affects only layer definitions, but not the actual layer contents since PDI always imports the contents of all layers on a page. In order to work with <code>uselayers=false</code> the generated document must not contain any layers at all, i.e. all PDF documents with layers must be opened with <code>uselayers=false</code> and <code>PDF_define_layer()</code> must not be called. Default: true
usetags	(Boolean; only relevant for tagged PDF input and output) If true, the structure hierarchy of the imported document is read so that structure element tags can later be imported along with the pages. Default: false in object scope, otherwise true

```
C int PDF_open_pdi_callback(PDF *p, void *opaque, pdf_off_t filesize,
    size_t (*readproc)(void *opaque, void *buffer, size_t size),
    int (*seekproc)(void *opaque, pdf_off_t offset),
    const char *optlist)
```

Open a PDF document from a custom data source and prepare it for later use.

opaque A pointer to some user data that might be associated with the input PDF document. This pointer will be passed as the first parameter of the callback methods, and can be used in any way. PDI will not use the opaque pointer in any other way.

filesize The size of the complete PDF document in bytes.

readproc A callback function which copies *size* bytes to the memory pointed to by *buffer*. If the end of the document is reached it may copy less data than requested. The function must return the number of bytes copied.

seekproc A callback function which sets the current read position in the document. *offset* denotes the position from the beginning of the document (0 meaning the first byte). If successful, this function must return 0, otherwise -1.

optlist An option list specifying PDF open options; all options of `PDF_open_pdi_document()` are supported.

Returns A PDI document handle which can be used for processing individual pages of the document or for querying document properties. A return value of -1 indicates that the PDF document couldn't be opened. An arbitrary number of PDF documents can be opened simultaneously. The return value can be used until the end of the enclosing *document* scope. If the method call fails you can request the reason of the failure with `PDF_get_errmsg()`.

Details This is a specialized interface for applications which retrieve arbitrary chunks of PDF data from some data source instead of providing the PDF document in a disk file or in memory. The `pdf_off_t` type is defined conditionally in `pdflib.h`. It usually holds 64-bit values as offset type for large files beyond 2GB. The application must be built with Large File Support (LFS).

Scope any; in *object* scope a PDI document handle can only be used to query information from a PDF document.

Bindings Only available in the C language binding.

C++ Java C# **void close_pdi_document(int doc)**

Perl PHP **close_pdi_document(int doc)**

C **void PDF_close_pdi_document(PDF *p, int doc)**

Close all open PDI page handles, and close the input PDF document.

doc A valid PDF document handle retrieved with `PDF_open_pdi_document()`.

Details This method closes a PDF import document, and releases all resources related to the document. All document pages which may be open are implicitly closed. The document handle must not be used after this call. A PDF document should not be closed if more

pages are to be imported. Although you can open and close a PDF import document an arbitrary number of times, doing so may result in unnecessary large PDF output files.

Scope any

10.2 PDF Page Import Methods

C++ Java C# `int open_pdi_page(int doc, int pagenumber, String optlist)`

Perl PHP `int open_pdi_page(int doc, int pagenumber, string optlist)`

C `int PDF_open_pdi_page(PDF *p, int doc, int pagenumber, const char* optlist)`

Prepare a page for later use with `PDF_fit_pdi_page()`.

doc A valid PDF document handle retrieved with `PDF_open_pdi_document()`.

pagenumber The number of the page to be opened. The first page has page number 1.

optlist An option list specifying page-specific options:

- ▶ General options: `errorpolicy` (see Table 1.5)
- ▶ PDF page options according to Table 10.2:
`boxexpand`, `checktransgroupprofile`, `clippingarea`, `cloneboxes`, `forcebox`, `ignorepdfversion`,
`initgraphicsstate`, `pdiusebox`, `usetags`
- ▶ Common XObject options (see Table 9.11):
`associatedfiles`, `iconname`, `layer`, `metadata`, `pdfvt`, `transform`, `transparencygroup`
- ▶ Encoding option for C and Perl, PHP, Ruby with `stringformat=legacy`:
`hypertextencoding` (see Table 2.1)

Returns A PDI page handle which can be used for placing pages with `PDF_fit_pdi_page()`. A return value of -1 (in PHP: 0) indicates that the page couldn't be opened. If the method call fails you can request the reason of the failure with `PDF_get_errmsg()`. The returned handle can be used until the end of the enclosing `document` scope.

The error behavior can be changed with the `errorpolicy` option.

Details This method copies all data comprising the imported page to the output document, but does not have any visible effect on the output. In order to actually place the imported page somewhere in the generated output document, `PDF_fit_pdi_page()` must be used. This method fails in the following cases:

- ▶ The document uses a PDF version which is incompatible to the current PDF document. For PDF versions up to PDF 1.6 all versions up to and including the same version are compatible. PDF 1.7, PDF 1.7ext3, PDF 1.7ext8 and PDF 2.0 are all compatible to each other as far as page import with PDI is concerned. However, in PDF/A mode the input PDF version number is ignored since PDF version headers must be ignored in PDF/A.
- ▶ The document is not compatible to the current PDF/A, PDF/X, PDF/VT or PDF/UA output conformance level, or uses an incompatible output intent.
- ▶ If the document contains an inconsistent PDF/A or PDF/X output intent no pages can be imported.

In order to get more detailed information about a problem related to PDF import (bad PDF data, etc.) you can call `PDF_get_errmsg()`.

An arbitrary number of pages can be opened simultaneously. If the same page is opened multiply, different handles will be returned, and each handle must be closed exactly once.

Tagged PDF 2.0 pages cannot be imported into PDF 1.7 with `usetags=true` if structure elements on the imported page use a namespace different from PDF 1.7.

PDF/A The imported document must be compatible to the current PDF/A output conformance level (see PDFlib Tutorial for details) and output intent.

PDF/UA Imported documents must conform to PDF/UA. Pages from untagged documents, non-PDF/UA documents or tagged documents loaded with `usetags=false` in `PDF_open_pdi_document()` can also be placed. However, such pages are automatically marked as Artifact. The currently active element must therefore allow Artifact as child in this case.

The role map of the imported document must be compatible with the mapping provided by the `rolemap` option of `PDF_begin_document()` (see PDFlib Tutorial for details). This means that custom structure types must not be mapped to different standard types by the `rolemap` option (or previously imported documents) and the role map of the imported document.

The heading structure of the imported page must be compatible with the structure type of the generated document, i.e. if `structuretype=weak` only `H1`, `H2`, etc. (but not `H`) must be used on the page; if `structuretype=strong` only `H` (but not `H1`, `H2`, etc.) must be used on the imported page. Pages with both numbered and unnumbered headings are rejected.

PDF/VT The imported document must conform to PDF/X-3/4/4p or PDF/VT-1 and must use the same output intent as the generated document. Document Part Metadata (DPM) in the imported document is not imported. This call may fail if the `usestransparency=false` option was specified in `PDF_begin_document()`, but the imported page contains transparency nevertheless.

The Form XObject created from the imported page is marked as encapsulated if certain conditions are met (see PDFlib Tutorial).

PDF/X The imported document must be compatible to the current PDF/X output conformance level (see PDFlib Tutorial for details), and must use the same output intent as the generated document.

PDF/X-4/5: the imported page is rejected if it uses a CMYK ICC profile which is identical to the generated document's output intent profile.

Scope any except `object`

Table 10.2 Options for `PDF_open_pdi_page()`

option	description
boxexpand	(Float or list with four floats) Expand the page box selected via the <code>pdiusebox</code> option on all four sides by the same amount (if one value is provided) or on the left/right/bottom/top sides individually (if four values are provided). Negative values are allowed to reduce the page size. This option may be used to place content which is located outside of all page boxes of the imported page, or to add margins. Default: 0
checktrans-groupprofile	(Boolean, only relevant for PDF/A and PDF/X) If <code>true</code> and the imported page contains a transparency group, its color space is checked for consistency and compatibility with the generated output document. This guards against inconsistent input documents and color space conflicts which could lead to non-conforming PDF/X or PDF/A output. Setting this option to <code>false</code> reduces memory requirements, but should be used only if the imported page is known to contain a conforming transparency group (if any). Default: <code>true</code>

Table 10.2 Options for `PDF_open_pdi_page()`

option	description
clippingarea	(Keyword) Specify which of the page boxes of the imported page is used for clipping. Page content outside the specified area are not visible after placing the imported page on a new page. Supported keywords (default: <code>pdiusebox</code>): art Use the ArtBox if present, else the CropBox bleed Use the BleedBox if present, else the CropBox crop Use the CropBox if present, else the MediaBox media Use the MediaBox (which is always present) pdiusebox If <code>cloneboxes</code> is specified use the MediaBox, otherwise the box specified in the <code>pdiusebox</code> option trim Use the TrimBox if present, else the CropBox
cloneboxes	(Boolean; not allowed if <code>boxexpand</code> , <code>forcebox</code> , or <code>pdiusebox</code> is supplied; must match the <code>cloneboxes</code> option in <code>PDF_fit_pdi_page()</code>) If <code>true</code> , the page is prepared for box cloning with the <code>cloneboxes</code> option of <code>PDF_fit_pdi_page()</code> . Default: <code>false</code>
forcebox	(Rectangle) Force the page box to the specified values. This option overrides the <code>pdiusebox</code> and <code>boxexpand</code> options. It may be used to place content which is located outside of all page boxes of the imported page. The values must be chosen carefully if the imported page contains a Rotate key. The <code>boxexpand</code> option is preferable since it works regardless of any Rotate key. Default: the box selected with the <code>pdiusebox</code> option
ignore-pdfversion	(Boolean) If <code>true</code> , the PDF version number of the input PDF document is ignored, i.e. pages from documents with a higher PDF version than the current PDF output document can be imported. This may be useful for PDF documents with a higher PDF version which are nevertheless fully compatible to the current PDF output level. The user is responsible for making sure that the imported pages do not violate the PDF output compatibility. Default: <code>false</code> in general, but <code>true</code> in PDF/A and PDF/X mode
initgraphics-state	(Boolean; forced to <code>true</code> in PDF/VT mode) If <code>true</code> , all graphics state parameters are initialized explicitly with the default values for the imported page. This prevents the current graphics state parameters from being applied to imported pages which rely on the defaults instead of explicitly setting all values. Default: <code>false</code>
pdiusebox	(Keyword; not allowed if <code>cloneboxes</code> is supplied) Specifies which box dimensions are used for determining an imported page's size. The box size will be used for scaling operations in <code>PDF_fit_pdi_page()</code> . This box also determines the visible contents of the page unless modified with the <code>clippingarea</code> option. Default: <code>crop</code> . art Use the ArtBox if present, else the CropBox bleed Use the BleedBox if present, else the CropBox crop Use the CropBox if present, else the MediaBox media Use the MediaBox (which is always present and must enclose all other boxes) trim Use the TrimBox if present, else the CropBox
usetags	(Boolean; only relevant for tagged PDF input and output and if the document has been opened with <code>usetags=true</code>) If <code>true</code> , the structure tags of the imported page are copied to the structure hierarchy of the output document. In this case <code>PDF_fit_pdi_page()</code> can only be called in page scope. Default: <code>true</code>

C++ Java C# **void close_pdi_page(int page)**

Perl PHP **close_pdi_page(int page)**

C **void PDF_close_pdi_page(PDF *p, int page)**

Close the page handle and free all page-related resources.

page A valid PDF page handle (not a page number!) retrieved with **PDF_open_pdi_page()**.

Details This method closes the page associated with the page handle identified by *page*, and releases all related resources. *page* must not be used after this call.

Scope any except *object*

C++ Java C# **void fit_pdi_page(int page, double x, double y, String optlist)**

Perl PHP **fit_pdi_page(int page, float x, float y, string optlist)**

C **void PDF_fit_pdi_page(PDF *p, int page, double x, double y, const char *optlist)**

Place an imported PDF page on the output page subject to various options.

page A valid PDF page handle (not a page number!) retrieved with **PDF_open_pdi_page()**. The page handle must not have been closed.

x, y The coordinates of the reference point in the user coordinate system where the page will be located, subject to various options.

optlist An option list specifying page options:

- ▶ Fitting options according to Table 6.1:
blind, boxsize, fitmethod, matchbox, orientate, position, rotate, scale, showborder
- ▶ Options for page processing according to Table 9.3: *adjustpage, gstate*
- ▶ Processing options according to Table 10.3: *cloneboxes, useactions, useannots, usefields*
- ▶ Option for abbreviated structure element tagging according to Table 14.2 (only in *page* scope): *tag*

Details This method is similar to **PDF_fit_image()**, but operates on imported PDF pages instead. A tagged page (i.e. tagged PDF is created and the page is imported with *usetags=true* from a tagged PDF) cannot be placed more than once.

In Tagged PDF mode it is recommended to use **PDF_info_pdi_page()** with the *fitting-possible* keyword before calling **PDF_fit_pdi_page()** to check whether **PDF_fit_pdi_page()** will succeed (and avoid an exception in case of a failure).

An imported page can be placed with form fields only once. If imported fields or annotations use scripts from the JavaScript name tree these scripts can be imported with the option *usejavascript* of **PDF_open_pdi_document()**.

If a page containing one of the annotation types *Caret*, *FreeText*, *Line* or *Stamp* is imported with *useannots* (enabled by default) or *usefields* and the annotation or field doesn't have an appearance stream, access to the Noto fonts must be configured. These fonts are included in the PDFlib distribution (see PDFlib Tutorial).

PDF/UA Restrictions apply to the *useannots* and *usefields* options.

PDF/X Options *useannots* and *usefields*: annotations are only imported if they are located completely outside of the BleedBox (or TrimBox/ArtBox if no BleedBox is present) of the generated page.

Scope *page, pattern, template, glyph*; however, if a page from a Tagged PDF document has been loaded with *usetags=true* this method can only be called in *page* scope.

Table 10.3 Additional option for `PDF_fit_pdi_page()`

option	description
clip-annotations	(Boolean) If true, annotations and form fields located outside the imported page are ignored. If false, annotations and form fields are imported regardless of their position. Default: false
cloneboxes	(Boolean; not allowed if the <i>topdown</i> option has been supplied in <code>PDF_begin_page_ext()</code> ; must match the <i>cloneboxes</i> option in <code>PDF_open_pdi_page()</code> ; only in page scope). This option clones the geometry of the PDF input page. Setting this option to true has the following consequences (default: false): <ul style="list-style-type: none"> ▶ All of the Rotate, MediaBox, TrimBox, ArtBox, BleedBox and CropBox entries in the imported page are copied to the output page. ▶ The page contents are placed such that the input page is duplicated; the user cannot change position or size of the placed page. The parameters <i>x</i>, <i>y</i> and the following options are ignored: <i>adjustpage</i>, <i>boxsize</i>, <i>fitmethod</i>, <i>orientate</i>, <i>position</i>, <i>rotate</i>, <i>scale</i>. Duplication of the input page geometry is only possible if the default coordinate system is active when calling <code>PDF_fit_pdi_page()</code>. ▶ Page boxes created by the <i>cloneboxes</i> option override the <i>artbox</i>, <i>bleedbox</i>, <i>cropbox</i>, <i>trimbox</i>, <i>mediabox</i>, and <i>rotate</i> options as well as the width and height parameters of <code>PDF_begin_page_ext()</code>.
useactions	(Boolean; only relevant in page scope; not for PDF/A) If true, all page actions of the placed page are copied to the output page. Individual imported actions override actions which may have been imported earlier from other PDI pages or which have been specified with the <i>action</i> option of <code>PDF_begin/end_page_ext()</code> . Default: false
useannots	(Option list; only in page scope; not allowed in PDF/UA if the page has been opened with <i>usetags=false</i>) Annotation types which are imported with the page (default: none if the current scope is different from page, otherwise all): 3D, Caret, Circle, FileAttachment, FreeText, Highlight, Ink, Line, Link, Movie¹, Polygon, PolyLine, Popup, Projection¹, Redact¹, RichMedia, Screen, Sound¹, Square, Squiggly, Stamp, StrikeOut, Text, Underline (Boolean) Annotation type(s) to import all (Keyword) If true, all annotation types are imported except types with the value false. none (Keyword) If true, no annotation types are imported except types with the value true. The option <i>all</i> overrides <i>none</i> .
usefields	(Option list; only in page scope; in PDF/UA not allowed if the page has been opened with <i>usetags=false</i>) Field types which are imported with the page (default: none): barcode, checkbox, combobox, listbox, pushbutton, radiobutton, signature, textfield (Boolean) Field type(s) to import all (Keyword) If true, all field types are imported except types with the value false. none (Keyword) If true, no fields are imported except types with the value true. The option <i>all</i> overrides <i>none</i> . Form fields can only be imported from a single PDF input document, but not from multiple documents. Form fields can be created either by importing from another PDF document or with <code>PDF_create_field()</code> . Both methods cannot be combined.

1. This annotation type cannot be created with `PDF_create_annotation()`.

C++ Java C# **double** *info_pdi_page*(int page, String keyword, String optlist)

Perl PHP **float** *info_pdi_page*(int page, string keyword, string optlist)

C **double** *PDF_info_pdi_page*(PDF *p, int page, const char *keyword, const char *optlist)

Perform formatting calculations for a PDI page and query the resulting metrics.

page A valid page handle retrieved with *PDF_open_pdi_page()*.

keyword A keyword specifying the requested information:

- ▶ Keywords for querying the results of object fitting according to Table 6.3: *boundingbox, fitscalex, fitscaley, height, objectheight, objectwidth, width, x1, y1, x2, y2, x3, y3, x4, y4*
- ▶ Page-related keywords according to Table 10.4: *mirroringx, mirroringy, pageheight, pagewidth, rotate, xid*
- ▶ Keywords related to Tagged PDF according to Table 10.4: *fittingpossible, lang, topleveltag, topleveltagcount*

optlist An option list specifying scaling and placement details:

- ▶ General option: *errorpolicy* (see Table 1.5)
- ▶ Fitting options according to Table 6.1 (if the PDF page has been opened with the *cloneboxes* option of *PDF_open_pdi_page()* these options are ignored): *boxsize, fitmethod, matchbox, orientate, position, rotate, scale*
- ▶ Options for page processing according to Table 9.3 and Table 10.3 don't make sense; however, they can be supplied but are ignored to facilitate unified option lists for *PDF_fit_pdi_page()* and *PDF_info_pdi_page()*: *adjustpage, gstate, useannots, usefields*
- ▶ Option for abbreviated structure element tagging according to Table 14.2: *tag*
- ▶ Option for selecting one of the page's top-level structure elements to retrieve some information from it: *index*

Returns The value of some page property as requested by *keyword*. If the requested property is not available for the page, the method returns 0. If an object handle is requested (e.g. *clippingpath*) this method will return a handle to the object, or -1 (in PHP: 0) if the object is not available. If the requested keyword produces *text*, a string index is returned, and the corresponding string must be retrieved with *PDF_get_string()*.

Details This method performs all calculations required for placing the imported page according to the supplied options, but will not actually create any output on the page. The reference point for placing the page is assumed to be {0 0}. If the *cloneboxes* option of *PDF_open_pdi_page()* has been supplied, the page will be placed on the same location (relative to the page boxes) as in the original page.

PDF/UA The check for *fittingpossible* is stricter than in non-PDF/UA mode.

Scope any except *object*

Table 10.4 Keywords for `PDF_info_pdi_page()`

keyword	description
fittingpossible	<p>(Only relevant for Tagged PDF output) Check whether the page can be placed with <code>PDF_fit_pdi_page()</code> in the current context. The value 1 is returned if the page can be placed. The value 0 is returned if fitting would fail (i.e. <code>PDF_fit_pdi_page()</code> would throw an exception) for one of the following reasons:</p> <ul style="list-style-type: none"> ▶ One of the page's top-level tags is not allowed under the currently active tag according to the nesting rules for structure elements. ▶ (Doesn't apply if the page has been loaded with <code>usetags=false</code> or placed as Artifact) The non-empty page is untagged or contains no structure elements, and direct content is not allowed as child of the currently active tag. ▶ (Doesn't apply if the page has been loaded with <code>usetags=false</code> or placed as Artifact) The page has already been placed. ▶ PDF/UA-1 with weak document structure: there is a gap in the heading level numbers between the current structure element and its parents, and the imported structure sub-hierarchy. <p>The tag option of <code>PDF_fit_pdi_page()</code> can be supplied and is taken into account for the result. Only the tagname suboption of the tag option is evaluated.</p> <p>Since the result is valid only for the current context this keyword should be used immediately before attempting to place a page.</p>
lang	String index for the Lang attribute of one of the imported page's top-level structure elements, or -1 if no Lang attribute could be determined. The index option can be used to select one of the top-level elements if there is more than one.
mirroringx, mirroringy	Horizontal or vertical mirroring of the page (expressed as 1 or -1) according to the supplied options
pageheight, pagewidth	Original page height and width in points
rotate	<p>If <code>cloneboxes=true</code>: the rotation angle of the imported page in degrees, i.e. the value of the page's Rotate key. Possible values are 0, 90, 180, and 270.</p> <p>If <code>cloneboxes=false</code>: always 0</p>
topleveltag	String index for the name of one of the imported page's top-level structure elements if the page has been opened with <code>usetags=true</code> and contains marked content associated with a structure element, otherwise -1 (e.g., for a page representing an Artifact). The index option can be used to select one of the top-level elements if there is more than one. If the tag is a custom element which is role mapped in the imported document's role map, the corresponding standard element name is reported, and not the custom element name.
topleveltagcount	Number of structure elements at the top level of the imported page's structure hierarchy. The lang and topLeveltag keywords can be used to retrieve information about these elements, using the index option to select one. 0 is returned if no structure elements are imported, either because the page has been imported with <code>usetags=false</code> or is untagged or contained no marked content corresponding to a structure element.
xid	(Only for PDF/VT) String index for the GTS_XID entry of the page, or -1 if no GTS_XID value has been assigned. The GTS_XID string can be used in the CIP4/Summary/Content/Referenced metadata property for DPM.

Table 10.5 Option for `PDF_info_pdi_page()`

option	description
index	(Integer; only relevant for the lang and topLeveltag keywords) Selects one of the page's top-level structure elements whose attribute is retrieved. The value must be in the range 0..(toplevelcount-1). Default: 0

10.3 Other PDI Processing

C++ Java C# *int process_pdi(int doc, int pagenumber, String optlist)*

Perl PHP *int process_pdi(int doc, int pagenumber, string optlist)*

C *int PDF_process_pdi(PDF *p, int doc, int pagenumber, const char* optlist)*

Process certain elements of an imported PDF document.

doc A valid PDF document handle retrieved with *PDF_open_pdi_document()*.

pagenumber (Positive integer; only for *action=copypageoutputintent*; should be -1 otherwise) The 1-based number of the page in the input document on which the output intent is located.

optlist An option list specifying PDI processing options:

- ▶ General option: *errorpolicy* (see Table 1.5)
- ▶ PDI processing options according to Table 10.6: *action, block*

Returns The value 1 if the method succeeded, or an error code of -1 (in PHP: 0) if the method call failed. If *errorpolicy=exception* this method throws an exception in case of an error.

If *action=copypageoutputintent* an output intent handle is returned if a valid page-level output intent was found for the page, or an error code of -1 (in PHP: 0) otherwise. It is recommended to check the presence of a page-level output intent with pCOS before calling *PDF_process_pdi()*.

If no Blocks were found on the input page for *action=copyallblocks* the method returns 1.

PDF/A The output intent can be set using this method with the *copyoutputintent* option or with *PDF_load_iccprofile()*. If only device-independent colors are used in the document no output intent is required.

PDF/X The output intent must be set using this method with the *copyoutputintent* option or with *PDF_load_iccprofile()*.

Scope *document* for *action=copyoutputintent* and *copypageoutputintent*,
page for *action=copyallblocks* and *action=copyblock*

Table 10.6 Options for `PDF_process_pdi()`

option	description
action	<p>(Keyword; required; this option does not require a page handle) Kind of PDF processing:</p> <p>copyallblocks (Only available in PPS) Copy all PDFlib Blocks from a page of the input document to the current output page according to the <code>block</code> option.</p> <p>copyblock (Only available in PPS) Copy a PDFlib Block from a page of the input document to the current output page according to the <code>block</code> option.</p> <p>copyoutputintent (Only relevant for PDF/A, PDF/X, and ignored otherwise) Copy the PDF/X or PDF/A output intent ICC profile of the imported document to the output document. The second and subsequent attempts to copy an output intent are ignored. If the document contains more than one output intent the first one is used. The existence of a document-level output intent can be queried with the <code>pCOS</code> path <code>/Root/OutputIntents[]</code>. If the input and output documents conform to PDF/X-4p or PDF/X-5n the reference to the external output intent ICC profile is copied. If a referenced profile is embedded the attached profile is also copied. The option <code>action=copyoutputintent</code> is not allowed if the input conforms to PDF/X-4p/5n, but not the output. If no page output intent is present the handle <code>-1</code> (in PHP <code>o</code>) is returned or an exception is thrown (depending on <code>errorpolicy</code>). PDI input document and the generated output must obey the following conditions (otherwise the action is rejected): If the input conforms to PDF/X-4p/5pg the output must conform to PDF/X-4p/5pg or PDF 2.0. If the input conforms to PDF/A the output must conform to PDF/A or PDF 2.0. If the input conforms to PDF/X the output must conform to PDF/X or PDF 2.0.</p> <p>copypageoutputintent (Only available for PDF/A or PDF/X) Return a handle for the PDF/A or PDF/X page-level output intent of the page designated by <code>pagenumber</code> and copy the output intent to the output document. The returned handle can be used with the <code>outputintents</code> option of <code>PDF_begin_page_ext()</code>. The existence of a page-level output intent can be queried with the <code>pCOS</code> path <code>pages[]/OutputIntents[]</code>.</p>
block	<p>(Option list; required for <code>action=copyallblocks</code> and <code>action=copyblock</code>) Specify details of the Block copying process. The following suboptions are supported:</p> <p>blockname (Name string; only for <code>action=copyblock</code> and required in this case) Name of the Block</p> <p>outputblockname (Name string; only for <code>action=copyblock</code>) Name under which the Block will be stored in the output page. Default: value of the <code>blockname</code> option</p> <p>pagenumber (Integer; required) The 1-based number of the page in the input document on which the Block is located.</p>

10.4 pCOS Methods

All pCOS methods work with paths designating the target object in the PDF document. pCOS paths are discussed in detail in the *pCOS Path Reference*.

Note In evaluation mode pCOS accepts input documents up to a maximum of 1 MB or 10 pages. However, the following elements can also be queried for larger documents in evaluation mode: page count, page dimensions, Block details, and all universal pseudo objects.

C++ Java C# **double** *pcos_get_number*(int doc, string path)

Perl PHP **double** *pcos_get_number*(long doc, string path)

C **double** *PDF_pcos_get_number*(PDF *p, int doc, const char *path, ...)

Get the value of a pCOS path with type *number* or *boolean*.

doc A valid document handle obtained with *PDF_open_pdi_document()*.

path A full pCOS path for a numerical or boolean object.

Additional parameters (C language binding only) A variable number of additional parameters can be supplied if the *key* parameter contains corresponding placeholders (%s for strings or %d for integers; use %% for a single percent sign). Using these parameters will save you from explicitly formatting complex paths containing variable numerical or string values. The client is responsible for making sure that the number and type of the placeholders matches the supplied additional parameters.

Returns The numerical value of the object identified by the pCOS path. For Boolean values 1 will be returned if they are *true*, and 0 otherwise.

Scope any

C++ Java C# **string** *pcos_get_string*(int doc, string path)

Perl PHP **string** *pcos_get_string*(long doc, string path)

C **const char ****PDF_pcos_get_string*(PDF *p, int doc, const char *path, ...)

Get the value of a pCOS path with type *name*, *number*, *string*, or *boolean*.

doc A valid document handle obtained with *PDF_open_pdi_document()*.

path A full pCOS path for a name, string, or boolean object.

Additional parameters (C language binding only) A variable number of additional parameters can be supplied if the *key* parameter contains corresponding placeholders (%s for strings or %d for integers; use %% for a single percent sign). Using these parameters will save you from explicitly formatting complex paths containing variable numerical or string values. The client is responsible for making sure that the number and type of the placeholders matches the supplied additional parameters.

Returns A string with the value of the object identified by the pCOS path. For Boolean values the strings *true* or *false* will be returned.

Details This method raises an exception if pCOS does not run in full mode and the type of the object is *string*. However, some objects can nevertheless be retrieved in restricted mode under certain conditions; see pCOS Path Reference for details.

This method assumes that strings retrieved from the PDF document are text strings. String objects which contain binary data should be retrieved with *PDF_pcos_get_stream()* instead which does not modify the data in any way.

Scope any

Bindings C language binding: The string is returned in UTF-8 format (on IBM System i and IBM Z: EBCDIC-UTF-8) without BOM. The returned strings are stored in a ring buffer with up to 10 entries. If more than 10 strings are queried, buffers will be reused, which means that clients must copy the strings if they want to access more than 10 strings in parallel. For example, up to 10 calls to this method can be used as parameters for a *printf()* statement since the return strings are guaranteed to be independent if no more than 10 strings are used at the same time.

The returned string is only valid until the next call to any API method; it must also not be passed as a parameter to other API method calls.

Java, .NET, and Python: the result is provided as Unicode string. If no text is available a null object is returned.

Perl and PHP language bindings: the result is provided as UTF-8 string. If no text is available a null object is returned.

C++ `const unsigned char *pcos_get_stream(int doc, int *len, string optlist, string path)`

Java C# `final byte[] pcos_get_stream(int doc, String optlist, String path)`

Perl PHP `string pcos_get_stream(long doc, string optlist, string path)`

C `const unsigned char *PDF_pcos_get_stream(PDF *p, int doc, int *len, const char *optlist, const char *path, ...)`

Get the contents of a pCOS path with type *stream*, *fstream*, or *string*.

doc A valid document handle obtained with *PDF_open_pdi_document()*.

len (C, C++ and RPG language bindings only) A pointer to a variable which will receive the length of the returned stream data in bytes.

optlist An option list specifying stream retrieval options according to Table 10.7.

path A full pCOS path for a stream or string object.

Additional parameters (C language binding only) A variable number of additional parameters can be supplied if the *key* parameter contains corresponding placeholders (*%s* for strings or *%d* for integers; use *%%* for a single percent sign). Using these parameters will save you from explicitly formatting complex paths containing variable numerical or string values. The client is responsible for making sure that the number and type of the placeholders matches the supplied additional parameters.

Returns The unencrypted data contained in the stream or string. The returned data will be empty (in C: NULL) if the stream or string is empty, or if the contents of encrypted attachments in an unencrypted document are queried and the attachment password has not been supplied.

If the object has type *stream*, all filters will be removed from the stream contents (i.e. the actual raw data will be returned) unless *keepfilter=true*. If the object has type *fstream* or *string* the data will be delivered exactly as found in the PDF file, with the exception of ASCII85 and ASCIIHex filters which will be removed.

Details This method throws an exception if pCOS does not run in full mode. As an exception, the object */Root/Metadata* can also be retrieved in restricted pCOS mode if *nocopy=false* or *plainmetadata=true*. An exception will also be thrown if *path* does not point to an object of type *stream*, *fstream*, or *string*.

Despite its name this method can also be used to retrieve objects of type *string*. Unlike *PDF.pcos_get_string()*, which treats the object as a text string, this method will not modify the returned data in any way. Binary string data is rarely used in PDF, and cannot be reliably detected automatically. The user is therefore responsible for selecting the appropriate method for retrieving string objects as binary data or text.

Scope any

Bindings C language binding: If *convert=unicode* is supplied, the string is returned in UTF-8 format without BOM (on IBM System i and IBM Z: EBCDIC-UTF8).

C and C++ language bindings: The returned data buffer can be used until the next call to this method.

Note Python: the result will be returned as 8-bit string (Python 3: *bytes*).

Table 10.7 Options for *PDF.pcos_get_stream()*

option	description
convert	(Keyword; ignored for streams which are compressed with unsupported filters) Controls whether or not the string or stream contents are converted (default: none): none Treat the contents as binary data without any conversion. unicode Treat the contents as textual data (i.e. exactly as in <i>PDF.pcos_get_string()</i>), and normalize it to Unicode. In non-Unicode-capable language bindings the data is converted to UTF-8 without BOM. This option is required for the data type »text stream« in PDF which is rarely used (e.g. it can be used for JavaScript, although the majority of JavaScripts is contained in string objects, not stream objects).
keepfilter	(Boolean; recommended only for image data streams; ignored for streams which are compressed with unsupported filters) If true, the stream data will be compressed with the filter which is specified in the image's <i>filterinfo</i> pseudo object. If false, the stream data will be uncompressed. Default: true for all unsupported filters, false otherwise

11 Block Filling Methods (PPS)

API methods in this chapter:

- ▶ `PDF_fill_textblock()`
- ▶ `PDF_fill_imageblock()`
- ▶ `PDF_fill_pdfblock()`
- ▶ `PDF_fill_graphicsblock()`

The PDFlib Personalization Server (PPS) offers dedicated methods for processing variable Blocks of type *Text*, *Image*, and *PDF*. These PDFlib Blocks must be contained in the imported PDF page, but will not be retained in the generated output. The imported page must have been placed on the output page with `PDF_fit_pdi_page()` before using any of the Block filling methods. When calculating the Block position on the page, the Block methods take into account the scaling options which have been in effect when placing the imported page with `PDF_fit_pdi_page()`.

Cookbook The Block processing methods discussed in this chapter require the PDFlib Personalization Server (PPS). The PDFlib Block plugin for Adobe Acrobat is required for creating PDFlib Blocks in PDF templates interactively. Alternatively, Blocks can be created with PPS itself.

11.1 Options for all Block Types

This section contains options for `PDF_fill_textblock()`, `PDF_image_block()`, `PDF_fill_pdfblock()`, and `PDF_graphics_block()`. Options which are specific for a particular Block type are listed in the next sections. Almost all Block properties can be overridden with options with the same name, except for the following properties which cannot be overridden with options:

Name, Description, Subtype, Type

11.1.1 Rectangle Options

Table 11.1 lists rectangle options for `PDF_fill_textblock()`, `PDF_image_block()`, `PDF_fill_pdfblock()`, and `PDF_graphics_block()`.

Table 11.1 Rectangle options for the `PDF_fill_*block()` methods

option	description
background-color	(Color) Fill color for the Block; this color will be applied before filling the Block. This may be useful to hide existing page contents. Default: none
bordercolor	(Color) Border color for the Block; this color will be applied before filling the Block. Default: none
linewidth	(Float; must be greater than 0) Stroke width of the line used to draw the Block rectangle; only used if bordercolor is set. Default: 1
Rect	(Rectangle) Block coordinates in the coordinate system of the Block PDF. Block position and size can be specified with the <code>repoint</code> and <code>boxsize fitting</code> options (see Table 6.1). Both options accept absolute and relative values.

Table 11.1 Rectangle options for the `PDF_fill_*block()` methods

option	description
Status	(Keyword) Describes how the Block will be processed (default: active):
active	The Block is processed according to its properties.
ignore	The Block is ignored.
ignoredefault	Like active except that the defaulttext/image/pdf properties are ignored, i.e. the Block remains empty if no contents have been provided. This may be useful to make sure that the Block's default contents will not be used for filling Blocks on the server side although the Block may contain default contents for the Preview in the Block Plugin. It can also be used to disable the default contents for previewing a Block without removing it from the Block properties.
static	No variable contents will be placed; instead, the Block's default text, image, or PDF contents will be used if available.

11.1.2 Tagging Option

Table 11.2 describes the tagging option for `PDF_fill_textblock()`, `PDF_image_block()`, `PDF_fill_pdfblock()`, and `PDF_graphics_block()`. Options which are specific for a particular Block type (i.e. text, image, or PDF Blocks) are listed in the next sections.

Table 11.2 Tagging option for the `PDF_fill_*block()` methods

option	description
tag	(Option list) Options for structure element tagging according to Table 14.2.

11.2 Textline and Textflow Blocks

C++ Java C# *int fill_textblock(int page, String blockname, String text, String optlist)*

Perl PHP *int fill_textblock(int page, string blockname, string text, string optlist)*

C *int PDF_fill_textblock(PDF *p,
int page, const char *blockname, const char *text, int len, const char *optlist)*

Fill a Textline or Textflow Block with variable data according to its properties.

page A valid PDF page handle for a page containing PDFlib Blocks. The input PDF page with Blocks must have been placed on the page earlier, either directly with *PDF_fit_pdi_page()*, indirectly in a table cell with *PDF_fit_table()*, or as contents of a PDF Block with *PDF_fill_pdfblock()*.

blockname (Name string) Name of the Block.

text (Content string) The text to be filled into the Block, or an empty string if the default text (as defined by Block properties) is to be used. If the *textflowhandle* option is supplied and contains a valid Textflow handle this parameter is ignored.

len (C language binding only) Length of *text* (in bytes). If *len = 0* a null-terminated string must be provided.

optlist An option list specifying text Block filling options. The following options are supported:

- ▶ General option: *errorpolicy* (see Table 1.5)
- ▶ Rectangle options for Block filling methods according to Table 11.1:
Rect, Status, backgroundcolor, bordercolor, linewidth
- ▶ Fitting options (see Section 6.1, »Object Fitting«, page 123)
- ▶ Textline Blocks, i.e. the *textflow* property or option is *false*:
all Textline options (see Section 5.1, »Single-Line Text with Textlines«, page 89)
- ▶ Textflow Blocks, i.e. the *textflow* property or option is *true*:
all options for *PDF_add/create_textflow()* (see Section 5.2, »Multi-Line Text with Textflows«, page 96) and all options for *PDF_fit_textflow()* (see Table 5.13)
- ▶ Text Block options according to Table 11.3: *textflow, textflowhandle*
- ▶ Option for default contents: *defaulttext* (see PDFlib Tutorial)

Returns -1 (in PHP: 0) if the named Block doesn't exist on the page, the Block cannot be filled (e.g. due to font problems), the Block has wrong type, or the Block requires a newer PPS version for processing; 1 if the Block could be processed successfully.

If the *textflowhandle* option is supplied, a Textflow handle is returned which can be used in subsequent calls to *PDF_fill_textblock()* or *PDF_info_textflow()*. If the property *Status* has the value *ignore*, a handle to an empty Textflow is returned if *textflowhandle=1*; otherwise the supplied Textflow handle is returned without modifying the Textflow (since no output is created). The user is responsible for deleting the returned Textflow handle with *PDF_delete_textflow()* at the end.

If the PDF document is found to be corrupt, this method throws an exception or returns -1 subject to the *errorpolicy* option.

Details The supplied text is formatted into the Block subject to the Block's properties. If *text* is empty the method uses the Block's default text if available (unless *Status=*

ignoredefault), and silently return otherwise. This may be useful to take advantage of other Block properties, such as fill or stroke color.

Font selection: If neither the *font* option is supplied nor implicit font loading based on options is used, the font is implicitly loaded based on the Block properties. Since the encoding for the font can only be specified as an option, but not as a Block property it will be set as follows by default:

- ▶ *builtin* if the font is a symbolic font and *charref=false* and (only relevant for non-Unicode aware languages) *textformat=auto* or *bytes*.
- ▶ *unicode* otherwise.

It is recommended to avoid the *encoding*, *charref* and *textformat* options if *defaulttext* is to be used.

Linking Textflow Blocks. If a Textflow doesn't fit into a Block, the *textflowhandle* option can be used to connect multiple Blocks to a chain so that they hold multiple parts of the same Textflow:

- ▶ In the first call a value of -1 (in PHP: 0) must be supplied. The Textflow handle created internally is returned by *PDF_fill_textblock()*, and must be stored by the user.
- ▶ In the next call the Textflow handle returned in the previous step can be supplied to the *textflowhandle* option (the text supplied in the *text* parameter is ignored in this case, and should be empty). The Block is filled with the remainder of the Textflow.
- ▶ This process can be repeated with more Textflow Blocks.
- ▶ The returned Textflow handle can be supplied to *PDF_info_textflow()* in order to determine the results of Block filling, e.g. the end position of the text.

This process can be repeated for an arbitrary number of Blocks.

PDF/UA Block decoration, i.e. ruling and shading created according to the *backgroundcolor*, *bordercolor*, *linewidth* properties is automatically tagged as *Artifact*.

Scope *page, pattern, template, glyph*

Table 11.3 Additional options for *PDF_fill_textblock()*

option	description
textflow	<p>(Boolean) Control single- or multiline processing. This property can be used to switch between Textline and Textflow Blocks:</p> <p>false Text can span a single line and will be processed with <i>PDF_fit_textline()</i>.</p> <p>true Text can span multiple lines and will be processed with <i>PDF_fit_textflow()</i>.</p> <p>The default depends on the Block type: true for Textflow Blocks, and false for Textline Blocks</p>
textflow-handle	<p>(Textflow handle; only for <i>PDF_fill_textblock()</i> with <i>textflow=true</i>) This option can be used for Textflow Block chaining. For the first Block in a chain of Blocks a value of -1 (in PHP: 0) must be supplied; the value returned by this method can be supplied as Textflow handle in subsequent calls for other Blocks in the chain. If this option is supplied with a value different from -1 (in PHP: 0) the default of <i>fitmethod</i> is changed to <i>clip</i>.</p> <p>All properties in the Text Preparation, Text Formatting and Appearance property groups of the Block are ignored if <i>textflowhandle</i> is supplied since the corresponding values used for creating the Textflow will be applied.</p>

11.3 Image Blocks

C++ Java C# *int fill_imageblock(int page, String blockname, int image, String optlist)*

Perl PHP *int fill_imageblock(int page, string blockname, int image, string optlist)*

C *int PDF_fill_imageblock(PDF *p, int page, const char *blockname, int image, const char *optlist)*

Fill an image Block with variable data according to its properties.

page A valid PDF page handle for a page containing PDFlib Blocks. The input PDF page with Blocks must have been placed on the page earlier, either directly with *PDF_fit_pdi_page()*, indirectly in a table cell with *PDF_fit_table()*, or as contents of a PDF Block with *PDF_fill_pdfblock()*.

blockname (Name string) Name of the Block.

image A valid image handle for the image to be filled into the Block, or -1 if the default image (as defined by Block properties) is to be used.

optlist An option list specifying image Block filling options. The following options are supported:

- ▶ General option: *errorpolicy* (see Table 1.5)
- ▶ Rectangle options for Block filling methods according to Table 11.1:
Rect, Status, backgroundcolor, bordercolor, linewidth
- ▶ Fitting options (see Section 6.1, »Object Fitting«, page 123)
- ▶ Options for image processing according to Table 9.3
- ▶ Option for default contents: *defaultimage* (see PDFlib Tutorial)

Returns -1 (in PHP: 0) if the named Block doesn't exist on the page, the Block cannot be filled, the Block has wrong type, or the Block requires a newer PPS version for processing; 1 if the Block could be processed successfully. Use *PDF_get_errmsg()* to get more information about the nature of the problem.

Details The image referred to by the supplied image handle will be placed in the Block, subject to the Block's properties. If *image* is -1 (in PHP: 0) the method will use the Block's default image if available (unless *Status=ignoredefault*), and silently return otherwise.

If the PDF document is found to be corrupt, this method will either throw an exception or return -1 subject to the *errorpolicy* option.

PDF/UA All raster images must be tagged as *Artifact* or *Figure* with a preceding call to *PDF_begin_item()*.

Block decoration, i.e. ruling and shading created according to the *backgroundcolor, bordercolor, linewidth* properties is automatically tagged as *Artifact*.

Scope *page, pattern, template, glyph*

11.4 PDF Blocks

C++ Java C# *int fill_pdfblock(int page, String blockname, int contents, String optlist)*

Perl PHP *int fill_pdfblock(int page, string blockname, int contents, string optlist)*

C *int PDF_fill_pdfblock(PDF *p, int page, const char *blockname, int contents, const char *optlist)*

Fill a PDF Block with variable data according to its properties.

page A valid PDF page handle for a page containing PDFlib Blocks. The input PDF page with Blocks must have been placed on the page earlier, either directly with *PDF_fit_pdi_page()*, indirectly in a table cell with *PDF_fit_table()*, or as contents of a PDF Block with *PDF_fill_pdfblock()*.

blockname (Name string) Name of the Block.

contents A valid PDF page handle for the PDF page to be filled into the Block, or -1 if the default PDF page (as defined by Block properties) is to be used.

optlist An option list specifying PDF Block filling options. The following options are supported:

- ▶ General option: *errorpolicy* (see Table 1.5)
- ▶ Rectangle options for Block filling methods according to Table 11.1:
Rect, Status, backgroundcolor, bordercolor, linewidth
- ▶ Fitting options (see Section 6.1, »Object Fitting«, page 123)
- ▶ Options for page processing according to Table 9.3
- ▶ Options for default contents: *defaultpdf, defaultpdfpage* (see PDFlib Tutorial)

Returns -1 (in PHP: 0) if the named Block doesn't exist on the page, the Block cannot be filled, the Block has wrong type, or the Block requires a newer PPS version for processing; 1 if the Block could be processed successfully. Use *PDF_get_errmsg()* to get more information about the nature of the problem.

Details The PDF page referred to by the supplied page handle *contents* will be placed in the Block, subject to the Block's properties. If *contents* is -1 (in PHP: 0) the method will use the Block's default PDF page if available (unless *Status=ignoredefault*), and silently return otherwise.

If the PDF document is found to be corrupt, this method will either throw an exception or return -1 subject to the *errorpolicy* option.

PDF/UA Block decoration, i.e. ruling and shading created according to the *backgroundcolor, bordercolor, linewidth* properties is automatically tagged as *Artifact*.

Scope *page, pattern, template, glyph*

11.5 Graphics Blocks

C++ Java C# *int fill_graphicsblock(int page, String blockname, int contents, String optlist)*

Perl PHP *int fill_graphicsblock(int page, string blockname, int contents, string optlist)*

C *int PDF_fill_graphicsblock(PDF *p, int page, const char *blockname, int contents, const char *optlist)*

Fill a graphics Block with variable data according to its properties.

page A valid PDF page handle for a page containing PDFlib Blocks. The input PDF page with Blocks must have been placed on the page earlier, either directly with *PDF_fit_pdi_page()*, indirectly in a table cell with *PDF_fit_table()*, or as contents of a PDF Block with *PDF_fill_pdfblock()*.

blockname (Name string) Name of the Block.

contents A valid graphics handle for the graphics to be filled into the Block, or -1 if the default graphics (as defined by Block properties) is to be used.

optlist An option list specifying graphics Block filling options. The following options are supported:

- ▶ General option: *errorpolicy* (see Table 1.5)
- ▶ Rectangle options for Block filling methods according to Table 11.1:
Rect, Status, backgroundcolor, bordercolor, linewidth
- ▶ Fitting options (see Section 6.1, »Object Fitting«, page 123)
- ▶ Options for graphics processing according to Table 9.3
- ▶ Option for default contents: *defaultgraphics* (see PDFlib Tutorial)

Returns -1 (in PHP: 0) if the named Block doesn't exist on the page, the Block cannot be filled, the Block has wrong type, or the Block requires a newer PPS version for processing; 1 if the Block could be processed successfully. Use *PDF_get_errmsg()* to get more information about the nature of the problem.

Details The graphics referred to by the supplied graphics handle will be placed in the Block, subject to the Block's properties. If *graphics* is -1 (in PHP: 0) the method will use the Block's default graphics if available (unless *Status=ignoredefault*), and silently return otherwise.

If the PDF document is found to be corrupt, this method will either throw an exception or return -1 subject to the *errorpolicy* option.

PDF/UA Block decoration, i.e. ruling and shading created according to the *backgroundcolor, bordercolor, linewidth* properties is automatically tagged as *Artifact*.

Scope *page, pattern, template, glyph*



12 Interactive Features

API methods in this chapter:

- ▶ `PDF_create_bookmark()`
- ▶ `PDF_create_annotation()`
- ▶ `PDF_create_field()`
- ▶ `PDF_create_fieldgroup()`
- ▶ `PDF_create_action()`
- ▶ `PDF_add_nameddest()`

12.1 Bookmarks

C++ Java C# `int create_bookmark(String text, String optlist)`

Perl PHP `int create_bookmark(string text, string optlist)`

C `int PDF_create_bookmark(PDF *p, const char *text, int len, const char *optlist)`

Create a bookmark subject to various options.

text (Hypertext string) Text for the bookmark.

len (C language binding only) Length of *text* (in bytes). If *len = 0* a null-terminated string must be provided.

optlist An option list specifying the bookmark's properties. The following options can be used:

- ▶ General option: *errorpolicy* (see Table 1.5)
- ▶ Bookmark control options according to Table 12.1:
action, destination, destname, fontstyle, index, item, open, parent, textcolor
- ▶ Encoding options for C and Perl, PHP, Ruby with *stringformat=legacy*:
hypertextencoding, hypertextformat (see Table 2.1)

Returns A handle for the generated bookmark, which may be used with the *parent* option in subsequent calls.

Details This method adds a PDF bookmark with the supplied *text*. Unless the *destination* option is specified the bookmark points to the current page (or the least recently generated page if used in *document* scope, or the first page if used before the first page).

Creating bookmarks sets the *openmode* option of `PDF_begin/end_document()` to *bookmarks* unless another mode has explicitly been set.

PDF/UA Creating bookmarks is recommended for PDF/UA-1.

Scope *document, page*; if a rendition action is supplied which has been created with the *target* option this method can only be called in *page* scope and on the page where the target annotation is located.

Table 12.1 Options for `PDF_create_bookmark()`

option	description
action	(Action list) List of bookmark actions for the following event (default: GoTo action with the target specified in the destination option): activate Actions to be performed when the bookmark is activated. All types of actions are permitted.
destination	(Option list; ignored if an activate action has been specified) Bookmark destination according to Table 12.11. Default: {type fitwindow page 0} if destination, destname, and action are absent.
destname	(Hypertext string; may be empty; ignored if the destination option has been specified) Name of a destination which has been defined with <code>PDF_add_nameddest()</code> . Destination or destname actions will be dominant over this option. If destname is an empty string (i.e. {}) and neither destination nor action are specified, the bookmark won't have any action. This may be useful for separator bookmark.
fontstyle	(Keyword) Font style of the bookmark text: normal, bold, italic, bolditalic. Default: normal
index	(Integer) Index where to insert the bookmark within the parent. Values between 0 and the number of bookmarks of the same level will be used to insert the bookmark at that specific location within the parent. The value -1 can be used to insert the bookmark as the last one on the current level. Default: -1. However, for inserted or resumed pages bookmarks will be placed as if all pages had been generated in their physical order (i.e. the bookmarks reflect the page order).
item	(Item handle or keyword; the handle must refer to an active structure element, but not to an inline or pseudo element; only for Tagged PDF) Handle for a structure item which will be associated with the bookmark. The value 0 always refers to the structure tree root. The value -1 and the equivalent keyword current refer to the currently active element.
open	(Boolean) If false, subordinate bookmarks will not be visible. If true, all children will be folded out. Default: false
parent	(Bookmark handle) The new bookmark will be specified as a subordinate of the bookmark specified in the handle. If parent=0 a new top-level bookmark will be created. Default: 0
textcolor	(Color) Color of the bookmark text. Supported color spaces: none, gray, rgb. Default: rgb {0 0 0} (=black)

12.2 Annotations

```
C++ Java C# void create_annotation(double llx, double lly, double urx, double ury, String type, String optlist)
Perl PHP create_annotation(float llx, float lly, float urx, float ury, string type, string optlist)
C void PDF_create_annotation(PDF *p,
    double llx, double lly, double urx, double ury, const char *type, const char *optlist)
```

Create an annotation on the current page.

llx, lly, urx, ury x and y coordinates of the lower left and upper right corner of the annotation rectangle in default coordinates (if the *usercoordinates* option is *false*) or user coordinates (if *usercoordinates* is *true*).

Note that annotation coordinates are different from the parameters of `PDF_rect()`: while `PDF_create_annotation()` expects the coordinates of two corners, `PDF_rect()` expects the coordinates of one corner plus width and height.

If the *usematchbox* option has been specified, *llx/lly/urx/ury* are ignored. The annotation rectangle may be slightly enlarged for some combinations of annotation type and options.

type Annotation type according to Table 12.2. Markup7 annotations are designated in the table since some options apply only to markup annotations.

Table 12.2 Annotation types

type	markup annotation¹	options specific for this type (in addition to common options)
3D		(PDF 1.6; 3D artwork can also be created with RichMedia annotations which provide more functionality than 3D annotations) 3D model: 3Dactivate, 3Ddata, 3Dinteractive, 3Dshared, 3Dinitialview
Caret	yes	(PDF 1.5) font, rectdiff, symbol
Circle	yes	cloudy, createrichtext, inreplyto, interiorcolor, replyto
File-Attachment	yes	attachment, createrichtext, iconname, inreplyto, replyto
FreeText	yes	(Unlike Text annotations the text is always visible and doesn't require a popup) alignment, calloutline, cloudy, createrichtext, endingstyles, fillcolor, font, fontsize, inreplyto, orientate, replyto
Highlight	yes	createrichtext, inreplyto, polylinelist, replyto
Ink	yes	createrichtext, inreplyto, polylinelist, replyto
Line	yes	captionoffset, captionposition, createrichtext, endingstyles, font, interiorcolor, inreplyto, leaderlength, leaderoffset, line, showcaption, replyto
Link		destination, destname, highlight
Polygon	yes	(PDF 1.5; vertices connected by straight lines): cloudy, createrichtext, inreplyto, interiorcolor, polylinelist, replyto
PolyLine	yes	(PDF 1.5; similar to polygons, except that the first and last vertices are not connected) createrichtext, endingstyles, inreplyto, interiorcolor, polylinelist, replyto
Popup		open, parentname

Table 12.2 Annotation types

type	markup annotation ¹	options specific for this type (in addition to common options)
RichMedia		(PDF 1.7ext3; Screen annotations are recommended for video and sound) richmedia
Screen		(PDF 1.5) Media details are specified in an action with type=Rendition.
Square	yes	cloudy, createrichtext, inreplyto, interiorcolor, replyto
Squiggly	yes	(Squiggly-underline annotation) createrichtext, inreplyto, polylinelist, replyto
Stamp	yes	createrichtext, iconname, inreplyto, font, orientate, replyto The stamp text can be created with the options createrichtext, contents, iconname or template. If none of these options is supplied iconname=draft is used as default.
StrikeOut	yes	createrichtext, inreplyto, polylinelist, replyto
Text	yes	(The text is displayed in a separate window which can be opened or closed; in Acrobat this type is called sticky note) createrichtext, iconname, inreplyto, open, replyto, state, statemodel
Underline	yes	createrichtext, inreplyto, polylinelist, replyto

1. Relevant for Popup annotations and some options (see Table 12.3).

optlist Option list specifying annotation properties:

- ▶ Common options according to Table 12.3 are supported for all annotation types: *action, annotcolor, associatedfiles, blendmode, borderstyle, cloudy, contents, createdate, custom, dasharray, datestring, display, lang, layer, linewidth, locked, lockedcontents, name, opacity, opacityfill, readonly, rotate, subject, template, title, usematchbox, usercoordinates, zoom*
- ▶ Type-specific options according to Table 12.3 for some annotation types according to Table 12.2: *alignment, calloutline, createrichtext, destname, endingstyles, fillcolor, font, fontsize, highlight, iconname, inreplyto, interiorcolor, open, orientate, parentname, polylinelist, rectdiff, replyto, richmedia, state, statemodel, symbol*
- ▶ Options for type=Line according to Table 12.4: *captionoffset, captionposition, leaderlength, leaderoffset, line, showcaption*
- ▶ Options for type=3D according to Table 13.12: *3Dactivate, 3Ddata, 3Dinteractive, 3Dshared, 3Dinitialview*
- ▶ Option for abbreviated structure element tagging according to Table 14.2: *tag*
- ▶ Encoding option for C and Perl, PHP, Ruby with *stringformat=legacy: hypertextencoding* (see Table 2.1)

Details This method creates an annotation on the current page using the supplied box coordinates or the box specified in the *matchbox* option.

While most annotation types must have rectangles with edges parallel to the page edges, the following annotation types may also appear rotated or skewed (e.g. if the coordinate system has been rotated): *Highlight, Link, Squiggly, StrikeOut, Underline*.

The following annotation types are not supported in *PDF_create_annotation()*, but can be imported from another PDF document with *PDF_fit_pdi_page()*: *Movie, Projection, Redact, Sound*.

Unless the *font* or *template* option is supplied, the font *NotoSans-Regular* must be configured for type=*Caret, FreeText, Line* (only if *showcaption=true*), and the font *NotoSans-*

Bold must be configured for *type=Stamp*. Both fonts are included in the PDFlib distribution (see PDFlib Tutorial). Since these fonts don't include CJK glyphs a suitable CJK font should be supplied to the *font* option if CJK text is used with *type=FreeText* or *Line*.

In Tagged PDF mode this method automatically creates an *OBJR* element for the annotation. The user must create a corresponding *Link* or *Annot* container element (see PDFlib Tutorial) before calling this method.

PDF/A PDF/A-1: *type=FileAttachment* and *Highlight* are not allowed. The following types are only allowed if an RGB output intent has been specified: *Freetext, Text, Stamp, Underline, Squiggly, Strikeout, Polygon, Polyline, Line, Square, Circle, Ink*.
 PDF/A-2/3: *type=3D* and *Screen* not allowed.

Color restrictions for the options *annotcolor, fillcolor* and *interiorcolor*:

PDF/A-1/2/3: grayscale color is only allowed if an output intent (any type) has been specified; CMYK color is only allowed if a CMYK output intent has been specified.

PDF/A-1a/2a/3a: the *contents* option is recommended for annotations which do not display text.

Some options are restricted, see Table 12.3.

PDF/UA Annotations of *type=Link* must be contained in a *Link* structure element. Annotations of all other types except *Popup* must be contained in an *Annot* structure element.

The option *contents* or the option *tag* with the suboption *ActualText* is required for visible annotations.

PDF/X Annotations are only allowed if they are positioned completely outside the *BleedBox* (or *TrimBox/ArtBox* if no *BleedBox* is present). For the following types the annotation rectangle must be located outside the relevant page box at least one half the line width: *Square, Circle, Ink, PolyLine, Polygon*.

The caption of *Line* annotations must be located completely inside the rectangle.

Some options are restricted, see Table 12.3.

PDF/X-3: *type=Caret, FileAttachment* and *Highlight* are not allowed.

PDF/X-4/5: *type=Caret, Highlight, Screen* are not allowed.

Scope page

Table 12.3 Options for *PDF_create_annotation()*

option	description
action	(Action list; not for <i>type=Popup</i>) List of annotation actions for the following events (default: empty list). All types of actions are permitted except rendition actions which are only allowed for <i>type=Screen</i> : activate (Only for <i>type=Link</i> and <i>Screen</i>) Actions to be performed when the annotation is activated. close (PDF 1.5) Actions to be performed when the page is closed. invisible (PDF 1.5) Actions to be performed when the page is no longer visible. open (PDF 1.5) Actions to be performed when the page is opened. visible (PDF 1.5) Actions to be performed when the page becomes visible.
alignment	(Keyword; only for <i>type=FreeText</i>) Alignment of text in the annotation: left, center, or right. Default: left

Table 12.3 Options for `PDF_create_annotation()`

option	description
annotcolor	(Color; not for type=Popup; PDF/A color restrictions apply, see above table) Color of the background of the annotation's icon when closed, the title bar of the annotation's popup window, the color of markings and the border of Link annotations. Supported color spaces: none, gray, rgb, and (in PDF 1.6) cmyk. Default: various RGB colors depending on the annotation type
associatedfiles	(List of asset handles; only for PDF 2.0 and PDF/A-3) Asset handles for files associated with the annotation. The files must have been loaded with <code>PDF_load_asset()</code> and type=attachment.
attachment	(Asset handle; only for type=FileAttachment; required) File attachment which must have been loaded with <code>PDF_load_asset()</code> and type=attachment.
blendmode	(Keyword; PDF 2.0) Blend mode for transparency operations (see blendmode in Table 7.2). Default: None
borderstyle	(Keyword) Style of the annotation border and the line for type=Polygon, PolyLine, Line, Square, Circle, Ink: solid, beveled, dashed, inset, or underline. Note that the beveled, inset, and underline styles do not work reliably in Acrobat. Default: solid
calloutline¹	(List of four or six floats; PDF 1.6; only for type=FreeText; not allowed in PDF/X mode) List of 4 or 6 float values specifying a callout line attached to the annotation. Six coordinates {x1 y1 x2 y2 x3 y3} represent the start, knee point, and end point (which is usually placed on the annotation border) of the line. Four coordinates {x1 y1 x2 y2} represent the start and end point of the line. The start point is decorated with the symbol specified in the first keyword of the endingstyles option.
cloudy	(Float in the range 0-2; only for type=FreeText in PDF 1.6 and type=Square in PDF 1.5; not allowed in PDF/X) Intensity of the «cloud» effect used to render the polygon from 0 (no effect) to 2 (maximum effect). This option may have unexpected effects for very small annotation rectangles. If this option is used the borderstyle option is ignored. Default: 0
contents²	(String for type=FreeText and Line with showcaption=true, otherwise Hypertext string; not for type=Popup; PDF/UA-1: required if no ActualText is supplied in the tag option, and always required for type=Link) Text to be displayed for the annotation or (if the annotation does not display text) an alternate description of its contents in human-readable form. Carriage return or line feed characters can be used to force a new line.
createdate	(Boolean; PDF 1.5) If true, a date/time entry is created for the annotation. Default: true for Markup annotations, false otherwise
createrich-text²	(Option list; only for markup annotations; not in PDF/A or PDF/X mode; PDF 1.5) Create rich text contents from a Textflow. This may be useful to generate formatted text for annotations (the Textflow can be placed partially on the page and partially in an annotation). The specified rich text should be equivalent to the contents option if supplied. <p>textflow (Textflow handle; required) A Textflow which is attached to the annotation as rich text. Color specifications in the Textflow must use only grayscale or RGB colors. If the Textflow handle has been supplied to <code>PDF_fit_textflow/table()</code> before the call to <code>PDF_create_annotation()</code> only the remaining text is used for the annotation. If no more text is available the Textflow is restarted from the beginning. Using a Textflow for an annotation does not affect subsequent calls to <code>PDF_fit_textflow/table()</code>.</p> <p>userunit (Keyword) Measurement unit for scalar values (e.g. firstlinedist, fontsize): cm (centimeter), in (inches), mm (millimeters), or pt (points). Default: pt</p> <p>The following Textflow options are honored when creating rich text, all others are ignored: nextline, alignment, fillcolor, underline, strikeout, font, fontsize, textrise, textformatting options. Rich Text is emitted as XFA, an XML language. Therefore XML syntax characters in a Textflow for annotation rich text must be quoted appropriately, e.g. &lt; ; However, this is difficult if the Textflow is used with <code>PDF_fit_textflow()</code> and <code>PDF_create_annotation()</code> since the boundary between both is usually not known (Textflow does not require quoted characters, while XFA for annotations does). Setting font and alignment doesn't have the expected effect in Acrobat. It is recommended to use the text option features={_none} in the Textflow to disable all OpenType features since these cannot be represented in rich text.</p>

Table 12.3 Options for `PDF_create_annotation()`

option	description
custom	(List of option lists) Create custom annotation entries which may be useful for applications such as inserting processing instructions for NexPress printing machines. Each option list describes an entry in the annotation dictionary: key (String; required) Name of the dictionary key. Any non-standard PDF key can be specified, as well as the following standard keys: Contents, Name (option <code>iconname</code>), NM (option <code>name</code>), and Open. type (Keyword; required) Type of the corresponding value: boolean, name, or string value (Hypertext string if <code>type=string</code> , otherwise string; required) Value of the entry
dasharray	(List of two non-negative floats; only for <code>borderstyle=dashed</code>). Lengths of dashes and gaps for a dashed border. Default: 3 3
datestring	(String; forces <code>createdate=true</code> ; not for <code>type=Popup</code>) PDF date string with the desired creation date for the annotation. The date string is written unmodified to the annotation dictionary. It is an error to supply an invalid date string. Default (only relevant if <code>createdate=true</code> : the current date
destination	(Option list; only for <code>type=Link</code> ; ignored if an activate action has been specified) Option list according to Table 12.11 defining the link's destination
destname	(Hypertext string; only for <code>type=Link</code> ; ignored if the destination option has been specified) Name of a destination which has been defined with <code>PDF_add_nameddest()</code> . Actions created with the destination or <code>destname</code> options of <code>PDF_create_action()</code> are dominant over this option.
display	(Keyword; only visible allowed in PDF/A) Visibility on screen and paper: visible, hidden, noview, noprint. Default: visible
endingstyles	(Keyword list; only for <code>type=FreeText</code> , Line, PolyLine) List with two keywords specifying the line ending styles. The second keyword is ignored for <code>type=FreeText</code> (default: {none none}): butt (PDF 1.5), circle, closedarrow, diamond, none, openarrow, rclosedarrow (PDF 1.5), ropenarrow (PDF 1.5), slash (PDF 1.6), square
fillcolor	(Color; only for <code>type=FreeText</code> ; PDF/A color restrictions apply, see above table) Border color and fill color of the text. Supported color spaces are none, gray, rgb, and (in PDF 1.6) cmYk. Default: {rgb 0 0 0}
font	(Font handle; only for <code>type=Caret</code> , FreeText, Line, Stamp) Font to be used for the annotation. Default: NotoSans-Bold for <code>type=Stamp</code> , NotoSans-Regular otherwise; access to these fonts must be configured.
fontsize¹	(FontSize; only for <code>type=Caret</code> , FreeText, Line; required) Font size. The value 0 or keyword auto means that the fontsize is adjusted to the rectangle.
highlight	(Keyword; only for <code>type=Link</code>) Highlight mode of the annotation when the user clicks on it: none, invert, outline, push. Default: invert
iconname²	(String; only for <code>type=Text</code> , Stamp, FileAttachment) Name of an icon for displaying the annotation (to create an annotation without any visible icon set <code>opacity=0</code>): For <code>type=Text</code> (default: note): checkmark, circle, comment, cross, crosshairs, help, insert, key, newparagraph, note, paragraph, rightarrow, rightpointer, star, uparrow, upleftarrow For <code>type=Stamp</code> (default: draft; arbitrary stamp text can be created with the contents option): approved, asis, confidential, departmental, draft, experimental, expired, final, forcomment, forpublicrelease, notapproved, notforpublicrelease, sold, topsecret, For <code>type=FileAttachment</code> (default: pushpin): graph, paperclip, pushpin, tag
inreplyto	(Hypertext string; PDF 1.5; only for markup annotations; required if <code>replyto</code> is supplied) Name of the annotation (see option <code>name</code>) that this annotation is in reply to. Both annotations must be on the same page. The relationship between the two annotations can be specified with the <code>replyto</code> option.

Table 12.3 Options for `PDF_create_annotation()`

option	description
interiorcolor	(Color; only for type=Line, Polygon, PolyLine, Square, Circle; PDF/A color restrictions apply, see above table) The color for the annotation's interior area. Supported color spaces are none, gray, rgb, and (in PDF 1.6) cmyk. Default: none
lang	(String; PDF 2.0) Natural language of the annotation in the format described in Table 3.3 for the lang option. The format of the language specification is identical to the lang option of <code>PDF_begin_document()</code> (see Table 3.3).
layer	(Layer handle; PDF 1.5) Layer to which the annotation belongs. The annotation is only visible if the corresponding layer is visible.
linewidth	(Float) Width of the annotation border or the line of the annotation types Line, PolyLine, Polygon, Square, Circle, Ink in default units. If linewidth=0 the border is invisible. Default: 1
locked	(Boolean) If true, the annotation properties (e.g. position and size) cannot be edited. However, the contents can still be modified. Default: false
locked-contents	(Boolean; PDF 1.7) If true, the annotation contents cannot be edited. However, the annotation can still be deleted and its properties changed (e.g. position and size). Default: false
name	(Hypertext string; required for type=Screen) Name of the annotation. It must be unique among all annotations on the page. The name is required if the annotation is used in the following situations: <ul style="list-style-type: none"> ▶ <code>PDF_create_annotation()</code>: options <code>inreplyto</code>, <code>parentname</code> ▶ <code>PDF_create_action()</code> with type=Hide: option <code>namelist</code> ▶ <code>PDF_create_action()</code> with type=RichMediaExecute, GoTo3DView or Rendition: option <code>target</code> ▶ <code>PDF_create_action()</code> with type=GoToE: option <code>targetpath</code> with suboption <code>annotation</code> This option is recommended if you intend to digitally sign the document.
opacity	(Float or percentage; not for PDF/A-1; should not be used in PDF/VT-1; up to PDF 1.7ext8 only for markup annotations; in PDF 2.0 for all annotation types) Opacity (0-1 or 0%-100%) for stroke and (in the absence of <code>opacityfill</code>) fill operations when painting the annotation. Default: 1
opacityfill	(Float or percentage; PDF 2.0) Opacity for fill operations when painting the annotation. Default: 1
open	(Boolean; only for type=Text, Popup) If true, the annotation is initially open. Default: false
orientate	(Keyword; only for type=FreeText, Stamp) Orientation of the annotation within its rectangle. Supported keywords: north (upright), east (pointing to the right), south (upside down), west (pointing to the left). Default: north
parentname	(String; only for type=Popup and required in this case) Name of the parent annotation for the Popup annotation. This name must refer to a markup annotation with a type other than FreeText. The options contents, <code>annotcolor</code> and <code>title</code> of the popup annotation are inherited from the parent annotation.
polylinelist¹	(List containing one or more polylines or quadrilaterals; only for type=Polygon, PolyLine, Ink, Highlight, Underline, Squiggly, Strikeout). Default: a polyline connecting the vertices of the annotation rectangle. For type=Polygon, PolyLine, Ink: A list containing a polyline consisting of one or more points. Other types: The list contains n sublists with 8 float values each, specifying n quadrilaterals. Each quadrilateral encompasses a word or group of contiguous words in the text underlying the annotation. The quadrilaterals must be provided in zigzag order (top right, top left, lower right, lower left).
readonly	(Boolean) If true, do not allow the annotation to interact with the user. The annotation may be displayed or printed, but should not respond to mouse clicks or change its appearance in response to mouse motions. Default: false
rectdiff	(Rectangle; only for type=Caret) Four non-negative numbers which describe the difference between the annotation rectangle and the boundaries of the underlying caret.

Table 12.3 Options for `PDF_create_annotation()`

option	description
replyto	(Keyword; PDF 1.6; only for markup annotations) Relationship (the reply type) between this annotation and the annotation specified by the <code>inreplyto</code> option (default: <code>reply</code>): reply The annotation must be considered a reply to the annotation specified by <code>inreplyto</code> . group The annotation must be grouped with the annotation specified by <code>inreplyto</code> .
richmedia	(Option list; only for <code>type=RichMedia</code> , and required in this case) Rich media options according to Table 13.5
rotate	(Boolean; must not be set to <code>true</code> for text annotations in PDF/A) If <code>true</code> , rotate the annotation to match the rotation of the page. Otherwise the annotation's rotation remains fixed. This option is ignored for icons of text annotations. Default: <code>false</code> for text annotations in PDF/A, otherwise <code>true</code>
state	(String; only for <code>type=Text</code> ; PDF 1.5) State to which the annotation specified in the <code>inreplyto</code> option will be set. Keywords for <code>statemodel=Marked</code> (default: <code>Unmarked</code>): <code>Marked</code> , <code>Unmarked</code> Keywords for <code>statemodel=Review</code> (default: <code>None</code>): <code>Accepted</code> , <code>Rejected</code> , <code>Cancelled</code> , <code>Completed</code> , <code>None</code>
statemodel	(String; required if the state option is supplied; only for <code>type=Text</code> ; PDF 1.5). State model: Marked Supports the state values <code>Marked</code> , <code>Unmarked</code> Review Supports the state values <code>Accepted</code> , <code>Rejected</code> , <code>Cancelled</code> , <code>Completed</code> , <code>None</code>
subject	(Hypertext string; PDF 1.5) Short description of the subject addressed by the annotation
symbol	(Keyword; only for <code>type=Caret</code>) Type of the caret symbol (default: <code>none</code>): paragraph paragraph symbol none no symbol
template²	(Option list; not for <code>type=Popup</code>) Visual appearance of the annotation for one or more states: normal/rollover/down (Template handle; <code>rollover</code> and <code>down</code> are not allowed in PDF/A-3 mode) Template for the annotation's normal, mouse rollover, or mouse button down appearance. . Default for <code>normal</code> : internally generated appearance; default for <code>rollover</code> and <code>down</code> : no appearance fitmethod (Keyword) Method to fit the template into the annotation rectangle. If <code>fitmethod</code> is different from <code>entire</code> the annotation rectangle is adapted to the template box (default: <code>entire</code>): nofit Position the template only, without any scaling or clipping. meet Position the template according to the <code>position</code> option and scale it so that it entirely fits into the rectangle while preserving its aspect ratio. Generally at least two edges of the template will meet the corresponding edges of the rectangle. entire Position the template according to the <code>position</code> option and scale it so that it entirely covers the rectangle. Generally this method will distort the template. position (List of floats or keywords) One or two values specifying the position of the reference point (x, y) within the template with {0 0} being the lower left corner of the template and {100 100} the upper right corner. The values are expressed as percentages of the template's width and height. If both percentages are equal it is sufficient to specify a single float value. The keywords <code>left</code> , <code>center</code> , <code>right</code> (in x direction) or <code>bottom</code> , <code>center</code> , <code>top</code> (in y direction) can be used as equivalents for the values 0, 50, and 100. If only one keyword has been specified the corresponding keyword for the other direction will be added. Default: { <code>left bottom</code> }.
title	(Hypertext string; not for <code>type=Popup</code>) Label in the title bar of the annotation's popup window when open and active. This string corresponds to the <code>Author</code> field in Acrobat. The maximum length is 255 single-byte characters or 126 Unicode characters. However, a practical limit of 32 characters is advised. Default: <code>none</code>

Table 12.3 Options for `PDF_create_annotation()`

option	description
usematchbox	(List of strings; cannot be combined with the <code>tag</code> option if the matchbox has been created with a structure element) The <code>llx/llx/urx/ury</code> parameters are ignored and the named matchbox is used instead. The first element in the option list is a name string which specifies a matchbox. The second element is either an integer specifying the number of the desired rectangle (starting with 1), or the keyword <code>all</code> to specify all rectangles of the selected matchbox. If the second element is missing, it defaults to <code>all</code> . If multiple matchbox rectangles are selected a separate annotation is created for each rectangle. If the matchbox or the specified rectangle does not exist on the page, the method silently returns without creating an annotation. When using matchboxes to create annotations in a table cell <code>PDF_create_annotation()</code> must be called after <code>PDF_fit_table()</code> .
user-coordinates	(Boolean) If false, annotation coordinates and font size are expected in the default coordinate system; otherwise the current user coordinate system is used. Default: the value of the global <code>usercoordinates</code> option
zoom	(Boolean; must not be set to true for text annotations in PDF/A) If true, scale the annotation to match the magnification of the page. Otherwise the annotation's size remains fixed. This option is ignored for icons of text annotations. Default: false for text annotations in PDF/A, otherwise true

1. The coordinates are interpreted in default coordinates (if the `usercoordinates` option is false) or user coordinates (if it is true).
2. At most one of the options `createrichtext`, `contents`, `iconname` or `template` can be specified for `type=Stamp`.

Table 12.4 Additional options for `PDF_create_annotation()` for `type=Line`

option	description
captionoffset	(2 Floats; PDF 1.7) Offset of the caption text from its normal position. The first value specifies the horizontal offset along the annotation line from its midpoint, with a positive value indicating offset to the right and a negative value indicating offset to the left. The second value specifies the vertical offset perpendicular to the annotation line, with a positive value indicating a shift up and a negative value indicating a shift down. Default: {0, 0}, i.e. no offset from the normal position
caption-position	(Keyword; PDF 1.7; ignored if <code>showcaption=false</code>) The annotation's caption position (default: <code>Inline</code>): Inline The caption will be centered inside the line. Top The caption will be positioned on top of the line.
leaderlength	(List with one or two floats; the second float must not be negative; PDF 1.6) The length of leader lines in default coordinates (if the <code>usercoordinates</code> option is false) or user coordinates (if it is true). Leader lines are auxiliary lines which are drawn from each endpoint of the line perpendicular to the line itself. The length is specified by two numbers (default: {0 0}): The first number is the extension from each endpoint of the line perpendicular to the line itself. A positive value means that the leader lines appear in the direction that is clockwise when traversing the line from its start point to its end point (as specified by the <code>line</code> option); a negative value indicates the opposite direction. The second value, which can be omitted, represents the length of leader line extension which is drawn on the opposite side of the line. A positive value is ignored if the first value is 0.
leaderoffset	(Non-negative float; PDF 1.7) Length of the leader line offset, which is the amount of space between the endpoints of the annotation and the beginning of the leader lines. Default: 0
line¹	(Line; required; in PDF/X mode the line must be located completely inside the annotation rectangle) List with four coordinates specifying the start and end points of the line.
showcaption	(Boolean; PDF 1.6) If true, the text specified in the <code>contents</code> or <code>createrichtext</code> options is replicated as a caption in the appearance of the line. Default: false

1. The coordinates are interpreted in default coordinates (if the `usercoordinates` option is false) or user coordinates (if it is true).

12.3 Form Fields

++ Java C# `void create_field(double llx, double lly, double urx, double ury, String name, String type, String optlist)`

Perl PHP `create_field(float llx, float lly, float urx, float ury, string name, string type, string optlist)`

C `void PDF_create_field(PDF *p, double llx, double lly, double urx, double ury, const char *name, int len, const char *type, const char *optlist)`

Create a new form field or fill an imported form field.

llx, lly, urx, ury x and y coordinates of the lower left and upper right corners of the field rectangle in default coordinates (if the `usercoordinates` option is `false`) or user coordinates (if it is `true`). The edges of PDF form fields are always parallel to the page edges.

Note that form field coordinates are different from the parameters of the `PDF_rect()` method. While `PDF_create_field()` expects coordinates for two corners directly, `PDF_rect()` expects the coordinates of one corner, plus width and height values.

name (Hypertext string) The form field name, possibly prefixed with the name(s) of one or more groups which have been created with `PDF_create_fieldgroup()`. Group names must be separated from each other and from the field name by a period `».«` character. Field names must be unique in a document and must not end in a period `».«` character.

len (C language binding only) Length of `text` (in bytes). If `len = 0` a null-terminated string must be provided.

type The field type according to Table 12.5.

Table 12.5 Form field types

type	icon	Options specific for this type (in addition to general options)
checkbox		buttonstyle, currentvalue, itemname
combobox		commitonselect, charspacing, currentvalue, editable, itemnamelist, itemtextlist, sorted, spellcheck
listbox		charspacing, commitonselect, currentvalue, itemnamelist, itemtextlist, multiselect, sorted, topindex
pushbutton		buttonlayout, caption, captiondown, captionrollover, charspacing, fitmethod, icon, icondown, iconrollover, position, submitname
radiobutton		buttonstyle, currentvalue, itemname, toggle, unisonselect <i>Requires a group with fieldtype=radiobutton.</i>
signature		fieldcontent, lockmode
textfield		comb, charspacing, currentvalue, fileselect, maxchar, multiline, password, richtext, scrollable, spellcheck
		<i>(Deprecated and not available in PDF 2.0) Text fields are also used for barcodes: barcode</i>

optlist An option list specifying field properties:

- ▶ Options for field properties according to Table 12.6. The following options are supported for all field types:
action, alignment, backgroundcolor, barcode, bordercolor, borderstyle, calcorder, dasharray, defaultvalue, display, exportable, fieldtype, fillcolor, font, fontsize, highlight, layer, linewidth, locked, orientate, readonly, required, strokecolor, taborder, tooltip, user-coordinates
- ▶ The options listed in Table 12.5 are supported for specific field types. They are also detailed in Table 12.6.
- ▶ (Not for *PDF_create_fieldgroup()* and not in form filling mode) Option for abbreviated structure element tagging according to Table 14.2: *tag*
- ▶ Encoding options for C and Perl, PHP, Ruby with *stringformat=legacy*:
hypertextencoding, hypertextformat (see Table 2.1)

Details The tab order of the fields on the page (the order in which they receive the focus when the tab key is pressed) is determined by the order of calls to *PDF_create_field()* by default, but a different order can be specified with the *taborder* option. The tab order cannot be modified after creating the fields. However, this behavior can be overridden with the *taborder* option of *PDF_begin/end_page_ext()*.

In Acrobat it is possible to assign a format (number, percentage, etc.) to text fields. However, this is not specified in the PDF Reference, but implemented with custom JavaScript. You can achieve the same effect by attaching a JavaScript action to the field which refers to the predefined JavaScript functions in Acrobat.

In Tagged PDF mode this method automatically creates a suitable *OBJR* element for the generated form field. The user must create the corresponding *Form* container element (see PDFlib Tutorial) before calling this method.

Fonts for use in form fields must be selected with care to work around restrictions in Acrobat. Some recommendations:

- ▶ The font should be installed on the viewing machine.
- ▶ TrueType and OpenType fonts should be loaded with the option *nosubsetting*. The option *encoding=builtin* can be used for symbol fonts.

Filling imported form fields. As an alternative to creating form fields this method can be used to modify the appearance and contents of form field which have been imported with *PDF_fit_pdi_page()* and the *usefields* option. Form filling mode works as follows:

- ▶ The *name* parameter must contain the fully qualified name of an imported field. It is an error to supply the name of a non-existing field. Field names in imported documents can be queried with pCOS pseudo object *pages[...]/fields[...]/fullname*.
- ▶ The *type* parameter is ignored; field types in imported documents can be queried with pCOS pseudo object *pages[...]/fields[...]/type*.
- ▶ The field coordinates *llx/lly/urx/ury* are ignored.
- ▶ Most options applicable to the respective field type can be supplied. The default for all options changes to »original field property remains unchanged«. Most importantly, the field contents can be changed with the option *currentvalue*.
- ▶ Some options are not allowed (see Table 12.6).

PDF/A Some options are restricted; see Table 12.6 for details. The options *backgroundcolor, bordercolor, fillcolor, and strokecolor* are subject to the following conditions: *RGB* colors are always allowed, *Grayscale* colors are only allowed with an output intent (any type), and *CMYK* colors are only allowed with a *CMYK* output intent.

PDF/UA A structure element of type *Form* must be created with `PDF_begin_item()` or the *tag* option when this method is called. The *tooltip* option is required.

PDF/X Form fields are only allowed if they are positioned completely outside of the *BleedBox* (or *TrimBox*/*ArtBox* if no *BleedBox* is present).

Scope *page*; if a rendition action is supplied which has been created with the *target* option this method can only be called on the page where the target annotation is located.

In form filling mode, i.e. if form fields have been imported from another document with the *usefields* option of `PDF_fit_pdi_page()` this method cannot be used to create new fields, but only to fill imported fields. This method can only be called on the page where `PDF_fit_pdi_page()` has been called to place the page containing the imported fields.

Table 12.6 Options for field properties with `PDF_create_field()` and `PDF_create_fieldgroup()`

option	description
action	(Action list; not allowed for PDF/A) List of field actions for one or more of the following events. The activate event is allowed for all field types, the other events are not allowed for type=pushbutton, checkbox, radiobutton. Default: empty list
activate	Actions to be performed when the field is activated.
blur	Actions to be performed when the field loses the input focus.
calculate	JavaScript actions to be performed in order to recalculate the value of this field when the value of another field changes.
close	(PDF 1.5) Actions to be performed when the page containing the field is closed.
down	Actions to be performed when the mouse button is pressed inside the field's area.
enter	Actions to be performed when the mouse enters the field's area.
exit	Actions to be performed when the mouse exits the field's area.
focus	Actions to be performed when the field receives the input focus.
format	JavaScript actions to be performed before the field is formatted to display its current value. This allows the field's value to be modified before formatting.
invisible	(PDF 1.5) Actions to be performed when the page containing the field is no longer visible.
keystroke	JavaScript actions to be performed when the user types into a text field or combo box, or modifies the selection in a scrollable list box.
open	(PDF 1.5) Actions to be performed when the page containing the field is opened.
up	Actions to be performed when the mouse button is released inside the field's area (this is typically used to activate a field).
validate	JavaScript actions to be performed when the field's value is changed. This allows the new value to be checked for validity.
visible	(PDF 1.5) Actions to be performed when the page containing the field becomes visible.
alignment	(Keyword) Alignment of text in the field: left, center, right. Default: left
background-color bordercolor	(Color; subject to PDF/A color restrictions) Color of the field background or border. Supported color spaces: none, gray, rgb, cmyk. Default: none
barcode	(Option list; only for type=textfield; implies readonly; PDF 1.7ext3; deprecated and not available in PDF 2.0) Create a barcode field according to the options in Table 12.7. The field should provide the action option with a calculate event script which determines the barcode contents based on the contents of other fields or supplies a static value: action={calculate=...}. Barcode fields work only in the full version of Acrobat, but neither in Acrobat Reader nor third-party PDF viewers.
borderstyle	(Keyword) Style of the field border: solid, beveled, dashed, inset, underline. Default: solid

Table 12.6 Options for field properties with `PDF_create_field()` and `PDF_create_fieldgroup()`

option	description
button-layout	(Keyword; only for type=pushbutton) The position of the button caption relative to the button icon, provided both have been specified: below, above, right, left, overlaid. Default: right
buttonstyle	(Keyword; only for type=radiobutton and checkbox) Symbol to be used for the field: check, cross, diamond, circle, star, square. Default: check
calcorder	(Integer; only used if the field has a JavaScript action for the calculate event) Calculation order of the field relative to other fields. Fields with smaller numbers will be calculated before fields with higher numbers. Default: 10 plus the maximum calcorder used on the current page (and 10 initially)
caption	(Content string; only for type=pushbutton; one of the caption or icon options must be supplied for push buttons) Caption text which is visible when the button doesn't have input focus. An empty string (i.e. caption {}) produces neither caption nor icon. Default: none
caption-down	(Content string; only for type=pushbutton; requires highlight=push; not allowed for PDF/A) Caption text which is visible when the button is activated. Default: none
caption-rollover	(Content string; only for type=pushbutton; requires highlight=push; not allowed for PDF/A) Caption text which is visible when the button has input focus. Default: none
charspacing	(Float; not for type=radiobutton, checkbox, signature) Character spacing for text in the field in units of the current user coordinate system. Default: 0
comb	(Boolean; only for type=textfield; PDF 1.5) If true and the multiline, fileselect, and password options are false, and the maxchar option has been supplied with an integer value, the field will be divided into a number of equidistant subfields (according to the maxchar option) for individual characters. Default: false
commit-onselect	(Boolean; only for type=listbox, combobox; PDF 1.5) If true, an item selected in the list will be committed immediately upon selection. If false, the item will only be committed upon exiting the field. Default: false
currentvalue	(Not for type=pushbutton, signature) The field's initial value. Type and default depend on the field type: checkbox, radiobutton (String) The string Off means that the button is deactivated. An arbitrary string other than Off means that the button is activated. This option should be set for the first button. Default: Off combobox (Content string) Selected value in the combobox. It must be one of the strings supplied in the option itemtextlist. Default: empty listbox (List of integers) Zero-based index of the selected item within itemtextlist (before sorting if the sorted option is supplied). Default: none textfield (Content string) Contents of the text field. Default: empty
dasharray	(List of two non-negative floats; only for borderstyle=dashed). The lengths of dashes and gaps for a dashed border in default units (see Table 7.1). Default: 3 3
defaultvalue	The field's value after a ResetForm action. Types and defaults are the same as for the currentvalue option. Exception: for listboxes only a single integer value is allowed.
display	(Keyword; forced to visible for PDF/A) Visibility on screen and paper: visible, hidden, noview, noprint. Default: visible
editable	(Boolean; only for type=combobox) If true, the currently selected text in the box can be edited. Default: false
exportable	(Boolean) The field will be exported when a SubmitForm action happens. Default: true
fieldcontent	(Template handle; only for type=signature) The supplied template is displayed in the signature field; it may contain signing instructions. The template must have the same width and height as the field.

Table 12.6 Options for field properties with `PDF_create_field()` and `PDF_create_fieldgroup()`

option	description
fieldtype	(Keyword; only for <code>PDF_create_fieldgroup()</code>) Type of the fields in the group (default: mixed): radiobutton <i>The group contains only radio buttons. The option <code>uniselect</code> must be supplied. The options <code>itemtextlist</code>, <code>itemnamelist</code>, <code>currentvalue</code> and <code>defaultvalue</code> must be specified in <code>PDF_create_fieldgroup()</code> and not in <code>PDF_create_field()</code>.</i> mixed <i>The group may contain an arbitrary mixture of field types.</i> pushbutton, checkbox, listbox, combobox, textfield <i>The group contains only fields of the specified type. The current value is displayed in all group fields simultaneously, even if the fields are located on separate pages. Common options, e.g. <code>currentvalue</code>, must be specified in <code>PDF_create_fieldgroup()</code> and not in <code>PDF_create_field()</code>.</i>
fileselect	(Boolean; only for <code>type=textfield</code>) If true, the text in the field will be treated as a file name. Default: false
fillcolor	(Color; subject to PDF/A color restrictions) Fill color for text. Supported color spaces: gray, rgb, cmyk. Default: {rgb 0 0 0} for PDF/A without output intent, otherwise {gray 0} (i.e. black)
fitmethod	(Keyword; only for <code>type=pushbutton</code>) Method of placing a template provided with the <code>icon</code> , <code>icondown</code> , and <code>iconrollover</code> options within the button. Supported keywords (default: meet): auto same as meet if the template fits into the button, otherwise clip nofit same as clip clip template will not be scaled, but clipped at the field border meet template will be scaled proportionally so that it fits into the button slice same as meet entire template will be scaled so that it fully fits into the button
font	(Font handle; required for <code>type=listbox</code> , <code>combobox</code> , and <code>textfield</code> , and for <code>type=pushbutton</code> if one or more of <code>caption</code> , <code>captionrollover</code> , or <code>captiondown</code> are specified; not for <code>type=checkbox</code> , <code>radiobutton</code> , <code>signature</code>) Font to be used for the field. Acrobat can display characters even if they are not included in the font's encoding. For example, you can use <code>encoding=winansi</code> and supply Unicode characters outside winansi.
fontsize	(FontSize; not for <code>type=signature</code>) Font size in user coordinates. The value 0 or keyword auto means that the fontsize is adjusted to the rectangle. Default: auto
highlight	(Keyword) Highlight mode of the field when the user clicks on it: none, invert, outline, push. Default: invert
icon	(Template handle ¹ ; only for <code>type=pushbutton</code> ; one of the <code>caption</code> or <code>icon</code> options must be supplied for push buttons) Handle for a template which will be visible when the button doesn't have input focus. Default: the appearance is created from the <code>caption</code> and <code>font</code> options
icondown	(Template handle ¹ ; only for <code>type=pushbutton</code> ; requires <code>highlight=push</code> ; not allowed for PDF/A) Handle for a template which will be visible when the button is activated. Default: the appearance is created from the <code>captiondown</code> and <code>font</code> options
iconrollover	(Template handle ¹ ; only for <code>type=pushbutton</code> ; requires <code>highlight=push</code> ; not allowed for PDF/A) Handle for a template which will be visible when the button has input focus. Default: the appearance is created from the <code>captionrollover</code> and <code>font</code> options
itemname	(Hypertext string; only for <code>type=radiobutton</code> and <code>checkbox</code> ; must be used if the export value is not a Latin 1 string) Export value of the field. Item names for multiple radio buttons in a group may be identical. Default: field name
item-namelist	(List of hypertext strings; only for <code>type=listbox</code> and <code>combobox</code>) Export values of the list items. Multiple items may have the same export value. Default: none

Table 12.6 Options for field properties with `PDF_create_field()` and `PDF_create_fieldgroup()`

option	description
itemtextlist	(List of content strings; only for type=listbox and combobox, and required in these cases; also required in form filling mode if itemnamelist is supplied) Text contents for all items in the list. If both itemnamelist and itemtextlist are specified both must contain the same number of strings.
layer	(Layer handle; PDF 1.5) Layer to which the field will belong. The field will only be visible if the corresponding layer is visible.
linewidth	(Float) Line width of the field border in default coordinates. Default: 1
locked	(Boolean) If true, the field properties cannot be edited in Acrobat. Default: false
lockmode	(Keyword; only for type=signature; PDF 1.5) Indicates the set of fields that should be locked when the field is signed: all All fields in the document will be locked.
maxchar	(Integer or keyword; only for type=textfield) The upper limit for the number of text characters in the field, or the keyword unlimited if there is no limit. Default: unlimited
multiline	(Boolean; only for type=textfield) If true, text will be wrapped to multiple lines if required. Default: false
multiselect	(Boolean; only for type=listbox) If true, multiple items in the list can be selected. Default: false
orientate	(Keyword) Orientation of the contents within the field: north, west, south, east. Default: north
password	(Boolean; only for type=textfield) If true, the text will be not be visible upon input. No currentvalue should be supplied for password fields. Default: false
position	(List of floats or keywords; only for type=pushbutton) One or two values specifying the position of a template provided with the icon/icondown/iconrollover options within the field rectangle, with {0 0} being the lower left corner of the field, and {100 100} the upper right corner. The values are expressed as percentages of the field rectangle's width and height. If both percentages are equal it is sufficient to specify a single float value. The keywords left, center, right (in x direction) or bottom, center, top (in y direction) can be used as equivalents for the values 0, 50, and 100. If only one keyword has been specified, the corresponding keyword for the other direction will be added. Default: {center}. Examples: {0 50} or {left center} left-justified template {50 50} or {center} centered template {100 50} or {right center} right-justified template
readonly²	(Boolean) If true, the field does not allow any input. Default: false
required	(Boolean) If true, the field must contain a value when the form is submitted. Default: false
richtext	(Boolean; only for type=textfield; PDF 1.5) Allow rich text formatting. If true, the fontsize must not be 0, and fillcolor must not use color space cmyk. Default: false
scrollable	(Boolean; only for type=textfield) If true, text will be moved to the invisible area outside the field if the text doesn't fit into the field. If false, no more input will be accepted when the text fills the full field. Default: true
sorted	(Boolean; only for type=listbox and combobox) If true, the contents of the list will be sorted. Default: false
spellcheck	(Boolean; only for type=textfield and combobox) If true, the spell checker will be active in the field. Default: false if type=textfield and password=true, otherwise true
strokecolor	(Color; subject to PDF/A color restrictions) Stroke color for text. Supported color spaces: gray, rgb, cmyk. Default: {rgb 0 0 0} for PDF/A without output intent, otherwise {gray 0} (i.e. black).

Table 12.6 Options for field properties with `PDF_create_field()` and `PDF_create_fieldgroup()`

option	description
submitname	(Hypertext string; recommended only for <code>type=pushbutton</code>) Mapping name to be used when exporting form field data.
taborder	(Integer; not allowed in form filling mode) Tab order of the field relative to other fields. Fields with smaller numbers will be reached before fields with higher numbers. Default: 10 plus the maximum <code>taborder</code> used on the current page (and 10 for the first field on the page); the result of this default is that the tab order is determined by the creation order.
toggle	(Boolean; only for <code>PDF_create_fieldgroup()</code> and <code>type=radiobutton</code>) If true, a radio button within a group can be activated and deactivated by clicking. If false, it can only be activated by clicking, and deactivating by clicking another button. Default: false
tooltip²	(Hypertext string; non-empty value required in PDF/UA-1) The text visible in the field's tooltip, also used by screen readers. For radio buttons and groups Acrobat uses the tooltip of the first button in the group, others are ignored. Default: none
topindex	(Integer; only for <code>type=listbox</code>) Index of the first visible entry. The first item has index 0. Default: 0
unisonselect	(Boolean; only for <code>PDF_create_fieldgroup()</code> , <code>type=radiobutton</code> and PDF 1.5) If true, radio buttons with the same field name or item name will be selected simultaneously. Default: false
user-coordinates	(Boolean; not allowed in form filling mode) If false, field coordinates are expected in the default coordinate system; otherwise the current user coordinate system will be used. Default: the value of the <code>usercoordinates</code> global option

1. Templates for icons can be created with the `PDF_begin_template_ext()` method; if the icon consists of an image only you can create the template by supplying the `template` option to `PDF_load_image()`.

2. For `type=radiobutton` this option should not be used with `PDF_create_field()`, but only with `PDF_create_fieldgroup()`.

C++ Java C# **void create_fieldgroup(String name, String optlist)**

Perl PHP **create_fieldgroup(string name, string optlist)**

C **void PDF_create_fieldgroup(PDF *p, const char *name, int len, const char *optlist)**

Create a form field group subject to various options.

name (Hypertext string) The name of the form field group, which may in turn be prefixed with the name of another group. Field groups can be nested to an arbitrary level. Group names must be separated with a period `».«` character. Group names must be unique within the document, and must not end in a period `».«` character.

len (C language binding only) Length of `text` (in bytes). If `len = 0` a null-terminated string must be provided.

optlist An option list with field options for `PDF_create_field()`

Details Field groups are useful for the following purposes depending on the `fieldtype` option:

- ▶ A group with `fieldtype=radiobutton` is required as container for radio buttons. For all other field types group membership is optional.
- ▶ Arbitrary fields with the same or different types can be assembled in a logical group with `fieldtype=mixed`. The names of all fields in the group start with the group name, e.g. `name.prefix`, `name.first`, `name.initial`, `name.last`.
- ▶ All other values of `fieldtype` can be used to create synchronized fields, i.e. one or more group fields of the same type which display the same value.

If the name of a field group (e.g. *button*) is provided as prefix for a field name created with *PDF_create_field()* (e.g. *button.1*), the new field becomes part of this group. Field options supplied in the *optlist* for a group are inherited by all fields in the group.

PDF/A See *PDF_create_field()*.

PDF/UA See *PDF_create_field()*.

Scope Any except *object*

If form fields have been imported from another document with the *usefields* option of *PDF_fit_pdi_page()* this method cannot be used.

Table 12.7 Suboptions for the barcode option of *PDF_create_field()* and *PDF_create_fieldgroup()*

option	description
caption	(Hypertext string) Caption which will be rendered below the barcode. By default, Acrobat creates the file: URL for the document as caption.
dataprep	(Integer) Applicable data preparation. Supported values (default: 0): 0 Do not apply any compression before encoding the data in the barcode. 1 Compress the data with the Flate compression algorithm before encoding the data.
ecc	(Integer; required for <i>symbology=PDF417</i> and <i>QRCode</i>) Error correction coefficient where higher values create better error correction through redundancy, but require a larger barcode. For <i>symbology=PDF417</i> the values must be in the range 0-8; for <i>symbology=QRCode</i> the values must be in the range 0-3.
resolution	(Positive integer) Resolution in dpi at which the barcode is rendered (default: 300)
symbology	(Keyword; required) Barcode technology to use: PDF417 PDF417 bar code according to ISO 15438 QRCode QR Code 2005 bar code according to ISO 18004 DataMatrix Data Matrix bar code according to ISO 16022
xsymheight	(Integer; only for <i>symbology=PDF417</i> , and required in this case) Vertical distance between two barcode modules in pixels. The ratio <i>xsymheight/xsymwidth</i> must be an integer value. The allowed range for this ratio is 1-4.
xsymwidth	(Integer; required) Horizontal distance between two barcode modules in pixels

12.4 Actions

C++ Java C# *int create_action(String type, String optlist)*

Perl PHP *int create_action(string type, string optlist)*

C *int PDF_create_action(PDF *p, const char *type, const char *optlist)*

Create an action which can be applied to various objects and events.

type Action type according to Table 12.8.

Table 12.8 Action types

type	description; options allowed for this type
GoTo	Go to a destination in the current document: destination, destname
GoTo3DView	(PDF 1.6) Set the current view of a 3D model: 3Dview, target
GoToE	(PDF 1.6) Go to a destination in an embedded document: destination, destname, filename, newwindow, targetpath
GoToR	Go to a destination in another (remote) document: destination, destname, filename, newwindow, remotestructdest
Hide	Hide or show a form field: hide, namelist
ImportData	Import form field values from a file: filename
JavaScript	Execute a script with JavaScript code: javascript, script, scriptname
Launch	Launch an application or document: filename, newwindow
Named	Execute an Acrobat menu item identified by its name: menuname
Rendition	(PDF 1.5) Control playback of multimedia content in a Screen annotation: javascript, operation, rendition, script, target
ResetForm	Set some or all form fields to their default values: exclude
RichMedia-Execute	(PDF 1.7ext3) Send a command to a RichMedia annotation: functionname, instance, richmediaargs, target
SetOCGState	(PDF 1.5) Hide or show layers: layerstate, preserveradio
SubmitForm	Send data to a uniform resource locator, i.e. an Internet address: canonicaldate, exclude, export-method, submitemptyfields, url
Trans	(PDF 1.5) Update the display using some visual effect. This can be useful to control the display during a sequence of multiple actions: duration, transition
URI	Resolve a uniform resource identifier, i.e. jump to an Internet address: ismap, url

optlist An option list specifying properties of the action:

- ▶ General options: *errorpolicy* (see Table 1.5)
- ▶ The following type-specific options according to Table 12.9:
3Dview, richmediaargs, canonicaldate, destination, destname, duration, exclude, export-method, filename, functionname, hide, instance, ismap, javascript, layerstate, menuname, namelist, newwindow, operation, preserveradio, rendition, remotestructdest, script, scriptname, submitemptyfields, target, targetpath, transition, url

- ▶ Encoding option for C and Perl, PHP, Ruby with `stringformat=legacy: hypertextencoding` (see Table 2.1)

Returns An action handle which can be used to attach actions to objects within the document. The action handle can be used until the end of the enclosing *document* scope.

Details This method creates a single action. Various objects (e.g. pages, form field events, bookmarks) can be provided with one or more actions, but each action must be generated with a separate call to `PDF.create_action()`. Using an action multiply for different objects is allowed. It is recommended to re-use existing handles if an action with the same options has already been created earlier.

PDF/A Only the following action types are allowed:
`GoTo`, `GoToE`, `GoToR`, `Named`, `SubmitForm`, `URI`

PDF/UA The option `ismap=true` is not allowed.

PDF/X This method must not be called.

Scope Any except *object*; if `type=Rendition` and the `target` option is supplied this method is only allowed in *page* scope and must be called on the page where the target annotation is located. The returned handle can be used until the next call to `PDF.end_document()`.

Table 12.9 Options for `PDF.create_action()`

option	description
3Dview	(Keyword or 3D view handle; <code>GoTo3DView</code> ; required) Selects the view of a 3D annotation; One of the keywords <code>first</code> , <code>last</code> , <code>next</code> , <code>previous</code> (referring to the respective entries in the annotation's <code>views</code> option), or <code>default</code> (referring to the annotation's <code>defaultview</code> option), or a 3D view handle created with <code>PDF.create_3dview()</code> .
canonical-date	(Boolean; <code>SubmitForm</code>) If <code>true</code> , any submitted field values representing dates are converted to a standard format. The interpretation of a field as a date is not specified explicitly in the field itself, but only in the JavaScript code that processes it. Default: <code>false</code>
destination	(Option list; <code>GoTo</code> , <code>GoToE</code> , <code>GoToR</code> ; required unless <code>destname</code> is supplied) Option list according to Table 12.11 defining the destination to jump to. The <code>structdest</code> suboption is only allowed for <code>type=GoTo</code> .
destname	(Hypertext string; required unless <code>destination</code> is supplied) <code>GoTo</code> : name of a destination which has been defined with <code>PDF.add_nameddest()</code> . The destination can be created before or after referring to it. Named structure destinations are only allowed for <code>type=GoTo</code> . <code>GoToR</code> , <code>GoToE</code> : name of a destination in the remote or embedded document.
duration	(Float; <code>Trans</code>) Set the duration of the transition effect in seconds for the current page. Default: <code>1</code>
exclude	(Boolean) <code>SubmitForm</code> : If <code>true</code> , the <code>namelist</code> option specifies which fields to exclude; all fields in the document are submitted except those listed in the <code>namelist</code> array and those whose <code>exportable</code> option is <code>false</code> . If <code>false</code> , the <code>namelist</code> option specifies which fields to include in the submission. All members of specified field groups will be submitted as well. Default: <code>false</code> <code>ResetForm</code> : If <code>true</code> , the <code>namelist</code> option specifies which fields to exclude; all fields in the document are reset except those listed in the <code>namelist</code> array. If <code>false</code> , the <code>namelist</code> option specifies which fields to include in resetting. All members of specified field groups will be reset as well. Default: <code>false</code>

Table 12.9 Options for `PDF_create_action()`

option	description
export-method	<p>(Keyword list; SubmitForm) Controls how the field names and values are submitted (default: fdf)</p> <p>html, fdf, xfdf, pdf In HTML, FDF, XFDF, or PDF format, respectively</p> <p>annotfields (Only for fdf) Include all annotations and fields.</p> <p>coordinate (Only for html) The coordinates of the mouse click that caused the submitform action will be transmitted as part of the form data. The coordinate values are relative to the upper-left corner of the field's rectangle.</p> <p>exclurl (Only for fdf) The submitted FDF will exclude the url string.</p> <p>getrequest (Only for html and pdf) Submit using HTTP GET; otherwise HTTP POST</p> <p>onlyuser (Only for fdf and annotfields) The submit will include only those annotations whose name matches the name of the current user, as determined by the remote server.</p> <p>updates (Only for fdf) Include all incremental updates contained in the underlying PDF document</p> <p>Example for combined options: <code>exportmethod={fdf updates onlyuser}</code></p>
filename	<p>(Hypertext string) GoToR, Launch (required): name of an external (PDF or other) file or application which will be opened when the action is triggered. UNC file names must be written as <code>\\server\volume</code>.</p> <p>ImportData (required): name of the external file containing forms data.</p> <p>GoToE: name of the root document of the target relative to the root document of the source. If this entry is absent, the source and target share the same root document.</p>
functionname	(Hypertext string; RichMediaExecute; required) String specifying a JavaScript method name (not a full script).
hide	(Boolean; Hide) Indicates whether to hide (true) or show (false) fields. Default: true
instance	(Integer; RichMediaExecute) Index of the option list (starting with 1) in the instances suboption of the configuration suboption of the richmedia option of <code>PDF_create_annotation()</code> to specify an instance of the RichMedia annotation for which to execute the script. Default: 1
ismap	(Boolean; URI; true not allowed for PDF/UA-1) If true, the coordinates of the mouse position will be added to the target URI when the url is resolved. Default: false
javascript	(Asset handle which has been loaded with type=JavaScript; JavaScript, Rendition; only one of script and javascript can be supplied) JavaScript for the action
layerstate	<p>(Option list; SetOCGState; required) List of pairs with a keyword and a layer handle:</p> <p>on Activate the layer</p> <p>off Deactivate the layer</p> <p>toggle Reverse the state of the layer. If this is used <code>preserveradio</code> should be set to false.</p>
menuname	(String; Named; required) The name of the menu item to be performed. In PDF/A only <code>nextpage</code> , <code>prevpage</code> , <code>firstpage</code> , <code>lastpage</code> are allowed. Otherwise more names are accepted. Code for finding the names of other menu items can be found in the Cookbook topic <code>interactive/acrobat_menu_items</code> .
namelist	<p>(List of strings; Hide; required) The names (including group names) of the annotations or fields to be hidden or shown.</p> <p>(SubmitForm) The names (including group names) of form fields to include in the submission or which to exclude, depending on the setting of the <code>exclude</code> option. Default: all fields are submitted except those whose <code>exportable</code> option is false.</p> <p>(ResetForm) The names (including group names) of form fields to include in the resetting or which to exclude, depending on the setting of the <code>exclude</code> option. Default: all fields are reset.</p>
newwindow	(Boolean; GoToE, GoToR) A flag specifying whether to open the destination document in a new window. If this flag is false, the destination document will replace the current document in the same window. Launch: This entry is ignored if the file is not a PDF document. Default: Acrobat behaves according to the current user preference.

Table 12.9 Options for `PDF_create_action()`

option	description
operation	(Keyword; Rendition; <i>required unless javascript or script is supplied</i>) Operation to be applied to the rendition: play Play the rendition from the beginning. If another rendition is associated with the annotation it is stopped. stop Stop the rendition if it is playing or paused. pause Pause the rendition if it is playing; no effect for most streaming formats. resume Resume the rendition if it is paused. playfrombeginning If the same or another rendition is associated with the annotation and paused, resume it; otherwise play the supplied rendition from the beginning. This is faster than play if the rendition is paused.
preserve-radio	(Boolean; SetOCGState) If true, preserve the radio-button state relationship between layers. Default: true
rendition	(List of rendition asset handles; Rendition; <i>required if operation=play or playfrombeginning</i>) One or more renditions on which the JavaScript or operation is applied. If multiple renditions are supplied the PDF viewer selects the first which is supported.
remote-structdest	(String; only for Tagged PDF and type=GoToR; PDF 2.0) Identifier of the structure element in the remote document which serves as destination. In documents created with PDFlib this identifier must be assigned to the structure element with the id tagging option (not elementid which isn't written to the PDF). The user is responsible for making sure that the supplied id actually exists in the target document.
richmediaargs	(POCA container handle created with the option usage=richmediaargs; RichMediaExecute) Handle for an array container which specifies an arbitrary number of arguments for the command. Valid arguments are objects of type string, integer, float, or Boolean. Default: no arguments
script	(Hypertext string; JavaScript, Rendition; <i>exactly one of script and javascript can be supplied</i>) String containing JavaScript code for the action. Quoting (escaping) requirements for option list syntax must be observed, e.g. for the characters \ and { }. In order to pass arbitrary code the option list syntax described in »Unquoted string values«, page 10, may be useful. Quoting is not required when passing code with the alternative option javascript.
scriptname	(Hypertext string; JavaScript) If present, the JavaScript supplied in the script or javascript option is inserted as document-level JavaScript with the supplied name. If the same scriptname is supplied more than once in a document only the first script is used. Document-level JavaScript is executed after loading the document. This may be useful for scripts which are used in form fields.
submit-emptyfields	(Boolean; SubmitForm) If true, all fields characterized by the namelist and exclude options are submitted, regardless of whether they have a value. For fields without a value, only the field name is transmitted. If false, fields without a value are not submitted. Default: false
target	(String; GoTo3DView, Rendition, RichMediaExecute; <i>required; for rendition actions only required if operation is supplied and the action will be used in a method other than PDF_create_annotation() with type=Screen</i>) Name of the target annotation as specified in the name option of <code>PDF_create_annotation()</code> for which to execute the action. If a rendition action is supplied to a Screen annotation this annotation is implicitly used as target for the rendition action.
targetpath	(Option list; GoToE; <i>required unless filename is specified</i>) A target option list (see Table 12.10) specifying path information for the target document. Each target list specifies one element in the full path to the target and may have nested target option lists with additional elements.
transition	(Keyword; Trans) Set the transition effect; see Table 3.9 for a list of keywords. Default: replace

Table 12.9 Options for `PDF_create_action()`

option	description
<code>url</code>	(String; URI and <code>SubmitForm</code> ; required) A Uniform Resource Locator encoded in 7-bit ASCII or EBCDIC (but only containing ASCII characters) specifying the link target (for <code>type=URI</code>) or the address of the script at the Web server that will process the submission (for <code>type=SubmitForm</code>). It can point to an arbitrary (Web or local) resource and must start with a protocol identifier such as <code>http://</code> . Characters with special meaning in URLs (e.g. '%') must be quoted according to RFC 3986.

Table 12.10 Suboptions for the `targetpath` option of `PDF_create_action()`

option	description
<code>annotation</code>	(Hypertext string; required if <code>relation=child</code> and the target is associated with a file attachment annotation) Name of the target's file attachment annotation on the page specified by <code>pagenumber</code> or <code>destname</code> .
<code>destname</code>	(Hypertext string; required unless <code>pagenumber</code> is supplied and <code>relation=child</code> and the target is associated with a file attachment annotation; ignored if <code>pagenumber</code> is specified) Named destination for a page in the current document which contains the target's file attachment annotation.
<code>name</code>	(Hypertext string; required if <code>relation=child</code> and the target is located in the attachments list; otherwise it must be absent; ignored if <code>annotation</code> is specified) Name of the target in the attachments list of <code>PDF_begin/end_document()</code> .
<code>pagenumber</code>	(Integer; required unless <code>destname</code> is supplied and <code>relation=child</code> and the target is associated with a file attachment annotation; ignored if <code>destname</code> is specified) Number of a page in the current document which contains the target's file attachment annotation.
<code>relation</code>	(Keyword; required) Relationship of the current document and the target (which may be an intermediate target). Supported keywords: parent The target is the parent of the current document. child The target is a child of the current document.
<code>targetpath</code>	(Option list) A target option list according to Table 12.10 specifying additional path information to the target document. If this option is absent the current document is the target file containing the destination.

12.5 Named Destinations

C++ Java C# **void add_nameddest(String name, String optlist)**

Perl PHP **add_nameddest(string name, string optlist)**

C **void PDF_add_nameddest(PDF *p, const char *name, int len, const char *optlist)**

Create a named destination on a page in the document.

name (Hypertext string) Name of the destination, which can be used as a target for links, bookmarks, or other triggers. Destination names must be unique within a document. If the same name is supplied more than once for a document only the first definition is used; the others are silently ignored.

len (C language binding only) Length of *name* (in bytes). If *len* = 0 a null-terminated string must be provided.

optlist Option list specifying the destination. An empty list is equivalent to `{type=fitwindow page=0}`. The following options can be used:

- ▶ General option: *errorpolicy* (see Table 1.5)
- ▶ Destination control options according to Table 12.11:
bottom, group, left, page, right, structdest, top, type, zoom
Coordinates in these options are always interpreted in the default coordinate system; user coordinates are not supported since the coordinate system of the target page is not known.
- ▶ Encoding options for C and Perl, PHP, Ruby with *stringformat=legacy*:
hypertextencoding, hypertextformat (see Table 2.1)

Details The destination details must be specified in *optlist*, and the destination may be located on any page in the current document. The provided *name* can be used with the *destname* option in *PDF_create_action()*, *PDF_create_annotation()*, *PDF_create_bookmark()*, and *PDF_begin/end_document()*. This way defining and using a destination can be split into two separate steps.

Alternatively, if the destination is known at the time when it is used, defining and using the named destination can be combined by using the *destination* option of those methods, and *PDF_add_nameddest()* is not required in this case.

Scope any except *object*

Table 12.11 Destination options for *PDF_add_nameddest()*, as well as for the destination option in *PDF_create_action()*, *PDF_create_annotation()*, *PDF_create_bookmark()*, and *PDF_begin/end_document()*.

option	description
bottom	(Float; only for type=fitrect) The y coordinate of the page in default coordinates which will be positioned at the bottom edge of the window. Default: 0
group	(String; required if the page option has been specified and the document uses page groups; not allowed otherwise.) Name of the page group that the destination page belongs to.
left	(Float; only for type=fixed, fitheight, fitrect, or fitvisibleheight) The x coordinate of the page in default coordinates which will be positioned at the left edge of the window. Default: 0
page	(Integer) Page number of the destination page (first page is 1). The page must exist in the destination PDF. Page 0 means the current page if in page scope, and page 1 if in document scope. Default: 0

Table 12.11 Destination options for `PDF_add_nameddest()`, as well as for the destination option in `PDF_create_action()`, `PDF_create_annotation()`, `PDF_create_bookmark()`, and `PDF_begin/end_document()`.

option	description
right	(Float; only for type=fitrect) The x coordinate of the page in default coordinates which will be positioned at the right edge of the window. Default: 1000
structdest	<p>(String; only for Tagged PDF; must not be combined with the group and page options; PDF 2.0) The destination consists of a structure element specified by a string which must be assigned to the destination element with the <code>elementid</code> tagging option. This assignment may be done before or after referencing it with the <code>structdest</code> option. If the id is never assigned to any structure element until the end of the document, it will silently be assigned to the structure tree root.</p> <p>The structure element determines the page number of the destination; the exact location on the page, zoom factor etc. can be specified with additional options. If a named structure destination is used in <code>PDF_create_action()</code>, the action type must be <code>GoTo</code>.</p>
top	(Float; only for type=fixed, fitwidth, fitrect, or fitvisiblewidth) The y coordinate of the page in default coordinates which will be positioned at the top edge of the window. Default: 1000
type	<p>(Keyword) Location of the window on the target page. Supported keywords (default: fitwindow):</p> <p>fitheight Fit the page height to the window, with the x coordinate left at the left edge of the window.</p> <p>fitrect Fit the rectangle specified by left, bottom, right, and top to the window.</p> <p>fitvisible Fit the visible contents of the page (the ArtBox) to the window.</p> <p>fitvisibleheight Fit the visible contents of the page to the window with the x coordinate left at the left edge of the window.</p> <p>fitvisiblewidth Fit the visible contents of the page to the window with the y coordinate top at the top edge of the window.</p> <p>fitwidth Fit the page width to the window, with the y coordinate top at the top edge of the window.</p> <p>fitwindow Fit the complete page to the window.</p> <p>fixed Use a fixed destination view specified by the left, top, and zoom options. If any of these is missing its current value will be retained.</p>
zoom	(Float or percentage; only for type=fixed) Zoom factor (1 means 100%) to display the page contents. If this option is missing or 0 the zoom factor which was in effect when the link was activated is retained.



13 Multimedia, 3D and Geospatial Features

API methods in this chapter:

- ▶ `PDF_load_asset()`
- ▶ `PDF_load_3ddata()`
- ▶ `PDF_create_3dview()`

13.1 Multimedia Assets and Attachments

Multimedia features are implemented with the following API methods and options:

- ▶ For embedded or external file- or URL-based multimedia you must load the video or sound with `PDF_load_asset()` and `type=Rendition`, create a rendition action with `PDF_create_action()` (see Section 12.4, «Actions», page 237), and finally supply the action to a Screen annotation (see Section 12.2, «Annotations», page 221).
- ▶ Multimedia assets can be loaded with `PDF_load_asset()` and `type=3D/Sound/Video`. The asset can be supplied to a RichMedia annotation with the `richmedia` option. The sub-options of this option are described in Table 13.5. Related actions can be created with `PDF_create_action()` and `type=RichMediaExecute`. Note that RichMedia annotations are not recommended for sound and video; use rendition annotations instead.
- ▶ JavaScript can be loaded with `PDF_load_asset()` and `type=JavaScript` and used for 3D and RichMedia annotations as well as for JavaScript actions.
- ▶ `PDF_load_asset()` with `type=Attachment` can be used to load attachments or associated files for use in various API methods.

13.1.1 Loading Assets

```
C++ Java C# int load_asset(String type, String filename, String optlist)
Perl PHP int load_asset(string type, string filename, string optlist)
C int PDF_load_asset(PDF *p, const char *type, const char *filename, int len, const char *optlist)
```

Load a multimedia asset or attachment from a file or URL.

type Keyword designating the type of the loaded asset according to Table 13.1.

Table 13.1 Asset types

type	asset can be used with API method and option/suboption
3D	▶ <code>PDF_create_annotation()</code> with <code>type=RichMedia</code> and options <code>richmedia/assets/asset</code>
Attachment	▶ <code>PDF_end_document()</code> with option <code>attachments</code> ▶ <code>PDF_create_annotation()</code> with <code>type=FileAttachment</code> and option <code>attachment</code> ▶ PDF/A-3 and PDF 2.0: <code>PDF_end_document()</code> , <code>PDF_begin/end_page_ext()</code> , <code>PDF_create_annotation()</code> , <code>PDF_begin_item()</code> , <code>PDF_begin/end_dpart()</code> with option <code>associatedfiles</code> and common XObject option <code>associatedfiles</code> ▶ PDF 2.0: <code>PDF_begin/end_document()</code> , <code>PDF_begin/end_page_ext()</code> , <code>PDF_load_font()</code> and common XObject option <code>associatedfiles</code> ▶ PDF 2.0: <code>PDF_end_document()</code> with option <code>structureassociatedfiles</code>

Table 13.1 Asset types

type	asset can be used with API method and option/suboption
JavaScript	<ul style="list-style-type: none"> ▶ <i>PDF_create_annotation()</i> with type=RichMedia and option richmedia/assets/asset ▶ <i>PDF_create_action()</i> with option javascript ▶ <i>PDF_load_3ddata()</i> with option javascript
Rendition	▶ <i>PDF_create_action()</i> with type=rendition and option rendition
Sound, Video	▶ <i>PDF_create_annotation()</i> with type=RichMedia and options richmedia/assets/asset

filename (Name string; interpreted according to the *filenamehandling* global option, see Table 2.1) Name of a disk-based or virtual file which will be embedded or referenced in the PDF file. Unicode file names are supported, but require PDF 1.7. If *external=true* this parameter contains a URL instead of a filename.

len (C language binding only) Length of *filename* (in bytes). If *len = 0* a null-terminated string must be provided.

optlist An option list which may contain the following options:

- ▶ General option for all types: *errorpolicy* (see Table 1.5)
- ▶ For *type=Rendition* additional options according to Table 13.2:
autoplay, alttext, baseurl, controller, duration, external, mimetype, monitor, name, nametree, opacity, permissions, style, url, volume, window
- ▶ For *type=3D/Attachment/JavaScript/Sound/Video* additional options according to Table 13.4:
description, documentattachment, external, filename, mimetype, name, password, relationship, thumbnail

Returns An asset handle which can be used with the methods and options listed in Table 13.1 until the end of the enclosing *document* scope.

If *errorpolicy=return* the caller must check for a return value of -1 (in PHP: 0) since it signals an error. If the call fails you can request the reason of the failure with *PDF_get_errmsg()*.

Details The requirements for the contents of *filename* depend on the asset type:

- ▶ *type=Rendition* (PDF 1.5): *filename* must refer to a media file or contain a URL for a remote media file.
- ▶ *type=3D* (PDF 1.7ext3): U3D or PRC data. This type is only for use with *RichMedia-Execute* actions. 3D assets for use with 3D annotations and *GoTo3DView* actions must be loaded with *PDF_load_3ddata()*.
- ▶ *type=Attachment*: arbitrary contents in the general case, but PDF/A restrictions apply
- ▶ *type=JavaScript*: PDF 1.7 supports ECMAScript 3. PDF 2.0 supports ECMAScript 5.1 plus XML extensions and »Adobe Acrobat 3D JavaScript Reference«. In both cases extensions according to »Adobe JavaScript for Acrobat API Reference« are supported. The script must be encoded in ISO 8859-1 or UTF-16 LE or BE with BOM.
- ▶ *type=Sound* and *Video* (PDF 1.7ext3): audio data in MP3 format or video data in a format supported by Acrobat. These types are only for use with *RichMedia* annotations and *RichMediaExecute* actions. Sound and video assets for use with Screen annotations and rendition actions must be loaded with *type=Rendition*.

PDF/A PDF/A-1: this method must not be called.

PDF/A-2: only *type=Attachment* is allowed; *filename* must refer to a PDF/A-1 or PDF/A-2 document. Some options are restricted.

PDF/A-3: only *type=Attachment* is allowed; some options are restricted. An asset handle created with *type=Attachment* must be supplied to at least one *associatedfiles* option.

PDF/X PDF/X-3/4: only *type=Attachment* is allowed.

Scope any except *object*

13.1.2 Options for Rendition Assets

Table 13.2 Options for `PDF_load_asset()` with *type=Rendition*

option	description
autoplay	(Boolean) If true, automatically play the media when it is activated; otherwise the media remains paused. Default: true
alttext	(Hypertext string) Alternate text for the rendition which can be accessed via JavaScript
baseurl	(Hypertext string) Absolute URL to be used as the base URL in resolving any relative URLs found within the media data. Default: URL of the PDF document
controller	(Boolean) If true, display a controller user interface when playing the media. Default: false
duration	(Non-negative float or keyword) Playing time of the media (up to its intrinsic length). It can be specified in seconds or with the following keywords (default: intrinsic): infinite Don't stop playing the media. intrinsic Play the normal length of the media.
external	(Boolean; ignored if url is supplied) If true, the contents of the file are not embedded in the PDF, but a reference to an external file is created. It is recommended to avoid absolute file names and use relative names only. Default: false
mimetype	(String; required) MIME type of the file or URL. It must correctly describe the rendition's content type to ensure reliable playback.
monitor	(Keyword; only for style=floating and fullscreen; only relevant for multi-monitor setups) Monitor on which to display the window (default: document): bestcolor Monitor with the greatest color depth document Monitor containing the largest section of the document window largest Monitor with the greatest area in pixels nondocument Monitor containing the smallest section of the document window primary Primary monitor if available, otherwise same as documentlarge tallest Monitor with the greatest height in pixels widest Monitor with the greatest width in pixels
name	(Hypertext string; required) Rendition name which must be unique among all renditions in the document
nametree	(Boolean) Insert the rendition asset in the document's renditions name tree which makes it accessible to JavaScript. Default: false
opacity	(Float or percentage; ignored for style=fullscreen and hidden) Background opacity. Default: 1.0

Table 13.2 Options for `PDF_load_asset()` with `type=Rendition`

option	description
permissions	(Keyword; not allowed if <code>external</code> or <code>url</code> is supplied) Condition under which the PDF viewer is allowed to create a temporary file to play the media (default: <code>access</code>): <ul style="list-style-type: none"> access Allowed only if the document allows content extraction including for accessibility purposes (see option permissions in Table 3.3) always Always allowed extract Allowed only if the document allows content extraction (see option permissions in Table 3.3) never Never allowed
style	(Keyword) Window style for playing the rendition (default: <code>annotation</code>): <ul style="list-style-type: none"> annotation Area of the screen annotation associated with the rendition floating Floating window (requires the <code>window</code> option) fullscreen Full-screen window that obscures all other windows on the same monitor hidden Hidden window; useful for audio playback
url	(Boolean) If <code>true</code> , the <code>filename</code> parameter is interpreted as URL that refers to a video file or streaming media. Acrobat DC supports the RTMP (Real Time Messaging Protocol), HTTP and HTTPS protocols. On HTTP and HTTPS servers H.264-compliant MOV and MP4 videos are supported. Default: <code>false</code>
volume	(Float or percentage in the range 0..1) Volume as percentage of the recorded volume level with 0%=mute and 100%=maximum. Default: 100%
window	(Option list; only for <code>style=floating</code> and required in this case) Suboptions for the floating window according to Table 13.3

Table 13.3 Suboptions for the option `window` of `PDF_load_asset()` with `type=Rendition` and `style=floating`

option	description
align	(List of two keywords) Specifies where the floating window is positioned relative to the window specified in the <code>over</code> option. The first keyword must be <code>left</code> , <code>center</code> or <code>right</code> for the horizontal position. The second keyword must be <code>bottom</code> , <code>center</code> or <code>top</code> for the vertical position. Default: <code>{center center}</code>
canresize	(Keyword) Specifies whether and how the floating window can be resized (default: <code>no</code>): <ul style="list-style-type: none"> no The floating window cannot be resized. keepratio The floating window can be resized provided the aspect ratio is preserved. yes The floating window can be resized even if the aspect ratio is not preserved.
hasclose	(Boolean; only for <code>hastitle=true</code>) If <code>true</code> , the floating window can be closed by the user. Default: <code>true</code>
hastitle	(Boolean; only relevant for Acrobat on Windows) If <code>true</code> , a title bar is displayed for the floating window. Default: <code>true</code>

Table 13.3 Suboptions for the option window of `PDF_load_asset()` with `type=Rendition` and `style=floating`

option	description
height	(Positive integer; required) Height of the floating window (excluding title bar) in pixels
ifoffscreen	(Keyword) Specify what to do if the floating window is positioned totally or partially offscreen, i.e. not visible (default: <code>forceonscreen</code>): allow No special action forceonscreen Move or resize the window so that it is on-screen. cancel Cancel playing the rendition.
over	(Keyword): Window relative to which the floating window is aligned (default: <code>pagewindow</code>): appwindow The application window desktop The full desktop monitor The monitor specified in the monitor option pagewindow The document window
title	(String; only if <code>hastitle=true</code>) Text for the floating window's title bar
width	(Positive integer; required) Width of the floating window (excluding title bar) in pixels

13.1.3 Options for other Asset Types

Table 13.4 Options for `PDF_load_asset()` with `type=3D/Attachment/JavaScript/Sound/Video`, for `PDF_add_portfolio_folder/file()`, and for use as suboptions for the attachments option of `PDF_begin/end_document()`

option	description
description	(Hypertext string; PDF 1.6; recommended for PDF/A-2/3 and PDF/UA-1; not for <code>type=3D</code> and JavaScript) Descriptive text associated with the file.
document-attachment	(Boolean; only for PDF/A-3 and PDF 2.0; only for <code>type=Attachment</code> ; not for <code>PDF_add_portfolio_file/folder()</code>) Store the associated file also as a document-level attachment. This may be useful for listing the attachments in the user interface of the PDF viewer. Default: <code>false</code>
external	(Boolean; must be <code>false</code> for PDF/A; not for <code>type=3D</code> and JavaScript, the attachments option and <code>PDF_add_portfolio_file/folder()</code>) If <code>true</code> , the contents of the file are not embedded in the PDF, but a reference to an external file is created. It is recommended to avoid absolute file names and use relative names only. Default: <code>false</code>
filename	(Name string) Name of the file. The contents must conform to the requirements in the Details section according to the specified type. This option can alternatively be provided via the method parameter <code>filename</code> of <code>PDF_load_asset()</code> and <code>PDF_add_portfolio_file/folder()</code> , but it is required if used with the attachments option of <code>PDF_begin/end_document()</code> . Only the base part of the filename without any directory components is written to the PDF output unless <code>external=true</code> .
mimetype	(String; required for <code>type=Sound</code> and <code>Video</code> ; not for <code>type=3D</code> and JavaScript and <code>PDF_add_portfolio_folder()</code>) MIME type of the file. Default: <code>application/octet-stream</code>
name	(Hypertext string; not for <code>PDF_add_portfolio_folder()</code> ; not for <code>type=3D</code> and JavaScript) Name of the attachment. Default: <code>filename</code> without any path component
password	(String with up to 127 characters; not for PDF/A-2; only for <code>type=Attachment</code> ; only if PDI is available; not for <code>PDF_add_portfolio_folder()</code>) Master password required to open a protected PDF document for fetching its date entries.

Table 13.4 Options for `PDF_load_asset()` with `type=3D/Attachment/JavaScript/Sound/Video`, for `PDF_add_portfolio_folder/file()`, and for use as suboptions for the attachments option of `PDF_begin/end_document()`

option	description
relationship	<p>(Hypertext string; only for PDF 2.0 and PDF/A-3; only for <code>type=Attachment</code>; not for <code>PDF_add_portfolio_folder()</code>) Relationship of the file to the part of the document with which it is associated. The value may be an arbitrary string, but the following keywords are predefined (default: Unspecified):</p> <p>Alternative The file is an alternative representation (e.g. audio).</p> <p>Data The file represents information used to derive a visual presentation (e.g. CSV data for a table or graph).</p> <p>FormData The file represents data associated with the form fields in the document.</p> <p>Schema The file contains a schema definition, e.g. an XML schema for XMP metadata.</p> <p>Source The file is the original source material (e.g. word processor document associated with the document; spreadsheet associated with an image).</p> <p>Supplement The file represents a supplemental representation of the original source or data which may be more easily consumable (e.g. MathML representation of an equation in an image).</p> <p>Unspecified The relationship is not known or cannot be described with the other keywords.</p>
thumbnail	<p>(Image handle; only for <code>type=Attachment</code>; PDF 1.7ext3) Image to be used as thumbnail for a file or folder. The handle must have been created with <code>PDF_load_image()</code> and the image must be a grayscale or RGB image without ICC profile. RGB images must therefore be loaded with <code>iccprofile=none</code>. Acrobat ignores the thumbnail for attachments.</p>

13.2 Suboptions for RichMedia Annotations

Table 13.5 Suboptions for the richmedia option of `PDF_create_annotation()` with type=RichMedia

option	description
activate	(Option list) Option list according to Table 13.6 which specifies the style of presentation, default script behavior, default view information, and animation style when the annotation is activated.
assets	(List of one or more option lists; required) Named asset which can be referenced: asset (Asset handle; required) 3D, JavaScript, Sound, or Video asset name (Hypertext string with 1-255 characters; the characters : * " < > must not be used; the last character must not be a period '.'; required) Name of the asset
configuration	(Option list, required) The configuration list contains one or more instance option lists: instances (List of option lists; required) Each list specifies an asset with settings to populate the artwork of an annotation: asset (Hypertext string; required) Name of an asset specified in the assets option. Only names of rich media assets of type 3D, Sound, or Video may be specified here. name (Hypertext string) Unique name of the configuration type (Keyword) Primary content type for the configuration. Supported keywords are 3D, Sound and Video. Default: type of the first asset in the instances suboption
deactivate	(Option list) Condition of unloading (restart or pause): condition (Keyword) Specifies when the annotation is deactivated (default: clicked): clicked The annotation is explicitly deactivated by a user action or script. closed The annotation is deactivated as soon as the page that contains the annotation loses the focus as current page. invisible The annotation is deactivated as soon as the page is no longer visible.
views	(List of 3D view handles) 3D view handles returned by <code>PDF_create_3dview()</code> . If no views are specified, default values are used for the components of a 3D view, including rendering/lighting modes, background color, and camera data. Default: empty list

Table 13.6 Suboptions for the activate suboption of the richmedia option of PDF_create_annotation()

option	description
animation	(Option list) Preferred method to drive keyframe animations present in the artwork:
playcount	(Integer) A nonnegative number represents the number of times the animation is played. A negative integer indicates that the animation is infinitely repeated. Default: -1
speed	(Positive Float) A value greater than one shortens the time it takes to play the animation, or effectively speeds up the animation. This allows authors to change the desired speed of animations without re-authoring the content. Default: 1
style	(Keyword) Animation style (default: none):
none	Keyframe animations should not be driven by the viewer. This value is used by documents that drive animations via JavaScript. The remaining suboptions of the animation option are ignored.
linear	Keyframe animations are driven linearly from begin to end. This results in repetitive playback.
oscillating	Keyframe animations should oscillate along their time range. This results in back-and-forth playback.
condition	(Keyword) Specifies when the annotation is activated (default: clicked):
clicked	The annotation is explicitly activated by a user action or script.
opened	The annotation is activated when the page containing the annotation receives the focus.
visible	The annotation is activated when any part of the page becomes visible.
presentation	(Option list according to Table 13.7) Specifies how the annotation and user interface elements are laid out and drawn
scripts	(List of hypertext strings) Names of JavaScript assets specified in the assets suboption of the richmedia option.
view	(Keyword or 3D view handle) Activation view of 3D rich media. The handle must also be contained in the views suboption of the richmedia option list. If the views option was not specified default values for the components of a 3D view are used. Supported keyword (default: first):
first	The first element in the views suboption of the richmedia option list

Table 13.7 Suboptions for the presentation suboption of the activate suboption of the richmedia option of PDF_create_annotation()

option	description
navigation-pane	(Boolean) Default behavior of the navigation pane user interface element. If true the navigation pane is visible when the content is initially activated. Default: false
passcontext-click	(Boolean) Indicates whether a context click on the rich media annotation is passed to the media player runtime or is handled by the PDF viewer. If false, the PDF viewer handles the context click. If true, the PDF viewer's context menu is not visible, and the user sees the context menu and any custom items generated by the media player runtime. Default: false
style	(Keyword) Specifies how the rich media will be presented (default: embedded): embedded embedded within the PDF page windowed in a separate window specified by the window option list
toolbar	(Boolean) Default behavior of an interactive toolbar associated with this annotation. If true a toolbar is displayed when the annotation is activated and given focus. Default: true for configuration type 3D, false otherwise
transparent	(Boolean) Indicates whether the page content is displayed through the transparent areas of the rich media content. If true, the rich media artwork is composited over the page content using an alpha channel. Otherwise the rich media artwork is drawn over an opaque background. Default: false
window	(Option list) Size of the floating window for style>windowed. Coordinates are expressed in user or default coordinates depending on the usercoordinates option. The behavior is similar to when the annotation options zoom and rotate are set to false: heightdefault (Float) Default height of the window. Default: 216 (in default coordinates) widthdefault (Float) Default width of the window. Default: 288 (in default coordinates)

13.3 3D Artwork

3D contents can be implemented with 3D annotations or *RichMedia* annotations. *RichMedia* annotations are newer and provide more scripting events than 3D annotations. Table 13.9 summarizes differences and commonalities of both methods.

Table 13.8 Options for `PDF_create_3dview()`

	3D annotations	RichMedia annotations with 3D content
PDF compatibility	PDF 1.6	PDF 1.7ext3
load 3D data	<code>PDF_load_3ddata()</code>	<code>PDF_load_asset()</code> with <code>type=3D</code>
annotations	<code>PDF_create_annotation()</code> with <code>type=3D</code> ; relevant options are described in Section 13.3.3, «Options for 3D Annotations», page 259.	<code>PDF_create_annotation()</code> with <code>type=RichMedia</code> ; relevant suboptions of the <code>richmedia</code> option are described in Section 13.2, «Suboptions for RichMedia Annotations», page 251
apply 3D data	<code>PDF_create_annotation()</code> : option <code>3Ddata</code>	<code>PDF_create_annotation()</code> : option <code>richmedia/assets/asset</code>
create 3D views	<code>PDF_create_3dview()</code> (see Section 13.3.1, «3D Views for RichMedia and 3D Annotations», page 254)	
apply 3D views	<code>PDF_load_3ddata()</code> : options <code>defaultview</code> and <code>views</code> <code>PDF_create_annotation()</code> : option <code>3Dinitialview</code> <code>PDF_create_action()</code> with <code>type=GoTo3DView</code> : option <code>3Dview</code>	<code>PDF_create_annotation()</code> : option <code>richmedia</code> , suboption <code>views</code>
load JavaScript		<code>PDF_load_asset()</code> with <code>type=JavaScript</code>
apply JavaScript	<code>PDF_load_3ddata()</code> : option <code>javascript</code>	<code>PDF_create_annotation()</code> : option <code>richmedia/activate/scripts</code>
create 3D actions	<code>PDF_create_action()</code> with <code>type=3Dview</code>	<code>PDF_create_action()</code> with <code>type=RichMediaExecute</code>
apply 3D actions		<code>PDF_create_annotation()</code> : option <code>action</code>

13.3.1 3D Views for RichMedia and 3D Annotations

C++ Java C# `int create_3dview(String username, String optlist)`

Perl PHP `int create_3dview(string username, string optlist)`

C `int PDF_create_3dview(PDF *p, const char *username, int len, const char *optlist)`

Create a 3D view.

username (Hypertext string) User interface name of the 3D view.

len (C language binding only) Length of *username* (in bytes). If *len* = 0 a null-terminated string must be provided.

optlist An option list specifying 3D view properties:

- General options: *errorpolicy* (see Table 1.5)
- Options for specifying 3D view properties according to Table 13.9: *background*, *camera2world*, *cameradistance*, *lighting*, *name*, *rendermode*, *type*, *U3Dpath*

- ▶ Encoding option for C and Perl, PHP, Ruby with *stringformat=legacy*: *hypertextencoding* (see Table 2.1)

Returns A 3D view handle which can be used until the end of the enclosing *document* scope. If *errorpolicy=return* the caller must check for a return value of -1 (in PHP: 0) since it signals an error.

Details The 3D view handle can be attached to 3D models with the *views* option in *PDF_load_3ddata()* or can be used to create 3D annotations with *PDF_create_annotation()* or 3D-related actions with *PDF_create_action()*.

Scope any except *object*. The returned handle can be used until the next call to *PDF_end_document()*; requires PDF 1.6

Table 13.9 Options for *PDF_create_3dview()*

option	description
background	(Option list) Background for the 3D model: fillcolor (Color) Background color in the RGB color space. Default: white entire (Boolean) If true, the background applies to the entire annotation; otherwise it applies only to the rectangle specified in the annotations's 3Dbox option. Default: false
camera2world	(List of 12 floats) 3D transformation matrix specifying position and orientation of the camera in world coordinates (see description below). Default: the initial view defined internally in the 3D model
camera-distance	(Float; must not be negative; ignored if camera2world is not specified) Distance between the camera and the center of the orbit. For details see description of the CO key in section 13.6.4 »3D Views« of ISO 32000-1. Default: defined internally in the 3D data
lighting	(Option list; PDF 1.7) Lighting scheme for the 3D artwork: type (Keyword) Lighting scheme (Default: Artwork): Artwork Lights are specified in the 3D artwork. None No lights; lights specified in the 3D artwork are ignored. White Three light-grey infinite lights, no ambient term Day Three light-grey infinite lights, no ambient term Night One yellow, one aqua, and one blue infinite light, no ambient term Hard Three grey infinite lights, moderate ambient term Primary One red, one green, and one blue infinite light, no ambient term Blue Three blue infinite lights, no ambient term Red Three red infinite lights, no ambient term Cube Six grey infinite lights aligned with the major axes, no ambient term CAD Three grey infinite lights and one light attached to the camera, no ambient term Headlamp Single infinite light attached to the camera, low ambient term
name	(Hypertext string) Name of the 3D view, which can be used in GoTo actions. This is an optional internal name which is treated separately from the required username parameter.
rendermode	(Option list; PDF 1.7) Render mode for displaying the 3D artwork according to Table 13.10.
type	(Keyword; required if the view will be used in <i>PDF_load_3ddata()</i> with type=PRC) Type of 3D data (default: U3D): PRC The view will be used in <i>PDF_load_3ddata()</i> with type=PRC. U3D The view will be used in <i>PDF_load_3ddata()</i> with type=U3D.
U3Dpath	(Hypertext string; ignored if the camera2world option is specified; only for type=U3D) A View Node name used to access a view node within the 3D artwork.

Camera position. The position of the camera can be specified with the *camera2world* option. Alternatively, JavaScript code can be attached to position and align the camera towards the model. The PDFlib Cookbook contains sample code for attaching JavaScript code to a 3D model.

The following values can be supplied to the *camera2world* option for common camera positions. *x*, *y*, and *z* are suitable values which describe the position of the camera. These values should satisfy the stated conditions:

View from the front:

```
{-1 0 0 0 0 1 0 1 0 x y z} x small, y large negative, z small
```

View from left:

```
{ 0 1 0 0 0 1 1 0 0 x 0 z} x large negative, z small
```

View from the top:

```
{-1 0 0 0 1 0 0 0 -1 x 0 z} x small, z large positive
```

View from the back:

```
{ 1 0 0 0 0 1 0 -1 0 x y z} x small, y large positive, z small
```

View from the bottom:

```
{-1 0 0 0 -1 0 0 0 1 x 0 z} x small, z large negative
```

View from right:

```
{ 0 -1 0 0 0 1 -1 0 0 x 0 z} x large positive, z small
```

Isometric view, i.e. the direction of projection intersects all three axes at the same angle. There are eight such views, one in each octant:

```
{0.707107 -0.707107 0 -0.5 -0.5 0.707107 -0.5 -0.5 -0.707107 x y z}  
x, y, z large positive
```

The *x*, *y*, *z* values should be selected depending on the position and size of the model.

»Large« means the values should be significantly larger than the size of the model in order to provide a large enough distance between the camera and the model. If the value is too large the model will appear very small and will quickly get out of sight when rotating the view. If the value is too small the model may not completely fit into the view.

»Small« means the absolute value should be small compared to the large value and should not exceed the size of the model very much.

Table 13.10 Suboptions for the rendermode option of PDF_create_3dview()

option	description
crease	(Float in the range 0..180) Crease value
facecolor	(RGB color or keyword; only for type=Illustration) Face color. The keyword backgroundcolor refers to the current background color. Default: backgroundcolor
opacity	(Float in the range 0..1 or percentage) Opacity for some render modes. Default: 0.5
rendercolor	(RGB color) Auxiliary color used by several render modes. Default: black
type	<p>(Keyword; PDF 1.7) Render mode for displaying the 3D artwork (default: Artwork):</p> <p>Artwork Render mode is specified in the 3D artwork; all other suboptions of the rendermode option are ignored.</p> <p>Solid Displays textured and lit geometric shapes.</p> <p>SolidWireframe Displays textured and lit geometric shapes (triangles) with single color edges on top of them.</p> <p>Transparent Displays textured and lit geometric shapes (triangles) with an added level of transparency.</p> <p>TransparentWireframe Displays textured and lit geometric shapes (triangles) with an added level of transparency.</p> <p>BoundingBox Displays textured and lit geometric shapes (triangles) with an added level of transparency, with single color opaque edges on top of it.</p> <p>TransparentBoundingBox Displays bounding boxes faces of each node, aligned with the axes of the local coordinate space for that node, with an added level of transparency.</p> <p>TransparentBoundingBoxOutline Displays bounding boxes edges and faces of each node, aligned with the axes of the local coordinate space for that node, with an added level of transparency.</p> <p>Wireframe Displays bounding boxes edges and faces of each node, aligned with the axes of the local coordinate space for that node, with an added level of transparency.</p> <p>ShadedWireframe Displays only edges, though interpolates their color between their two vertices and applies lighting.</p> <p>HiddenWireframe Displays edges in a single color, though removes back-facing and obscured edges.</p> <p>Vertices Displays only vertices in a single color.</p> <p>ShadedVertices Displays only vertices, though uses their vertex color and applies lighting.</p> <p>Illustration Displays silhouette edges with surfaces, removes obscured lines.</p> <p>SolidOutline Displays silhouette edges with lit and textured surfaces, removes obscured lines.</p> <p>ShadedIllustration Displays silhouette edges with lit and textured surfaces and an additional emissive term to remove poorly lit areas of the artwork.</p>

13.3.2 Loading Contents for 3D Annotations

C++ Java C# `int load_3ddata(String filename, String optlist)`

Perl PHP `int load_3ddata(string filename, string optlist)`

C `int PDF_load_3ddata(PDF *p, const char *filename, int len, const char *optlist)`

Load a 3D model from a disk-based or virtual file.

filename (Name string; will be interpreted according to the `filenamehandling` global option, see Table 2.1) Name of a disk-based or virtual file containing a 3D model.

len (C language binding only) Length of `filename` (in bytes). If `len = 0` a null-terminated string must be provided.

optlist An option list specifying properties of the 3D model:

- ▶ General options: `errorpolicy` (see Table 1.5)
- ▶ Options for specifying properties of the 3D model according to Table 13.11: `animation`, `defaultview`, `javascript`, `script`, `type`, `views`
- ▶ Encoding option for C and Perl, PHP, Ruby with `stringformat=legacy`: `hypertextencoding` (see Table 2.1)

Returns A 3D handle which can be used to create 3D annotations with `PDF_create_annotation()` and `type=3D`. The 3D handle can be used until the end of the enclosing `document` scope. If `errorpolicy=return` the caller must check for a return value of -1 (in PHP: 0) since it signals an error.

Details The file must contain 3D data in PRC or U3D format.

Scope any except `object`. The returned handle can be used until the next call to `PDF_end_document()`; requires PDF 1.6

Table 13.11 Options for `PDF_load_3ddata()`

option	description
animation	(Option list) Preferred method to drive keyframe animations present in the artwork. Supported options are <code>playcount</code> , <code>speed</code> , and <code>style</code> (same as for the <code>animation</code> suboption of the <code>activate</code> suboption of the <code>richmedia</code> option of <code>PDF_create_annotation()</code> , see Table 13.6)
defaultview	(Keyword or 3D view handle) Initial view of the 3D annotation; One of the keywords <code>first</code> or <code>last</code> (referring to the respective entries in the <code>views</code> option), or a 3D view handle created with <code>PDF_create_3dview()</code> . Default: <code>first</code>
javascript	(Asset handle which has been loaded with <code>type=JavaScript</code> ; exactly one of the options <code>script</code> and <code>javascript</code> is required) JavaScript to be executed when the 3D model is instantiated.
script	(Hypertext string) String containing JavaScript code to be executed when the 3D model is instantiated. Default: no script
type	(Keyword) Type of 3D data (default: U3D): PRC (PDF 1.7ext3) Product Representation Compact (PRC) format (ISO 14739-1) U3D (PDF 1.6) Universal 3D File Format (U3D) in the following flavors: Universal 3D File Format (U3D) 1st Edition and ECMA-363 Universal 3D File Format (U3D) 3rd Edition
views	(List of 3D view handles) List of views for the 3D model. Each list element is a 3D view handle created with <code>PDF_create_3dview()</code> . The <code>type</code> option used when creating the views with <code>PDF_create_3dview()</code> must match the <code>type</code> option in <code>PDF_load_3ddata()</code> . Default: empty list

13.3.3 Options for 3D Annotations

Table 13.12 3D options for `PDF_create_annotation()` with `type=3D`

option	description
3Dactivate	<p>(Option list) Specifies when the 3D annotation should be activated and its state upon activation/deactivation</p> <p>enable (Keyword) Specifies when the animation should be enabled (default: click):</p> <ul style="list-style-type: none"> open Activate when the page is opened. visible Activate when the page becomes visible. click Annotation must explicitly be activated by a script or user action. <p>enablestate (Keyword) Initial animation state (default: play):</p> <ul style="list-style-type: none"> pause The 3D model is instantiated, but script animations are disabled. play The 3D model is instantiated; script animations are enabled if present. <p>disable (Keyword) Specifies when the animation should be disabled (default: invisible):</p> <ul style="list-style-type: none"> close Deactivate when the page is closed. invisible Deactivate when the page becomes invisible. click Annotation must explicitly be deactivated by a script or user action. <p>disablestate (Keyword) State of the animation upon disabling (default: reset):</p> <ul style="list-style-type: none"> pause The 3D model can be rendered, but animations are disabled. play The 3D model can be rendered and animations are enabled. reset Initial state of the 3D model before it has been used in any way. <p>modeltree (Boolean; PDF 1.6) If true, the Model Tree navigation tab will be opened when the annotation is activated (default: false)</p> <p>toolbar (Boolean; PDF 1.6) If true, the 3D toolbar (at the top of the annotation) will be displayed when the annotation is activated (default: true)</p>
3Ddata	(3D handle; only for <code>type=3D</code> ; required) 3D handle created with <code>PDF_load_3ddata()</code> .
3Dinteractive	(Boolean) If true, the 3D model is intended for interactive use. If false, it is intended to be manipulated with JavaScript. Default: true
3Dshared	(Boolean;) If true, the 3D data specified in the <code>3Ddata</code> option will be referenced indirectly. Multiple 3D annotations which reference the same data share a single run-time instance of the model. This means that changes will be visible in all such annotations simultaneously. Default: false
3Dinitialview	(Keyword or 3D view handle) Initial view of the 3D model with one of the keywords first, last, (referring to the respective entries in the <code>views</code> option of <code>PDF_load_3ddata()</code>), or default (referring to the model's <code>defaultview</code> option), or a 3D view handle created with <code>PDF_create_3dview()</code> . Default: default

13.4 Geospatial Features

Geospatial features are implemented with the following methods and options:

- ▶ One or more georeferenced areas can be assigned to a page with the *viewports* option of *PDF_begin/end_page_ext()*.
- ▶ The *georeference* option of *PDF_load_image()* can be used to assign an earth-based coordinate system to an image.
- ▶ The *georeference* option of *PDF_open_pdi_page()*, *PDF_load_graphics()* and *PDF_begin_template_ext()* can be used to assign an earth-based coordinate system to a Form XObject. However, this method is not recommended since it is not supported in any known viewer including Acrobat DC.

Table 13.13 and subsequent tables specify the options for geospatial features in detail.

Table 13.13 Suboptions for the *viewports* option of *PDF_begin/end_page_ext()*

option	description
bounding-box	(Rectangle; required) A rectangle in default coordinates specifying the location of the viewport on the page.
georeference	(Option list according to Table 13.14; required) Description of a world coordinate system associated with the viewport to use for geospatial measuring
hypertext-encoding	(Keyword) Encoding for the name option. An empty string is equivalent to unicode. Default: the value of the <i>hypertextencoding</i> global option
name	(Hypertext string) A descriptive title of the viewport (map name). However, Acrobat does not display the viewport name in the user interface.

Table 13.14 Suboptions for the *georeference* option of *PDF_load_image()*, *PDF_open_pdi_page()*, *PDF_load_graphics()*, *PDF_begin_template_ext()* and the *georeference* suboption of the *viewports* option of *PDF_begin/end_page_ext()*

option	description
angularunit	(Keyword) Preferred angular display unit (default: deg): degree degrees grad grad (1/400 of the full circle, or 0.9 degrees)
areaunit	(Keyword) Preferred area display unit (default: sqm): sqm square meter ha hectar (10.000 square meters) sqkm square kilometer sqft square foot a acre sqmi square mile <i>The specified unit will be used for display only if the following Acrobat setting is disabled: »Preferences, Measuring (Geo), Use Default Area Unit«.</i>
bounds	(Polyline with two or more points) Bounds of an area for which the geospatial transformations are valid (for maps this bounding polyline is known as a neatline). The points are expressed relative to the boundingbox of a page viewport or the extent of an template or image. Default: {0 0 0 1 1 1 1 0}, i.e. the full viewport, template or image area will be used for the map.

Table 13.14 Suboptions for the georeference option of `PDF_load_image()`, `PDF_open_pdi_page()`, `PDF_load_graphics()`, `PDF_begin_template_ext()` and the georeference suboption of the viewports option of `PDF_begin/end_page_ext()`

option	description
displaysystem	(Option list) Coordinate system according to Table 13.15 for the user-visible display of position values, such as latitude and longitude. This entry can be used to display the coordinates in another system than the one supplied in the <code>coords</code> option to specify the map.
linearunit	(Keyword) Preferred linear display unit (default: m) : m meter km kilometer ft international foot usft US survey foot mi international mile nm nautical mile The specified unit will be used for display only if the following Acrobat setting is disabled: »Preferences, Measuring (Geo), Use Default Distance Unit«.
mappoints	(List with two or more pairs of floats; required) A list of numbers where each pair defines a point in a 2D unit square. The unit square is mapped to the rectangular bounds of the page viewport, graphics, template or PDI page, or image which contains the georeference option list. The <code>mappoints</code> list must contain the same number of points as the <code>worldpoints</code> list; each point is the map position in the unit square corresponding to the geospatial position in the <code>worldpoints</code> list.
worldpoints	(List with two or more pairs of floats; required) A list of coordinate pairs where each pair specifies the world coordinates of the corresponding point in the <code>mappoints</code> option. The number of pairs must match the number of pairs in the <code>mappoints</code> option. The coordinate values are based on the coordinate system specified in the <code>worldsystem</code> option: if <code>type=geographic</code> , latitude/longitude values in degrees must be provided. If <code>type=projected</code> , projected x/y values must be provided.
worldsystem	(Option list; required) World coordinate system (for interpretation of <code>worldpoints</code>) according to Table 13.15.

Table 13.15 Suboptions for the `worldsystem` and `displaysystem` suboptions of the georeference option of `PDF_load_image()`, `PDF_open_pdi_page()`, `PDF_load_graphics()`, `PDF_begin_template_ext()` and the georeference suboption of the viewports option of `PDF_begin/end_page_ext()`

option	description
epsg	(Integer; exactly one of <code>epsg</code> or <code>wkt</code> must be supplied) Coordinate system as an EPSG reference code. Note that Acrobat does not support EPSG codes for <code>type=geographic</code> ; use <code>wkt</code> in this case.
type	(Keyword; required) Type of the coordinate system: geographic geographic coordinate system (supports only <code>wkt</code>) projected projected coordinate system (supports <code>wkt</code> and <code>epsg</code>)
wkt	(String with up to 1024 ASCII characters; exactly one of <code>epsg</code> or <code>wkt</code> must be supplied) Coordinate system as a string of »Well Known Text« (WKT). WKT is recommended for custom coordinate systems without any EPSG code.



14 Tagged PDF and Marked Content

API methods in this chapter:

- ▶ `PDF_begin_item()`
- ▶ `PDF_end_item()`
- ▶ `PDF_activate_item()`
- ▶ `PDF_begin_mc()`
- ▶ `PDF_end_mc()`

14.1 Tagging Methods

The `tagged` option in `PDF_begin_document()` must be set to `true` in order to generate Tagged PDF. Tagged PDF mode is automatically activated for the PDF/A-1a/2a/3a and PDF/UA standards. It is strongly recommended to obey to PDF/UA rules when creating Tagged PDF.

C++ Java C# `int begin_item(String tagname, String optlist)`

Perl PHP `int begin_item(string tagname, string optlist)`

C `int PDF_begin_item(PDF *p, const char *tagname, const char *optlist)`

Open a structure element or other content element for Tagged PDF.

tagname Name of the item's element structure type. The following groups of structure types are supported according to Table 14.1 (see PDFlib Tutorial for descriptions):

- ▶ standard types (a detailed description of standard types can be found in the PDFlib Tutorial)
- ▶ pseudo types which are not structure elements
- ▶ The tag name `Plib_custom_tag` implies use of a custom type (this is equivalent to `customtag=true`); the actual tag name must be supplied in the `tagname` option. Custom types require the `rolemap` document option.

The tag name can alternatively be provided via the `tagname` option which overrides this parameter.

optlist An option list specifying details of the item. All inheritable settings are inherited to child elements and therefore need not be repeated. All properties of an item must be set here since they cannot be modified later. The following options can be used:

- ▶ Processing options according to Table 14.2:
associatedfiles, bookmark, contents, customtag, direct, elementid, id, index, metadata, namespace, parent, ref, tag, tagname, title, usercoordinates
- ▶ Options for structure element attributes according to Table 14.3:
ActualText, Alt, ARIARole, artifactssubtype, artifacttype, Attached, BBox, Checked, ColSpan, ContinuedFrom, ContinuedList, Desc, E, Headers, Height, Lang, ListNumbering, Placement, PrintFieldRole, RowSpan, Scope, Short, Summary, Width, WritingMode
- ▶ Encoding option for C and Perl, PHP, Ruby with `stringformat=legacy`:
hypertextencoding (see Table 2.1)

Returns An item handle which can be used in subsequent item-related calls.

Table 14.1 Standard, pseudo, and custom structure types for `PDF_begin_item()`, `PDF_begin_mc()`, and `PDF_mc_point()` and the tag option for abbreviated tagging with various methods

category	tags
standard structure types	
document	Document, DocumentFragment ¹
grouping	Art, Aside ¹ , BlockQuote, Caption, Div, Document, Index, NonStruct, Part, Private, Sect, TOC, TOCI
block and sub-block	P, FENote ¹ , H, H1, H2, H3, H4, H5, H6 H7 ¹ , H8 ¹ , etc. (also allowed in PDF/UA-1 with role mapping) Sub ¹ , Title ¹
label and list	L, LI, Lbl, LBody
table	Table, TR, TH, TD, THead ² , TBody ² , TFoot ² (all table tags can be created automatically by <code>PDF_fit_table()</code> , see PDFlib Tutorial)
inline	Em ¹ , BibEntry, Code, Lbl, Note, Quote, Reference, Span, Strong ¹
interactive	Annot ² Form, Link
illustration	Figure, Formula
Japanese	Ruby ² (grouping), RB ² , RT ² , RP ² , Warichu ² (grouping), WT ² , WP ²
Artifact	Artifact ¹ (as regular item, as opposed to pseudo element Artifact which is emitted directly)
pseudo structure types	
non-structural elements	Artifact Artifact, to be distinguished from real page content.
	ASpan (Accessibility span; written to PDF as Span, but must be distinguished from the direct item Span) Can be used to attach accessibility attributes to content which does not belong to a structure element or which resembles a part of a structure element.
	ReversedChars (Not recommended) Text in a right-to-left language with reversed characters
	Clip (Not recommended) Marked clipping sequence, i.e. only clipping paths or text in rendering mode 7, but no visible graphics or <code>PDF_save()</code> / <code>PDF_restore()</code>
custom structure types	
user-defined types	The tag name <code>Plib_custom_tag</code> must be supplied in the tagname parameter. The actual tag name which is written to PDF must be supplied in the tagname option. Custom types require the rolemap document option.

1. Requires PDF 2.0
2. Requires PDF 1.5 or above

Details Start a new structure element or Artifact (collectively called *item*). By default, the new element is inserted in the document structure tree as a child of the currently active item. However, another position in the structure tree can be specified with the *parent* and *index* options. Structure elements can be nested to an arbitrary level. Regular elements (as opposed to direct elements without any entry in the structure tree) are not bound to the page where they have been opened, but can be continued on an arbitrary number of pages.

Structure elements and *Alt/ActualText* attributes must be properly nested according to the rules in the PDFlib Tutorial. Some decorative elements are automatically tagged as *Artifact*; see PDFlib Tutorial for details.

PDF/A Although Tagging is required for PDF/A-1a/2a/3a, there are no specific requirements for tag usage or nesting. We recommend to obey PDF/UA-1 requirements.

PDF/UA All image and graphics contents must be tagged as *Artifact* or *Figure*. Additional rules apply to various structure types and options (see PDFlib Tutorial).

Scope *page*; for grouping elements also *document*; must always be paired with a matching `PDF_end_item()` call. This method is only allowed in Tagged PDF mode.

Table 14.2 Processing options for structure types for `PDF_begin_item()` as well as for abbreviated tagging with the tag option of various methods. Some options are also available in `PDF_begin_mc()` and `PDF_mc_point()`.

option	description
associatedfiles	(List of asset handles; only for regular elements; only for PDF 2.0 and PDF/A-3) Asset handle(s) for file(s) associated with the structure element. The files must have been loaded with <code>PDF_load_asset()</code> and <code>type=attachment</code> .
bookmark	(Bookmark handle; only for regular elements) Handle for a bookmark which will be associated with the structure element.
contents	(String; only for <code>PDF_begin_mc()</code>) Content string describing the marked content sequence.
customtag	(Boolean; requires the <code>rolemap document</code> option) If true, the type name supplied in the <code>tagname</code> option is a custom type which must be mapped to a standard type via the <code>rolemap document</code> option. Default: false Setting this option to true is equivalent to providing the parameter <code>tagname=Plib_custom_tag</code> .
direct	(Boolean; only for <code>tagname=Artifact, Code, BibEntry, Em, FENote, Note, Quote, Reference, Span, Strong, Sub</code> ; for <code>Lbl</code> as child of <code>BibEntry, TOCI</code> or <code>Note</code> ; and for the pseudo items <code>ASpan, ReversedChars, Clip</code>) If true only a marked content sequence («direct» element) is emitted without an entry in the structure tree. Otherwise, emit the item as marked content sequence and create an entry in the structure tree («regular» element). Default: false if the <code>Title</code> option is supplied, otherwise true
elementid	(String; only for regular elements; PDF 2.0) A unique string identifier for the structure element which can be used to reference this element as footnote or destination. Unlike the <code>id</code> option this identifier is used only internally and is not written to the PDF output.
id	(String; not for pseudo elements and inline elements except <code>Note</code> ; in PDF/UA-1 required for footnotes and endnotes tagged as <code>Note</code>) Assign an identifier to the element for use with the <code>Headers</code> option. The string must be unique among all structure elements.
index	(Integer; not for pseudo tags) The zero-based index at which to insert the element within the parent. Starting at this position, existing descendants of the parent are shifted upwards. Values between 0 and the current number of children can be supplied. The value -1 adds the element at the end, i.e. as the new last item. This is identical to supplying the current number of elements as <code>index</code> . Default: -1
metadata	(Option list; only for regular elements) Metadata for the structure element (see Section 15.2, «XMP Metadata», page 277).
namespace	(String; PDF 2.0; required for tags outside the PDF 1.7/PDF 2.0 standard tagsets) Namespace identifying the tagset which contains the type. It must be one of the namespaces specified in the tagsets <code>document</code> option.
parent	(Item handle; not for pseudo and inline structure types) The element's parent as returned by an earlier call to <code>PDF_begin_item()</code> or the <code>activeitemid</code> keyword of <code>PDF_get_option()</code> . The value 0 refers to the structure tree root. -1 refers to the currently active element. In other words, <code>parent=-1</code> opens a child of the current element. Default: -1

Table 14.2 Processing options for structure types for `PDF_begin_item()` as well as for abbreviated tagging with the `tag` option of various methods. Some options are also available in `PDF_begin_mc()` and `PDF_mc_point()`.

option	description
ref	(List of strings; only for regular elements; PDF 2.0) Reference id(s) for one or more structure elements to which the current structure element refers, e.g. footnotes, endnotes or sidebars. The referenced element may already exist or may be created later. If the referenced id is never assigned to any structure element until the end of the document, the reference is silently ignored.
tag	(Option list) Additional structure element which will be inserted as a child of the current element. All options according to Table 14.2 are supported as suboptions. Tags can be nested to an arbitrary level.
tagname	(Name string; except in <code>PDF_add_table_cell()</code> this option is required for abbreviated tagging with the <code>tag</code> option) Name of a standard type or a pseudo type according to Table 14.1, or name of a custom tag. The value of this option can alternatively be provided via the method parameter <code>tagname</code> . Specify the option <code>customtag=true</code> or the parameter <code>tagname=Plib_custom_tag</code> to supply custom tag names. In this case the <code>tagname</code> option contains an arbitrary name of a custom tag for which a mapping to a standard type must have been defined with the <code>rolemap</code> option of <code>PDF_begin_document()</code> . Custom type names are restricted to 127 winansi characters or a sequence of Unicode characters which expands to a maximum of 127 UTF-8 bytes. Custom type names must not start with the reserved prefix <code>Plib</code> .
title	(Hypertext string; not for inline and pseudo structure types; recommended for headings in PDF/UA-1) Title of the structure element. The title may be useful for viewing or manipulating the structure tree in Acrobat.
user-coordinates	(Boolean) If false, <code>BBox</code> , <code>Width</code> and <code>Height</code> are expected in default coordinates; otherwise user coordinates will be used. Default: the value of the <code>usercoordinates</code> global option

Table 14.3 Attribute options for structure types for `PDF_begin_item()` as well as for abbreviated tagging with the tag option of various methods. Some options are also available in `PDF_begin_mc()` and `PDF_mc_point()`.

option	description
ActualText	(Hypertext string; not for pseudo tags except in PDF 1.5 with ASpan; if used in PDF 1.4 the direct option must be set to true) Equivalent replacement text for the content item and its kids.
Alt	(Hypertext string; not for pseudo tags except in PDF 1.5 with ASpan; if used in PDF 1.4 the direct option must be set to true) Word or phrase as alternate description for the content item and its children. It should be provided for figures, images, etc. which cannot be recognized as text.
ARIARole	(String; PDF 2.0) Role of the structure element according to ARIA 1.1. The supplied string must contain one of the keywords listed in WAI-ARIA (www.w3.org/TR/dpub-aria-1.0/#roles).
artifact-subtype	(Keyword; only for tagname=Artifact and artifacttype=Pagination or artifacttype=Inline; PDF 1.7) Subtype of the artifact: Header, Footer, Watermark; in PDF 2.0 the following additional keywords are supported: Bates, LineNum, PageNum, Redaction
artifacttype	(Keyword; only for tagname=Artifact) Identifies the artifact type of the content item: Background (PDF 1.7) Images or colored items that run the entire length and/or width of the page or a structural element. Inline (PDF 2.0) Artifact content that has context in the document's logical structure, typically, artifacts of subtype LineNum or Redaction. Layout Typographic or design elements such as footnote rules or table shading Page Production aids such as cut marks and color bars Pagination Ancillary page features such as running heads or page numbers
Attached	(Keyword list; only for tagname=Artifact and artifacttype=Pagination or Background with full-page background artifacts) Specify the edges of the page, if any, to which the artifact is logically attached. The list contains one to four of the keywords Top, Bottom, Left, and Right. Including both Left and Right or both Top and Bottom indicates a full-width or full-height artifact, respectively.
BBox	(Rectangle; only for tagname=Artifact, Figure, Formula, Form, Table; required for artifacttype=Background, otherwise optional) The element's bounding box in default coordinates (if usercoordinates=false) or user coordinates (if usercoordinates=true). PDFlib automatically creates the BBox for placed images, graphics, PDF pages (with tagname=Figure or Artifact), form fields (with tagname=Form or Artifact) and tables created by the table engine (with tagname=Table or Artifact).
Checked	(Keyword; PDF 1.7; only for PrintFieldRole=rb or cb) State of a radio button or check box field: on, off or neutral
ColSpan	(Integer; only for tagname=TH and TD) Number of table columns spanned by a cell. Default: 1
Continued-From	(String; only for tagname=L; not allowed if continuedlist=false; PDF 2.0) Identifier of the list for which this list is a continuation. The identifier must have been assigned to another list with the id option.
Continued-List	(Boolean; only for tagname=L; forced to true if continuedfrom is supplied; PDF 2.0) If true, the list is a continuation of a previous list. The previous list is specified in the continuedfrom option or (if that option is absent) the preceding list at the same level in the structure hierarchy. Default: false
Desc	(Hypertext string; PDF 1.7) Alternate name of the field (similar to the tooltip option of <code>PDF_create_field()</code>).
E	(Hypertext string; not for pseudo tags except ASpan; requires PDF 1.5 for structure tags) Abbreviation expansion for the content item. It should be provided for explaining abbreviations and acronyms. Acrobat's Read Aloud feature considers the expansion text as a separate word even in the absence of explicit word breaks.

Table 14.3 Attribute options for structure types for `PDF_begin_item()` as well as for abbreviated tagging with the tag option of various methods. Some options are also available in `PDF_begin_mc()` and `PDF_mc_point()`.

option	description
Headers	(List of strings; only for tagname=TH and TD; PDF 1.5) Each string in the list is the identifier of a table header cell (TH element) which is associated with the cell. The identifier(s) must have been assigned to the target cell with the id option. If multiple header cells are referenced, row headers should be listed before column headers. Within these groups headers should be ordered from most specific to most general.
Height	(Float; only for tagname=Figure, Form, Formula, Table, TD, TH) Height of the element in default coordinates (if usercoordinates=false) or user coordinates (if usercoordinates=true)
Lang	(String; not for pseudo tags except ASpan) Language identifier for the content item (and associated options ActualText, Alt, E) in the format described in Table 3.3 for the lang option. It can be used to override the document's default language for individual content items.
List-Numbering	(Keyword; only for tagname=L; required in PDF/UA-1; must be None in PDF/UA-1 for lists where no LI child contains any Lbl element) Numbering system used for the contents of the Lbl elements in a numbered list, or the symbol which precedes each item in an unnumbered list (default: None): <ul style="list-style-type: none"> Circle Open circular bullet Decimal Decimal arabic numerals (1–9, 10–99, ...) Description (PDF 2.0) List of terms for corresponding definitions Disc Solid circular bullet LowerAlpha Lowercase letters (a, b, c, ...) LowerRoman Lowercase roman numerals (i, ii, iii, iv, ...) None No autonumbering; Lbl elements (if present) contain arbitrary text without numbering. In this case any graphics representing the list's labels should be marked as Artifact. Ordered (PDF 2.0) Ordered list with unspecified numbering Square Solid square bullet Unordered (PDF 2.0) Unordered list with unspecified bullets UpperAlpha Uppercase letters (A, B, C, ...) UpperRoman Uppercase roman numerals (I, II, III, IV, ...)
Placement	(Keyword; not for pseudo elements and the inline types Span, Quote, Note, Reference, BibEntry and Code) Positioning of the element with respect to the enclosing reference area. This is relevant when the document is reformatted or exported to other formats. The option Placement=Block is recommended for Figure, Formula, Form, Link, Annot elements if they are created as children of grouping elements (default: Inline): <ul style="list-style-type: none"> Before Placed so that the »before« edge of the element's allocation rectangle coincides with that of the nearest enclosing reference area. Block Stacked in the block progression direction within an enclosing reference area or parent BLSE. End Placed so that the »end« edge of the element's allocation rectangle coincides with that of the nearest enclosing reference area. Inline Packed in the inline progression direction within an enclosing BLSE. Start Placed so that the »start« edge of the element's allocation rectangle coincides with that of the nearest enclosing reference area.

Table 14.3 Attribute options for structure types for `PDF_begin_item()` as well as for abbreviated tagging with the `tag` option of various methods. Some options are also available in `PDF_begin_mc()` and `PDF_mc_point()`.

option	description
PrintField-Role	(Keyword; PDF 1.7) Type of printed (non-interactive) form field: rb Radio button cb Check box pb Push button tv Text-value field; the text value of the field shall be the content of a Form structure element. lb Listbox Non-interactive form fields and associated content should be enclosed within a Part structure element.
RowSpan	(Integer; only for tagname=TH and TD) The number of table rows spanned by a cell. Default: 1
Scope	(Keyword; only for tagname=TH; PDF 1.5; recommended for PDF/UA-1) One of the keywords Row, Column, or Both indicating whether the table header cell applies to the rest of the cells in the row that contains it, the column that contains it, or both the row and the column that contain it.
Short	(String; only for tagname=TH; PDF 2.0) Short form of a table header cell's content
Summary	(Hypertext string; only for tagname=Table; PDF 1.7) Summary of the table's purpose and structure
Width	(Float; only for tagname=Figure, Form, Formula, Table, TD, TH) Width of the element in default coordinates (if <code>usercoordinates=false</code>) or user coordinates (if <code>usercoordinates=true</code>)
Writing-Mode	(Keyword) Direction of layout progression for packing ILSEs and stacking BLSEs (default: LrTb): LrTb Western writing mode: left to right, top to bottom RlTb Semitic writing mode: right to left, top to bottom TbRl Chinese and Japanese writing mode: top to bottom, right to left TbLr, LrBt, RlBt, BtRl, BtLr (PDF 2.0)

C++ Java C# **void end_item(int id)**

Perl PHP **end_item(int id)**

C **void PDF_end_item(PDF *p, int id)**

Close a structure element or other content item.

id The item's handle, which must have been retrieved with `PDF_begin_item()`.

Details Direct items must be closed before the end of the page. Regular items must be closed before the end of the document. However, it is strongly recommended to close all regular items as soon as they are completed. An item can only be closed if all of its children have been closed. After closing an item its parent becomes the active item.

Scope *page*; for grouping items also *document*; must always be paired with a matching `PDF_begin_item()` call. This method is only allowed in Tagged PDF mode.

C++ Java C# **void activate_item(int id)**

Perl PHP **activate_item(int id)**

C **void PDF_activate_item(PDF *p, int id)**

Activate a previously created structure element or other content item.

id Handle of a regular item which must have been retrieved with *PDF_begin_item()* and must not yet have been closed.

Details Suspending a structure element and activating it later gives additional flexibility for efficiently creating Tagged PDF pages even when there are multiple parallel structure branches on a page, e.g. with multi-column layouts or text inserts which interrupt the main text.

While the *parent* and *index* tagging options (see Table 14.2) can be used to insert structure elements at a specific location in the structure tree, *PDF_activate_item()* can be used to add more content to a previously created structure element.

In order to work around problems in Acrobat, no content items should be added immediately after calling *PDF_activate_item()*, but only other structure elements.

Scope *document, page*; this method is only allowed in Tagged PDF mode.

14.2 Abbreviated Tagging

Structure elements and artifacts can be created with `PDF_begin/end_item()` pairs. As an alternative, abbreviated tagging is available with the `tag` option of the following methods (see Table 14.4):

- ▶ `PDF_add_table_cell()` and the corresponding options in `PDF_add_table_cell()`: `fitgraphics`, `fitimage`, `fitpath`, `fitpdipage`, `fittextline`, `fittextflow`, `fitannotation`, `fitfield`
- ▶ `PDF_begin_document()`: abbreviated tagging can be used to create the root element of the structure hierarchy
- ▶ `PDF_create_annotation()`
- ▶ `PDF_create_field()`
- ▶ `PDF_draw_path()`
- ▶ `PDF_fit_graphics()`
- ▶ `PDF_fit_image()`
- ▶ `PDF_fit_pdi_page()` and `PDF_info_pdi_page()`
- ▶ `PDF_fit_table()`; abbreviated tagging triggers automatic table tagging
- ▶ `PDF_fit_textflow()`
- ▶ `PDF_fit_textline()`
- ▶ the `matchbox` option of various methods

Except for `PDF_begin_document()` these methods create page content items and therefore cannot be used to create grouping elements unless the grouping element is accompanied by another element via a nested `tag` option. Except for `PDF_begin_document()` and `PDF_add_table_cell()` abbreviated tagging can only be used in `page` scope. A detailed description of abbreviated tagging can be found in the PDFlib Tutorial.

Table 14.4 Option for abbreviated tagging in `PDF_add_table_cell()` and the corresponding `fit*` options in `PDF_add_table_cell()`, `PDF_create_annotation()`, `PDF_create_field()`, `PDF_draw_path()`, `PDF_fit_graphics()`, `PDF_fit_image()`, `PDF_fit_pdi_page()`, `PDF_fit_table()`, `PDF_fit_textflow()`, `PDF_fit_textline()`, and the `matchbox` option of various methods

option	description
<code>tag</code>	(Option list) Create a structure element or Artifact for the placed content. The suboptions listed in Table 14.2 can be used. Since the <code>tag</code> option is also allowed as suboption nested tag structures can be created with a single call.

14.3 Marked Content

Marked content methods can be used to mark content items with properties. They don't require Tagged PDF mode. `PDF_begin_item()` creates marked content internally as required.

C++ Java C# **`void begin_mc(String tagname, String optlist)`**

Perl PHP **`begin_mc(string tagname, string optlist)`**

C **`void PDF_begin_mc(PDF *p, const char *tagname, const char *optlist)`**

Begin a marked content sequence with optional properties.

tagname The name of the marked content sequence. The following tags are supported:

- ▶ All regular and pseudo items in Table 14.1.
- ▶ The tag name `Plib_custom` can be used for custom entries with user-defined properties.
- ▶ The tag name `Plib` is reserved.

optlist The following options for marked content sequences are supported:

- ▶ Options for standard properties of the marked content sequence; the following subset of tagging options in Table 14.2 can be used:
ActualText, Alt, artifactssubtype, artifacttype, contents, customtag, E, Lang, ListNumbering, Placement, tagname
- ▶ The tags `Plib_custom` and `Plib` additionally support the *properties* option in Table 14.5.
- ▶ Encoding option for C and Perl, PHP, Ruby with *stringformat=legacy*:
hypertextencoding (see Table 2.1)

Details A marked content sequence with the specified tag and properties is started. If no options are provided a sequence without any properties is created. Marked content sequences can be nested to an arbitrary level. The user is responsible for creating properly nested sequences of `PDF_begin/end_item()` and `PDF_begin/end_mc()`.

Scope *page, pattern, template, glyph*; must always be paired with a matching `PDF_end_mc()` call in the same scope.

Table 14.5 Option for user-defined properties of tags with `PDF_begin_mc()` and `PDF_mc_point()`

option	description
properties	(List of option lists; only for <code>tagname=Plib</code> and <code>tagname=Plib_custom</code>) Each list contains three options which specify a user-defined property:
key	(String; required) Name of the property
type	(Keyword; required) Type of the property value: boolean, name, or string
value	(Hypertext string if <code>type=string</code> , otherwise string; required) Value of the property

C++ Java C# **void end_mc()**

Perl PHP **end_mc()**

C **void PDF_end_mc(PDF *p)**

End the least recently opened marked content sequence.

Details All marked content sequences must be closed before calling [PDF_end_page_ext\(\)](#).

Scope *page, pattern, template, glyph*; must always be paired with a matching [PDF_begin_mc\(\)](#) call in the same scope.

C++ Java C# **void mc_point(String tagname, String optlist)**

Perl PHP **mc_point(string tagname, string optlist)**

C **void PDF_mc_point(PDF *p, const char *tagname, const char *optlist)**

Add a marked content point with optional properties.

tagname The name of the marked content point. The following tags are supported:

- ▶ All regular and pseudo items in Table 14.1.
- ▶ The tag name *Plib_custom* can be used for custom entries.
- ▶ The tag name *Plib* is reserved.

optlist The following options are supported:

- ▶ Options for standard properties of the marked content point according to Table 14.2.
- ▶ The tags *Plib_custom* and *Plib* additionally support the option in Table 14.5.

Details A marked content point with the specified tag name and properties will be created. If no options are provided a marked content point without any properties will be created.

Scope *page, pattern, template, glyph*



15 Document Interchange

API methods in this chapter:

- ▶ `PDF_set_info()`
- ▶ `PDF_add_portfolio_folder()`
- ▶ `PDF_add_portfolio_file()`
- ▶ `PDF_begin_dpart()`
- ▶ `PDF_end_dpart()`

15.1 Document Information Fields

C++ Java C# `void set_info(String key, String value)`

Perl PHP `set_info(string key, string value)`

C `void PDF_set_info(PDF *p, const char *key, const char *value)`

C `void PDF_set_info2(PDF *p, const char *key, const char *value, int len)`

Fill document information field *key* with *value*.

key (Name string) The name of the document info field, which may be any of the standard names (see Table 15.1) or an arbitrary custom name. There is no limit for the number of custom fields. Regarding use and semantics of custom document information fields the Dublin Core Metadata element set¹ is recommended.

value (Hypertext string) The string to which the *key* parameter will be set. Acrobat imposes a maximum length of *value* of 255 bytes.

len (C language binding only) Length of *value* in bytes. If *len* = 0 a null-terminated string must be provided.

Details The supplied key/value pair is written to the document info dictionary. If the *autoxmp* option in `PDF_begin/end_document()` is *true* PDFlib synchronizes standard info keys according to Table 15.1 to the corresponding XMP metadata properties. Except in PDF/A mode custom info keys are synchronized to the *pdfx* schema in XMP (which stands for »PDF Extension schema«, and is unrelated to the PDF/X standard). This method can be used to create simple XMP properties without supplying XMP via the *metadata* document option. However, XMP is much more powerful than simple key/value pairs.

Document info fields override corresponding properties in XMP document metadata supplied to the *metadata* option of `PDF_begin/end_document()`.

In PDF 2.0 the document info dictionary is deprecated and user-supplied fields are only emitted if the *emitdocinfo* option in `PDF_begin/end_document()` is *true*. Even if no user-supplied fields in the document info dictionary are emitted PDFlib synchronizes document info entries to XMP.

PDF/A Custom document info fields are not synchronized to XMP since custom metadata properties require an extension schema description in PDF/A (see PDFlib Tutorial).

1. See dublincore.org

PDF/UA An info field with *key=Title* must be supplied with a non-empty value. Alternatively, the *metadata* option of *PDF_begin_document()* with the *dc:title* XMP property can be supplied.

PDF/X Info fields with *key=Title* and *key=Creator* must be supplied with non-empty values. Alternatively in PDF/X-4 and PDF/X-5 the *metadata* option of *PDF_begin_document()* with the *dc:title* and *xmp:CreatorTool* XMP properties can be supplied.
Only the values *True* and *False* are allowed for the *Trapped* info field.

Scope any; if used in *object* scope the supplied values are only used for the next document.

Table 15.1 Keys for document information fields and corresponding XMP properties

key	description	corresponding XMP property
Author	Name of the person who created the document	dc:creator[0]
Creator	Software used to create the document (as opposed to the Producer of the PDF output which is always PDFlib).	xmp:CreatorTool
Keywords	Keywords describing the contents of the document	pdf:Keywords
Subject	Subject of the document	dc:description["x-default"]
Title	Title of the document	dc:title["x-default"]
Trapped	Indicates whether trapping has been applied to the document. Allowed values are <i>True</i> , <i>False</i> , and <i>Unknown</i> .	pdf:Trapped
any other name	User-defined document information field. A custom field name should only be supplied once. Fields which are used for standard identification are not allowed (e.g. <i>GTS_PDFXVersion</i>).	pdfx:<key>

15.2 XMP Metadata

XMP metadata can be supplied for the whole document or individual pages, fonts, ICC profiles, images, templates, and imported PDF pages. PDFlib always creates XMP for PDF 2.0 output. Table 15.2 lists suboptions for the *metadata* option of various methods.

Table 15.2 Suboptions for the *metadata* option in `PDF_begin/end_document()`, `PDF_begin/end_page_ext()`, `PDF_load_font()`, `PDF_load_image()`, `PDF_begin_template_ext()`, `PDF_open_pdi_page()`, the *templateoptions* option of `PDF_load_graphics()` and `PDF_begin_item()` as well as the *tag* option of various methods

option	description
associatedfiles	(List of asset handles; PDF 2.0) Asset handle(s) for file(s) associated with the metadata. The file(s) must have been loaded with <code>PDF_load_asset()</code> and <code>type=attachment</code> .
compress	(Boolean; not for <code>PDF_begin/end_document()</code>) Compress the XMP metadata stream in the PDF output. If the option is only supplied in <code>PDF_begin_page_ext()</code> but not in <code>PDF_end_page_ext()</code> , its value takes precedence over the default. Default: false PDF/A-1 and PDF/X: <code>compress=true</code> is not allowed.
inputencoding	(Keyword) The encoding to interpret the metadata supplied in filename. Default: unicode
inputformat	(Keyword) The format of the metadata supplied in filename. Default: utf8, but bytes if <code>inputencoding</code> is an 8-bit encoding
keepxmp	(Boolean; only for <code>PDF_load_image()</code> and <code>PDF_load_graphics()</code> ; can not be combined with <code>filename</code>) XMP metadata present in an image or graphics file will be kept, i.e. attached to the resulting image in the PDF document. XMP metadata is honored in the TIFF, JPEG, and JPEG 2000 image formats as well as in SVG graphics. If no valid XMP metadata is found in the image or graphics file this option doesn't have any effect. Default: false
filename	(Name string; required unless <code>keepxmp</code> is supplied) The name of a file containing well-formed XMP metadata. It will be interpreted according to the <code>filenamehandling</code> global option, see Table 2.1.
strict	(Boolean) If false, certain XMP violations are not reported as errors, but are silently fixed. If true, such violations trigger an exception. Default: false

PDF/A The XMP identification properties for PDF/A are created automatically.

PDFlib synchronizes relevant entries in user-supplied XMP streams to standard document info fields (similar to *autoxmp* mode which synchronizes document info fields to XMP). However, PDFlib does not synchronize other XMP entries to custom document info fields. Additional PDF/A requirements for XMP document metadata are discussed in the PDFlib Tutorial. The following validation is applied to XMP metadata:

- ▶ PDF/A-1: document-level XMP must conform to XMP 2004 or include an extension schema description. The schema description for document-level XMP can be supplied in `PDF_begin_document()` or `PDF_end_document()`.
- ▶ PDF/A-2/3: document-level and component-level (e.g. page) XMP must conform to XMP 2005 or include an extension schema description. The schema description for component-level XMP can be supplied with the respective component-level XMP (e.g. in `PDF_begin_page_ext()`) or with the document-level XMP in `PDF_begin_document()`.

PDF/UA The XMP identification properties for PDF/UA are created automatically.

PDF/VT The XMP identification properties for PDF/VT are created automatically.

PDF/X The XMP identification properties for PDF/X-4/5 are created automatically.

15.3 PDF Packages and Portfolios

Portfolio features are implemented with the following methods and options:

- ▶ Portfolios can be created with the *portfolio* option of `PDF_end_document()`. This method is described in Section 3.1, »Document Methods«, page 43; the *portfolio* option is described in Table 15.5.
- ▶ Files and folders can be added to a portfolio with `PDF_add_portfolio_folder()` and `PDF_add_portfolio_file()`. These methods are described below.
- ▶ Actions for navigating within a portfolio can be created with `PDF_create_action()` and *type=GoToE* (see Section 12.4, »Actions«, page 237).

C++ Java C# **`int add_portfolio_folder(int parent, String, foldername, String optlist)`**

Perl PHP **`int add_portfolio_folder(int parent, string foldername, string optlist)`**

`C int PDF_add_portfolio_folder(PDF *p, int parent, const char *foldername, int len, const char *optlist)`

Add a folder to a new or existing portfolio.

parent The parent folder, specified by a folder handle returned by an earlier call to `PDF_add_portfolio_folder()`, or -1 (in PHP: 0) for the root folder.

foldername (Hypertext string with 1-255 characters; the characters `/ \ : * " < > |` must not be used; the last character must not be a period `'.`.) Name of the folder. Two folders with the same parent must not have the same name after case normalization. The name of the root folder is ignored by Acrobat.

len (C language binding only) Length of *foldername* (in bytes). If *len=0* a null-terminated string must be provided.

optlist An option list specifying portfolio properties. The following options can be used:

- ▶ General options: *errorpolicy* (see Table 1.5)
- ▶ Options for folder properties according to Table 13.4: *description*, *thumbnail*
- ▶ Metadata option according to Table 15.3: *fieldlist*
- ▶ Encoding options for C and Perl, PHP, Ruby with *stringformat=legacy*: *hypertextencoding*, *hypertextformat* (see Table 2.1)

Returns A handle which can be used in `PDF_add_portfolio_folder()` or `PDF_add_portfolio_file()`.

Details The generated folder structure will be used to create a PDF portfolio for the current document. The folder structure will be deleted after `PDF_end_document()`. This method must not be used if the *attachments* option has been supplied to `PDF_begin_document()`.

Scope any except *object*; requires PDF 1.7ext3

Table 15.3 Option for `PDF_add_portfolio_folder()` and `PDF_add_portfolio_file()`

option	description
<i>fieldlist</i>	(List of option lists) Specify metadata fields for the file or folder. Each list refers to a field in the schema suboption of the <i>portfolio</i> option of <code>PDF_end_document()</code> . Supported suboptions are listed in Table 15.4.

C++ Java C# **`int add_portfolio_file(int folder, String filename, String optlist)`**

Perl PHP **`int add_portfolio_file(int folder, string filename, string optlist)`**

C **`int PDF_add_portfolio_file(PDF *p, int folder, const char *filename, int len, const char *optlist)`**

Add a file to a portfolio folder or a package.

folder A folder handle returned by an earlier call to `PDF_add_portfolio_folder()` or -1 (in PHP: 0) for the root folder. Folders different from the root folder require PDF 1.7ext3.

filename (Name string; will be interpreted according to the `filenamehandling` global option, see Table 2.1) Name of a disk-based or virtual file which will be attached to the specified folder of the PDF portfolio. With the `createpvf` option of `PDF_begin_document()` you can create documents in memory and pass them on for inclusion in a PDF Portfolio without creating any temporary files on disk.

Note that Acrobat will use the file name suffix to determine which application to launch when interacting with the file in Acrobat. If a file name with the appropriate suffix cannot be used due to external restrictions you can create a PVF file (which supports arbitrary file names) instead.

len (C language binding only) Length of `filename` (in bytes). If `len=0` a null-terminated string must be provided.

optlist An option list specifying file properties:

- ▶ Options for file properties according to Table 13.4:
description, filename, mimetype, name, password, relationship, thumbnail
- ▶ Metadata option according to Table 15.3: *fieldlist*
- ▶ Encoding option for C and Perl, PHP, Ruby with `stringformat=legacy`:
hypertextformat (see Table 2.1)

Returns The value 1 if the file could be added successfully, or an error code of -1 (in PHP: 0) if the method call failed. If `errorpolicy=exception` this method will throw an exception in case of an error. PDF documents will be opened to fetch the modification and creation dates. If the PDF document cannot be opened (e.g. because no password was supplied) the document will be included in the PDF portfolio nevertheless.

Details The specified file will be attached to the specified folder of a PDF 1.7ext3 portfolio or a PDF 1.7 package. If PDI is available, PDF documents will be opened if possible and their creation and modification dates will be written to the portfolio. This method must not be used if the attachments option has been supplied to `PDF_begin_document()`.

PDF/A PDF/A-1: this method must not be called.

PDF/A-2: `filename` must refer to a PDF/A-1 or PDF/A-2 document. Some options are restricted, see Table 13.4.

PDF/A-3: arbitrary file types can be added. The `relationship` option is required. Files added to a package are implicitly associated with the whole document.

Scope any except *object*; requires PDF 1.7

Table 15.4 Suboptions of the fieldlist option of `PDF_add_portfolio_folder()` and `PDF_add_portfolio_file()`

option	description
key	(String; required) Name of the field, which must refer to a key in the schema suboption of the portfolio option list of <code>PDF_end_document()</code> . The name must be unique.
prefix	(Hypertext string) A prefix string which will be prepended to the field value presented to the user. Acrobat will use this entry only if <code>type=text</code> . Default: none
type	(Keyword) Data type of the field. Supported keywords (default: text): text Text field: the field value will be stored as hypertext string. date Date field: the field value will be stored as PDF date string. number Number field: the field value will be stored as PDF number.
value	(Hypertext string; required) Value of a field in the schema suboption of the portfolio option list of <code>PDF_end_document()</code> . The data type must be specified in the type option and must match the corresponding type suboption of the schema suboption of the portfolio option.

Table 15.5 Suboptions of the portfolio option of `PDF_end_document()`

option	description
coversheet	(Hypertext string) The name of the file within the portfolio which will be initially presented in the user interface. Default: the document which contains the portfolio
coversheet-folder	(Folder handle) The name of the folder within the portfolio which contains the file specified in the coversheet option. If a file with the name specified in coversheet exists in multiple portfolio folders and no coversheetfolder has been specified, the first occurrence is used. Default: none
initialview	(Keyword) Initial view. Supported keywords (default: detail): detail The portfolio is presented in details mode, with all information in the schema option presented in a multi-column format. This mode provides the most information to the user (Acrobat: View, Portfolio, Details). hidden The portfolio is initially hidden, without preventing the user from obtaining a file list via explicit action (Acrobat: »Minimize view«). tile The portfolio is presented in tile mode, with each file in the collection denoted by a small icon and a subset of information from the schema option. This mode provides top-level information about the file attachments to the user (Acrobat: View, Portfolio, Layout).

Table 15.5 Suboptions of the portfolio option of `PDF_end_document()`

option	description
schema	<p>(List of option lists) Metadata schema for the portfolio: each option list defines a field with a unique name which corresponds to a key in the <code>fieldlist</code> of a folder or file, or to the name of a standard field. These fields define the display behavior of the portfolio in Acrobat (default: Acrobat displays the file name and size, modification date, and description if specified):</p> <p>editable (Boolean) Specifies whether Acrobat should allow editing the field value. Default: false</p> <p>key (String; required) The internal field name, which must be unique. The following names (which can not be used for user-defined fields) can be used to assign new labels to the builtin fields: <code>_creationdate</code>, <code>_description</code>, <code>_filename</code>, <code>_moddate</code>, <code>_size</code>.</p> <p>label (Hypertext string; required) The textual field label that is displayed to the user.</p> <p>order (Integer) Relative order of the fields in the user interface (1,2,3,...)</p> <p>type (Keyword) Data type of the field. The following types can be used to refer to user-defined fields in the <code>fieldlist</code> option (default: text):</p> <ul style="list-style-type: none"> text hypertext string date PDF date string number number <p>visible (Boolean) Initial visibility of the field in the user interface. Default: true; however, in the presence of user-defined fields Acrobat hides builtin fields unless they are explicitly specified as visible.</p>
sort	<p>(List of option lists, where each list contains a string and an optional keyword) Order in which the fields specified in the <code>schema</code> option will be sorted in the user interface. Each sublist contains the field name (required) and a keyword (optional). Supported keywords (Default: ascending):</p> <p>ascending field values are sorted in ascending order</p> <p>descending field values are sorted in descending order</p> <p>Acrobat uses this list to sort the fields in the portfolio. The list is used to allow additional fields to contribute to the sort, where each additional field is used to break ties: if multiple fields in the <code>schema</code> option have the same value for the first field in the list, the values for successive fields in the list are used for sorting until a unique order is determined or until the field names are exhausted. Default: no sorting</p>
split	<p>(Option list; PDF 1.7ext3) Orientation and position of the splitter bar. The default depends on the <code>initialview</code> option: The value <code>detail</code> (or no value) implies horizontal orientation and <code>tile</code> indicates vertical orientation. No splitter is used if <code>initialview=hidden</code>:</p> <p>direction (Keyword) Orientation of the splitter bar. Supported keywords:</p> <ul style="list-style-type: none"> horizontal Split the window horizontally. vertical Split the window vertically. none Don't split the window. The entire window is dedicated to the file navigation view. <p>position (Percentage; ignored if <code>direction=none</code>.) Initial position of the splitter bar, specified as a percentage of the available window area. Allowed values are in the range from 0 to 100. Default: viewer dependent</p>

15.4 Document Part Hierarchy

C++ Java C# **void begin_dpart(String optlist)**

Perl PHP **begin_dpart(string optlist)**

C **void PDF_begin_dpart(PDF *p, const char *optlist)**

Open a new node in the document part hierarchy (requires PDF/VT or PDF 2.0).

optlist An option list specifying document part hierarchy options according to Table 15.6: *associatedfiles*, *dpm*

PDF/VT This method and its counterpart **PDF_end_dpart()** must be called at least once when creating PDF/VT. The first call to **PDF_begin_dpart()** implicitly creates the root node of the document part (DPart) hierarchy. It is an error to call **PDF_begin_dpart()** more than once at the top level.

A call to **PDF_begin_dpart()** followed by a call to **PDF_begin_page_ext()** defines the start of the page range of a document part. All subsequent pages until the next call to **PDF_begin_dpart()** belong to the same document part. All calls together create the document part hierarchy as a tree structure which can contain two types of nodes:

- ▶ Inner nodes have one or more other nodes as descendants. The descendants may be inner nodes or leaf nodes, but a particular inner node may not contain descendants of both types.
- ▶ Leaf nodes describe the page(s) in a range. Leaf nodes never have descendant nodes.

The calls to **PDF_begin_dpart()** and **PDF_end_dpart()** must match when **PDF_end_document()** is called. If a document part hierarchy is to be created for the document, this method must be called at least once before calling **PDF_begin_page_ext()** for the first time. Calling **PDF_begin_dpart()** multiply creates deeper levels of the document part hierarchy. The generated depth of the document part hierarchy (i.e. the maximum nesting levels of **PDF_begin_dpart()**) must match the length of the list specified with the *nodenamelist* document option.

Scope *document*; this method must always be paired with a matching **PDF_end_dpart()** call.

Table 15.6 Options for **PDF_begin/end_dpart()**

option	description
associatedfiles	(List of asset handles; PDF 2.0) Asset handles for files associated with the DPart node. The files must have been loaded with PDF_load_asset() and <i>type=attachment</i> .
dpm	(POCA container handle; may be supplied to PDF_begin_dpart() or PDF_end_dpart() , but not to both methods for the same document part) Handle for a dictionary container created with PDF_poca_new() which contains document part metadata for the new node. The dictionary must have been created with the option <i>usage=dpm</i> .

C++ Java C# **void end_dpart(String optlist)**

Perl PHP **end_dpart(string optlist)**

C **void PDF_end_dpart(PDF *p, const char *optlist)**

Close a node in the document part hierarchy (requires PDF/VT or PDF 2.0).

optlist An option list specifying document part hierarchy options according to Table 15.6: *associatedfiles, dpm*

PDF/VT The first call to *PDF_end_dpart()* after *PDF_end_page_ext()* implicitly defines the end of the page range belonging to a leaf of the document part hierarchy. The calls to *PDF_begin_dpart()* and *PDF_end_dpart()* must match when *PDF_end_document()* is called.

Scope *document*; this method must always be paired with a matching *PDF_begin_dpart()* call.



A List of API Methods

This appendix lists all API methods. Click on a method name to jump to the corresponding description.

General	Font	Text Formatting	Color
get_errnum	load_font	set_text_option	setcolor
get_errmsg	close_font	fit_textline	load_iccprofile
get_apiname	setfont	info_textline	makespotcolor
get_opaque (C/C++ only)	info_font	add_textflow	create_devicen
convert_to_unicode	begin_font	create_textflow	shading
set_option	end_font	fit_textflow	shading_pattern
get_option	begin_glyph_ext	info_textflow	shfill
get_string	end_glyph	delete_textflow	begin_pattern_ext
new (C only)	encoding_set_char		end_pattern
delete		Table Formatting	
create_pvf	Simple Text Output	add_table_cell	Image and template
delete_pvf	set_text_pos	fit_table	load_image
info_pvf	show	info_table	close_image
download	show_xy	delete_table	fit_image
poca_new	continue_text		info_image
poca_delete	stringwidth	Matchboxes	begin_template_ext
poca_insert		info_matchbox	end_template_ext
poca_remove			
			SVG Graphics
Document and Page			load_graphics
begin_document			close_graphics
begin_document_callback (C/C++ only)			fit_graphics
end_document			info_graphics
get_buffer			
begin_dpart			
end_dpart			
begin_page_ext			
end_page_ext			
suspend_page			
resume_page			
define_layer			
set_layer_dependency			
begin_layer			
end_layer			

Graphics State*set_graphics_option**setlinewidth**save**restore**create_gstate**set_gstate***Coordinate Transformation***translate**scale**rotate**align**skew**concat**setmatrix***Path Construction***moveto**lineto**curveto**circle**arc**arcn**circular_arc**ellipse**elliptical_arc**rect**closepath***Path Painting and Clipping***stroke**closepath_stroke**fill**fill_stroke**closepath_fill_stroke**clip**endpath***Path Objects***add_path_point**draw_path**info_path**delete_path***PDI***open_pdi_document**open_pdi_callback*
(C/C++ only)*close_pdi_document**open_pdi_page**close_pdi_page**fit_pdi_page**info_pdi_page**process_pdi***pCOS***pcos_get_number**pcos_get_string**pcos_get_stream***Block Filling (PPS)***fill_textblock**fill_imageblock**fill_pdfblock**fill_graphicsblock***Interactive Features***create_action**add_nameddest**create_annotation**create_field**create_fieldgroup**create_bookmark**add_portfolio_folder**add_portfolio_file***Multimedia***load_3ddata**create_3dview**load_asset***Document Interchange***set_info**begin_item**end_item**activate_item**begin_mc**end_mc**mc_point*

B List of all Options and Keywords

This index contains an alphabetical list of options along with the API methods in which they can be used. Click on the page number to jump to the description.

&

&name option list macro call in `fit_textflow()` 102

3D

3Dactivate in `create_annotation()` 259

3Ddata in `create_annotation()` 259

3Dinitialview in `create_annotation()` 259

3Dinteractive in `create_annotation()` 259

3Dshared in `create_annotation()` 259

3Dview in `create_action()` 238

A

acceptdynamicxfa in `open_pdi_document()` 194

acrobat in `info_font()` 73

action

in `begin/end_page_ext()` 56

in `create_annotation()` 223

in `create_bookmark()` 220

in `create_field/group()` 231

in `end_document()` 45

in `process_pdi()` 206

activate suboption of richmedia in

`create_annotation()` 251

activeitemid

keyword in `get_option()` 29

keyword in `info_matchbox()` 134

activeitemindex keyword in `get_option()` 29

activeitemisinline keyword in `get_option()` 30

activeitemkidcount keyword in `get_option()` 30

activeitemname keyword in `get_option()` 30

activeitemstandardname keyword in

`get_option()` 30

actual in `info_font()` 73

actualtext

in `begin_item()` and the tag option 267

in `set_text_option()`, `fit_textline()`, and

`fill_textblock()` 76

addfitbox suboption for wrap in `fit_textflow()` 110

addpath keyword in `add_path_point()` 152

adjustmethod in `add/create_textflow()` 99

adjustpage

in `fit_image/fit_graphics/fit_pdi_page()` 176

in `fit_pdi_page()` 201

advancedlinebreak in `add/create_textflow()` 99

align

in `draw_path()` 125

keyword for transform in

`begin_pattern_ext()`, `begin_template_ext()`,

`shading_pattern()` and `open_pdi_page()` 191

suboption for the window suboption of

`PDF_load_asset()` 248

alignchar in `fit/info_textline()` 125

alignment

in `add/create_textflow()` 97

in `create_annotation()` 223

suboption for leader in `fit/info_textline()` and

`add/create_textflow()` 92

alpha keyword for the type suboption of

`softmask` in `create_gstate()` 141

alphachannelname in `load_image()` 173

alphaisshape in `create_gstate()` 140

Alt in `begin_item()` and the tag option 267

alternate in `create_devicen()` 165

alltext in `load_asset()` 247

angle keyword in `info_textline()` 94

angularunit suboption for georeference 260

animation

in `load_3d()` 258

suboption for the activate suboption of

`richmedia` in `create_annotation()` 252

annotation suboption for `targetpath` in

`create_action()` 241

annotationtype in `add_table_cell()` and

suboption for `caption` 116

annotcolor in `create_annotation()` 224

antialias in `shading()` and suboption of the

`shading graphics appearance` option 167

api in `info_font()` 73

area suboption for fill in `fit_table()` 119

areaunit suboption for georeference 260

ARIARole in `begin_item()` and the tag option 267

artbox in `begin/end_page_ext()` 56

artifactssubtype in `begin_item()` and the tag

option 267

artifacttype in `begin_item()` and the tag option

267

ascender

in `info_font()` 72

in `load_font()` 66

keyword in `info_textline()` 94

asciifile in `set_option()` 26

assets

suboption of richmedia in

`create_annotation()` 251

associatedfiles

- in `begin_item()` and the tag option 265
- in `begin/end_dpart()` 282
- in `begin/end_page_ext()` 56
- in `create_annotation()` 224
- in `end_document()` 45
- in `load_image()`, `load_graphics()`, `open_pdi_page()`, and `begin_template_ext()` 190
- suboption for the metadata option of various API methods 277

Attached in `begin_item()` and the tag option 267

attachment in `create_annotation()` 224

attachmentpassword in `begin_document()` 50

attachmentpoint in `draw_path()` 125

attachments in `begin/end_document()` 45

autoplay in `load_asset()` 247

autospace in `set_option()` 26

autosubsetting in `load_font()` 66

autoxmp in `begin/end_document()` 45

avoidbreak in `add/create_textflow()` 99

avoiddemostamp in `set_option()` 26

avoidemptybegin in `add/create_textflow()` 98

avoidwordsplitting

- in `add_table_cell()` 114

- in `fit_textflow()` 106

B

backdropcolor suboption of `softmask` in `create_gstate()` 141

background in `create_3dview()` 255

backgroundcolor

- in `create_field/group()` 231

barcode in `create_field/group()` 231

baseurl in `load_asset()` 247

BBox in `begin_item()` and the tag option 267

bboxexpand

- in `draw_path()` 155

- in `load_graphics()` 180

bboxwidth, bboxheight keywords in `info_path()` 156

begoptlistchar in `create_textflow()` 103

beziers suboption for `wrap` in `fit_textflow()` 110

bleedbox in `begin/end_page_ext()` 56

blendmode

- in `create_annotation()` 224

- in `create_gstate()` 140

blind

- in `fit_table()` 118

- in `fit_textflow()` 106

- in many methods 125

block in `process_pdi()` 206

blockname suboption of `block` in `process_pdi()` 206

blocks in `begin/end_page_ext()` 56

bookmark in `begin_item()` and the tag option 265

bordercolor in `create_field/group()` 231

borderstyle

- in `create_annotation()` 224

- in `create_field/group()` 231

borderwidth in several methods 136

bottom in `add_nameddest()` and suboption for destination in `create_action()`, `create_annotation()`, `create_bookmark()` and `begin/end_document()` 242

boundingbox

- in `begin_glyph_ext()` 87

- in `begin_pattern_ext()` and

- `begin_template_ext()` 187

- in `draw_path()` 155

- in `shading()` and suboption of the shading graphics appearance option 167

- keyword in `info_*`() 130

- keyword in `info_matchbox()` 134

- keyword in `info_textflow()` 111

- suboption for viewports in `begin/end_page_ext()` 260

bounds suboption for `georeference` 260

boxes suboption for `wrap` in `fit_textflow()` 110

boxexpand in `open_pdi_page()` 199

boxheight suboption for `matchbox` 132

boxlinecount keyword in `info_textflow()` 111

boxsize in various methods 126

boxwidth suboption for `matchbox` 132

bpc in `load_image()` 175

buttonlayout in `create_field/group()` 232

buttonstyle in `create_field/group()` 232

C

calcolor in `create_field/group()` 232

calloutline in `create_annotation()` 224, 228

camerazworld in `create_3dview()` 255

cameradistance in `create_3dview()` 255

canonicaldate in `create_action()` 238

canresize suboption for the window suboption of `PDF_load_asset()` 248

capheight

- in `info_font()` 72

- in `load_font()` 66

- keyword in `info_textline()` 94

caption

- in `create_field/group()` 232

- in `fit_table()` 118

- suboption for `barcode` in `create_field/group()` 236

captiondown in `create_field/group()` 232

captionoffset in `create_annotation()` 228

captionposition in `create_annotation()` 228

captionrollover in `create_field/group()` 232

centerwindow suboption for `viewerpreferences` in `begin/end_document()` 53

charclass in `add/create_textflow()` 100

charmapping in `add/create_textflow()` 101

charref
in `set_option()` 26
in `set_text_option()`, `fit/info_textline()`,
`fill_textblock()` and `add/create_textflow()` 76

charspacing
in `create_field/group()` 232
in `set_text_option()`, `fit/info_textline()`,
`fill_textblock()` and `add/create_textflow()` 77

checkcolorspace keyword in `info_image()` 178

Checked in `begin_item()` and the tag option 267

checkoutputintentprofile in
`open_pdi_document()` 194

checktags
in `begin_document()` 49
in `open_pdi_document()` 194

checktransgroupprofile in `open_pdi_page()` 199

children in `set_layer_dependency()` 63

chromakey in `load_image()` 173

cid in `info_font()` 71, 72

cidfont in `info_font()` 72

circle keyword in `add_path_point()` 152

circles suboption for `wrap` in `fit_textflow()` 110

circular keyword in `add_path_point()` 152

classes suboption for `logging` in `set_option()` 19

clip in `draw_path()` 155

clipannotations in `fit_pdi_page()` 202

clipping suboption for `matchbox` 132

clippingarea in `open_pdi_page()` 200

clippingpath keyword in `info_image()` 178

clippingpathname in `load_image()` 173

cliprule in several methods 136

clockwise
in `add_path_point()` 153
in `elliptical_arc()` 149

cloneboxes
in `fit_pdi_page()` 202
in `open_pdi_page()` 200

close
in `add_path_point()` 153
in `draw_path()` 155

cloudy in `create_annotation()` 224

CMap in `set_option()` 26

code
in `info_font()` 71, 72

codepage in `info_font()` 72

codepagelist in `info_font()` 72

colorize in `load_image()` 173

colored in `begin_glyph_ext()` 87

colormode
in `info_font()` 72
in `load_font()` 67

colortype
in `info_font()` 73

colscalegroup in `add_table_cell()` 114

colspan in `add_table_cell()` 114

ColSpan in `begin_item()` and the tag option 267

colwidth in `add_table_cell()` 114

colwidthdefault in `fit_table()` 118

comb in `create_field/group()` 232

comment in `add/create_textflow()` 100

commitonselect in `create_field/group()` 232

compatibility in `begin_document()` 48

components in `load_image()` 175

compress
in `set_option()` 26
suboption for `metadata` 277

condition suboption for the `activate` suboption of
`richmedia` in `create_annotation()` 252

configuration suboption of `richmedia` in
`create_annotation()` 251

containertype in `poca_new()` 39

contents
in `begin_mc()` 265
in `create_annotation()` 224

ContinuedFrom in `begin_item()` and the tag
option 267

ContinuedList in `begin_item()` and the tag option
267

continuetextflow in `add_table_cell()` 114

control keyword in `add_path_point()` 152

controller in `load_asset()` 247

convert in `pcos_get_stream()` 209

convertlinks in `fit_graphics()` 184

copy in `create_pvf()` 34

copyglobals in `load_image()` 175

count keyword in `info_matchbox()` 134

coversheet suboption for `portfolio` in
`end_document()` 280

coversheetfolder suboption for `portfolio` in
`end_document()` 280

crease suboption for `rendermode` in
`create_3dview()` 257

createdate in `create_annotation()` 224

createfittext in `fit_textflow()` 106

createlastindent in `fit_textflow()` 106

creatematchboxes suboption for `wrap` in
`fit_textflow()` 110

createoutput in `begin_document()` 51

createpvf
in `begin_document()` 51
in `fetch_data()` 37

createrichtext in `create_annotation()` 224

createwrapbox suboption for `matchbox` 132

creatorinfo in `define_layer()` 61

cropbox in `begin/end_page_ext()` 56

ctm_a/b/c/d/e/f keywords in `get_option()` 30

currentvalue in `create_field/group()` 232

currentx/y keywords in `get_option()` 30

curve keyword in `add_path_point()` 152

custom in `create_annotation()` 225

customtag in `begin_item()` and the tag option 265

D

dasharray

in `add_path_point()` 153
in `create_annotation()` 225
in `create_field/group()` 232
in `create_gstate()` 140
in `set_text_option()`, `fit/info_textline()`,
`fill_textblock()` and `add/create_textflow()` 77
in several methods 136

dashphase

in `add_path_point()` 153
in `create_gstate()` 140
in several methods 136

dataprep suboption for barcode in `create_field/group()` 236

datestring in `create_annotation()` 225

deactivate suboption of richmedia in
`create_annotation()` 251

debugshow in `fit_table()` 118

decode in `load_image()` 173

decorationabove in `set_text_option()`, `fit/info_textline()`, `fill_textblock()` and `add/create_textflow()` 77

defaultcmyk

in `begin_font()` 86
in `begin_page_ext()` 56
in `load_image()`, `load_graphics()`,
`open_pdi_page()`, and `begin_template_ext()`
190

defaultfontfamily in `load_graphics()` 180

defaultfontoptions in `load_graphics()` 180

defaultgray

in `begin_font()` 86
in `begin_page_ext()` 56
in `load_image()`, `load_graphics()`,
`open_pdi_page()`, and `begin_template_ext()`
190

defaultimageoptions in `load_graphics()` 180

defaultrgb

in `begin_font()` 86
in `begin_page_ext()` 56
in `load_image()`, `load_graphics()`,
`open_pdi_page()`, and `begin_template_ext()`
190

defaultstate in `define_layer()` 61

defaultvalue in `create_field/group()` 232

defaultview in `load_3d()` 258

depend in `set_layer_dependency()` 63

Desc in `begin_item()` and the tag option 267

descender

in `info_font()` 73
in `load_font()` 67
keyword in `info_textline()` 94

description

in `load_asset()` and suboption for other
methods 249
in `load_iccprofile()` 161
keyword in `info_graphics()` 185

destination

in `begin/end_document()` 45
in `create_action()` 238
in `create_annotation()` 225
in `create_bookmark()` 220

destname

in `create_action()` 238
in `create_annotation()` 225
in `create_bookmark()` 220
in `end_document()` 45
suboption for `targetpath` in `create_action()`
241

devicencolors in `load_graphics()` 180

direct

in `begin_item()` and the tag option 265
in `poca_insert()` 41

direction suboption for `viewerpreferences` in
`begin/end_document()` 53

disable

suboption for `3Dactivate` in
`create_annotation()` 259
suboption for logging in `set_option()` 18
suboption for network option lists 38
suboption for shadow in `add/create_textflow()` 78

disablestate suboption for `3Dactivate` in
`create_annotation()` 259

display

in `create_annotation()` 225
in `create_field/group()` 232

displaydoctitle suboption for `viewerpreferences`
in `begin/end_document()` 53

displaysystem suboption for `georeference` 261

documentattachment in `load_asset()` and
suboption for other methods 249

doubleadapt suboption for `matchbox` 132

doubleoffset suboption for `matchbox` 132

down suboption for `template` in
`create_annotation()` 227

downloadlifetime in `load_graphics()` 181

dpi in `load_image()` 126

dpm in `begin/end_dpart()` 282

drawbottom, **drawleft**, **drawright**, **drawtop**
suboptions for `matchbox` 132

duplex suboption for `viewerpreferences` in `begin/end_document()` 53

duration

in `begin/end_page_ext()` 56
in `create_action()` 238
in `load_asset()` 247

E

E in `begin_item()` and the tag option 267

ecc suboption for barcode in `create_field/group()`
236

editable in `create_field/group()` 232

elementid in `begin_item()` and the tag option 265

ellipse keyword in `add_path_point()` 152

elliptical keyword in `add_path_point()` 152

embedding in `load_font()` 67

embedprofile in `load_iccprofile()` 161

emitdocinfo in `begin/end_document()` 45

enable
 suboption for `3Dactivate` in `create_annotation()` 259
 suboption for logging in `set_option()` 18

enablestate suboption for `3Dactivate` in `create_annotation()` 259

encoding
 in `info_font()` 72, 73
 in `load_font()` 67

Encoding in `set_option()` 26

end
 suboption for `matchbox` 132
 suboption of the shading graphics appearance option 167

endcolor in `shading()` and suboption of the shading graphics appearance option 167

endingstyles in `create_annotation()` 225

endoptlistchar in `create_textflow()` 103

endx, endy keywords in `info_textline()` 94

enforce suboption for viewerpreferences in `begin/end_document()` 53

entire suboption for background in `create_3dview()` 255

enumeratefonts in `set_option()` 26

environment suboption for pdfvnt in `load_image(), load_graphics(), open_pdi_page()` and `begin_template_ext()` 192

epsq suboption for the coords and displaycoords suboptions of georeference 261

errorconditions in `load_graphics()` 181

errorpolicy in various methods 19

escapesequence
 in `set_option()` 26
 in `set_text_option(), fit/info_textline(), fill_textblock()` and `add/create_textflow()` 76

exceedlimit suboption for `matchbox` 132

exchangefillcolors in `fit_textflow()` 106

exchangestrokecolors in `fit_textflow()` 106

exclude in `create_action()` 238

exists keyword in `info_matchbox()` 134

exportable in `create_field/group()` 232

exportmethod in `create_action()` 239

extendo, extend1 in `shading()` and suboption of the graphics appearance option 167

external
 in `load_asset()` and suboption for other methods 249
 in `load_asset()` with `type=rendition` 247

externalrefs in `load_graphics()` 181

F

facecolor suboption for `rendermode` in `create_3dview()` 257

fakebold in `set_text_option(), fit/info_textline(), fill_textblock()` and `add/create_textflow()` 77

faked in `info_font()` 72

fallbackfont in `info_font()` 73

fallbackfontfamily in `load_graphics()` 181

fallbackfontoptions in `load_graphics()` 181

fallbackfonts in `load_font()` 67

fallbackheight/fallbackwidth in `load_graphics()` 181

fallbackimage in `load_graphics()` 182

familyname
 in `begin_font()` 86
 in `info_font()` 73

feature in `info_font()` 73

featurelist in `info_font()` 73

features in `fit/info_textline(), fill_textblock()` and `add/create_textflow()` 93

fieldcontent in `create_field/group()` 232

fieldlist in `add_portfolio_folder()` 278

fieldname in `add_table_cell()` and suboption for `caption` 116

fieldtype
 in `add_table_cell()` and suboption for `caption` 116
 in `create_fieldgroup()` 233

filemode in `begin_document()` 51

filename
 in `create_action()` 239
 in `load_asset()` and suboption for other methods 249
 keyword in `info_graphics()` 185
 keyword in `info_image()` 178
 suboption for logging in `set_option()` 18
 suboption for metadata 277
 suboption for search in `begin/end_document()` 47

filenamehandling in `set_option()` 27

fileselect in `create_field/group()` 233

fill
 in `add_path_point()` 153
 in `draw_path()` 155
 in `fit_table()` 119

fillcolor
 in `add_path_point()` 153
 in `create_annotation()` 225
 in `create_field/group()` 233
 in `set_text_option(), fit/info_textline(), fill_textblock()` and `add/create_textflow()` 77
 in several methods 136
 suboption for background in `create_3dview()` 255
 suboption for leader in `fit/info_textline()` and `add/create_textflow()` 92
 suboption for shadow in `add/create_textflow()` 78

fillrule
in `add_path_point()` 153
in several methods 136
suboption for `wrap` in `fit_textflow()` 110

firstbodyrow keyword in `info_table()` 121

firstdraw in `fit_table()` 119

firstlinedist
in `fit_textflow()` 106
keyword in `info_textflow()` 111

firstparalinecount keyword in `info_textflow()` 111

fitannotation in `add_table_cell()` and suboption for caption 116

fitfield in `add_table_cell()` 116

fitgraphics in `add_table_cell()` and suboption for caption 115

fitheight keyword for the type option for `add_nameddest()` as well as for destination 243

fitimage in `add_table_cell()` and suboption for caption 115

fitmethod
in `create_field/group()` 233
in `fit_textflow()` 107
in various methods 126
suboption for template in `create_annotation()` 227

fitpath in `add_table_cell()` and suboption for caption 115

fitpdipage in `add_table_cell()` and suboption for caption 115

fitrect keyword for type for `add_nameddest()` as well as for destination 243

fitscalex, fitscaley keywords in `info_*`() 130

fittext keyword in `info_textflow()` 111

fittextflow in `add_table_cell()` and suboption for caption 115

fittextline in `add_table_cell()` and suboption for caption 116

fittingpossible
keyword in `info_graphics()` 185
keyword in `info_pdi_page()` 204

fitvisible keyword for the type option of `add_nameddest()` as well as for destination 243

fitvisibleheight, fitvisiblewidth keywords for type option of `add_nameddest()` as well as for destination 243

fitwidth keyword for type in `add_nameddest()`, as well as for destination 243

fitwindow
keyword for type in `add_nameddest()`, as well as for destination 243
suboption for viewerpreferences in `begin/end_document()` 53

fixed keyword for type in `add_nameddest()` as well as for destination 243

fixedleading in `add/create_textflow()` 98

fixedtextformat in `create_textflow()` 103

flatness
in `add_path_point()` 153
in `create_gstate()` 140
in several methods 136

flush
in `begin_document()` 51
suboption for logging in `set_option()` 18

font
in `create_annotation()` 225
in `create_field/group()` 233
in `set_text_option()`, `fit/info_textline()`, `fill_textblock()` and `add/create_textflow()` 77
suboption for leader in `fit/info_textline()` and `add/create_textflow()` 92

fontfile in `info_font()` 73

fontname
in `info_font()` 73
in `load_font()` 67

FontnameAlias in `set_option()` 27

FontOutline in `set_option()` 27

fontscale
in `fit_textflow()` 107
keyword in `info_textflow()` 111

fontsize
in `create_annotation()` 225
in `create_field/group()` 233
in `info_font()` 72
in `set_text_option()`, `fit/info_textline()`, `fill_textblock()` and `add/create_textflow()` 78
suboption for leader in `fit/info_textline()` and `add/create_textflow()` 92

fontstyle
in `create_bookmark()` 220
in `load_font()` 67

fonttype in `info_font()` 73

footer in `fit_table()` 119

forcebox in `open_pdi_page()` 200

forcedheight/forcedwidth in `load_graphics()` 182

forcesrgb in `load_graphics()` 182

foregroundcolor in `load_font()` 68

foregroundopacity in `load_font()` 68

full in `info_font()` 73

functionname in `create_action()` 239

G

georeference
in `load_image()` 190
suboption for viewports in `begin/end_page_ext()` 260

glyphcheck
in `set_option()` 27
in `set_text_option()`, `fit/info_textline()`, `fill_textblock()` and `add/create_textflow()` 76

glyphid in `info_font()` 71, 73

glyphname
in `begin_glyph_ext()` 87
in `info_font()` 71, 73

graphics in `add_table_cell()` and suboption for caption 116

graphicsheight, graphicswidth keywords in `info_graphics()` 185

group

- in `begin_page_ext()` 56
- in `resume_page()` 60
- in `set_layer_dependency()` 63
- option in `add_nameddest()` and suboption for destination in `create_action()`, `create_annotation()`, `create_bookmark()` and `begin/end_document()` 242
- suboption for labels in `begin_document()` 52

groups in `begin_document()` 45

gstate

- in `add_path_point()` 153
- in `fit_image/fit_graphics/pdi_page()` 177
- in `fit_pdi_page()` 201
- in `fit_table()` 119
- in `fit/info_textline()` and `add/create_textflow()` 107
- in many graphics methods 136
- in `set_text_option()`, `fit/info_textline()`, `fill_textblock()` and `add/create_textflow()` 78
- in `shading_pattern()` 168
- suboption for shadow in `add/create_textflow()` 78

H

halftoneorigin in `create_gstate()` 140

hasclose suboption for the window suboption of `PDF_load_asset()` 248

hastitle suboption for the window suboption of `PDF_load_asset()` 248

header in `fit_table()` 119

Headers in `begin_item()` and the tag option 268

height

- in `add_path_point()` 153
- in `begin_template_ext()` 187
- in `begin/end_page_ext()` 57
- in `load_image()` 175
- keyword in `info_*`() 130
- keyword in `info_matchbox()` 134
- suboption for the window suboption of `PDF_load_asset()` 249

Height in `begin_item()` and the tag option 268

hide in `create_action()` 239

hidemenubar suboption for viewerpreferences in `begin/end_document()` 53

hidetoolbar suboption for viewerpreferences in `begin/end_document()` 53

hidewindowui suboption for viewerpreferences in `begin/end_document()` 53

highlight

- in `create_annotation()` 225
- in `create_field/group()` 233

honorioclipingpath in `load_image()` 173

honoriccprofile

- in `load_graphics()` 182
- in `load_image()` 173

horboxgap keyword in `info_table()` 121

horzscaling in `set_text_option()`, `fit/info_textline()`, `fill_textblock()` and `add/create_textflow()` 78

horshrinking keyword in `info_table()` 121

horshrinklimit in `fit_table()` 119

hortabmethod in `add/create_textflow()` 98

hortabsize in `add/create_textflow()` 98

hostfont in `info_font()` 73

HostFont in `set_option()` 27

httpauthentication suboption for network option lists 38

hypertextencoding

- in `set_option()` and various other methods 28
- suboption for labels in `begin/end_document()` and label in `begin/end_page_ext()` 52
- suboption for viewports in `begin/end_page_ext()` 260

hypertextformat in `set_option()` and various other methods 28

hyphenchar in `add/create_textflow()` 101

I

icccomponents keyword in `get_option()` 30

iccprofile

- in `get_option()` 30
- in `load_image()` 173
- keyword in `info_image()` 178

ICCProfile in `set_option()` 27

iccprofilecmyk, iccprofilegray, iccprofilergb

- in `load_graphics()` 182
- in `set_option()` 27

icon in `create_field/group()` 233

icondown in `create_field/group()` 233

iconname

- in `create_annotation()` 225
- in `load_image()`, `load_graphics()`, `open_pdi_page()` and `begin_template_ext()` 190

iconrollover in `create_field/group()` 233

id in `begin_item()` and the tag option 265

ifoffscreen suboption for the window suboption of `PDF_load_asset()` 249

ignoreclippingpath in `fit_image()` 177

ignoremask in `load_image()` 174

ignoreorientation

- in `fit_image()` 177
- in `load_image()` 174

ignorepdfversion in `open_pdi_page()` 200

image in `add_table_cell()` and suboption for caption 116

imagehandle in `load_image()` 175

imageheight keyword in `info_image()` 178

imagemask keyword in `info_image()` 178

imagetype keyword in `info_image()` 178

imagewidth keyword in `info_image()` 178

includepid, includetid, includeoid suboption for logging in `set_option()` 18

index

- in `begin_item()` and the tag option 265
- in `create_bookmark()` 220
- in `info_font()` 74
- in `info_pdi_page()` 204
- in `poca_insert()` and `poca_remove()` 41

indextype suboption for search in `begin/end_document()` 47

infomode

- in `load_image()` 174
- in `open_pdi_document()` 194
- keyword in `info_image()` 178

initgraphicsstate in `open_pdi_page()` 200

initialexportstate in `define_layer()` 61

initialprintstate in `define_layer()` 62

initialsubset in `load_font()` 68

initialview suboption for portfolio in `end_document()` 280

initialviewstate in `define_layer()` 62

inittextstate

- in `set_text_option()`, `fit/info_textline()`, `fill_textblock()` and `add/create_textflow()` 78
- in `set_text_state()` 136

inline

- in `load_graphics()` 182
- in `load_image()` 175

inlineoptions in `add/create_textflow()` 100

inmemory

- in `begin_document()` 51
- in `open_pdi_document` 194

innerbox suboption for `matchbox` 133

inputencoding suboption for metadata 277

inputformat suboption for metadata 277

inreplyto in `create_annotation()` 225

instance in `create_action()` 239

intent in `define_layer()` 62

interiorcolor in `create_annotation()` 226

interpolate in `load_image()` 174

inversefill suboption for `wrap` in `fit_textflow()` 110

invert

- in `load_image()` 174
- suboption of `softmask` in `create_gstate()` 141

ismap in `create_action()` 239

istemplate keyword in `info_graphics()` 185

italicangle

- in `info_font()` 73
- in `set_text_option()`, `fit/info_textline()`, `fill_textblock()` and `add/create_textflow()` 78

item in `create_bookmark()` 220

itemname in `create_field/group()` 233

itemnamelist in `create_field/group()` 233

itemtextlist in `create_field/group()` 234

J

javascript

- in `create_action()` 239
- in `load_3ddata()` 258

justifymethod in `fit/info_textline()` 90

K

keepfilter in `pcos_get_stream()` 209

keepfont in `load_font()` 68

keephandles in `delete_table()` 122

keepxmp suboption for metadata 277

kernamount in `info_font()` 74

Kerning

- in `set_option()` 27
- in `set_text_option()`, `fit/info_textline()`, `fill_textblock()` and `add/create_textflow()` 78

kerningpairs in `info_font()` 74

key

- in `poca_insert()` and `poca_remove()` 41
- suboption for custom in `create_annotation()` 225
- suboption for `fieldlist` in `add_portfolio_file/folder()` 280
- suboption for properties in `begin_mc()` and `mc_point()` 272

L

label in `begin/end_page_ext()` 57

labels in `begin/end_document()` 45

lang

- in `begin_document()` 49
- in `create_annotation()` 226
- in `load_graphics()` 182
- keyword in `info_pdi_page()` 204

Lang in `begin_item()` and the tag option 268

language

- in `define_layer()` 62
- in `fit/info_textline()`, `fill_textblock()` and `add/create_textflow()` 93
- in `info_font()` 73

largearc

- in `add_path_point()` 153
- in `elliptical_arc()` 149

lastalignment in `add/create_textflow()` 98

lastbodyrow keyword in `info_table()` 121

lastfont keyword in `info_textflow()` 111

lastfontsize keyword in `info_textflow()` 111

lastlinedist

- in `fit_textflow()` 107
- keyword in `info_textflow()` 111

lastmark keyword in `info_textflow()` 111

lastparalinecount keyword in `info_textflow()` 111

layer

- in `create_annotation()` 226
- in `create_field/group()` 234
- in `load_image()`, `load_graphics()`,

`open_pdi_page()` and `begin_template_ext()`
190

layerstate in `create_action()` 239

leader
in `add/create_textflow()` 98
in `fit/info_textline()` 90

leaderlength in `create_annotation()` 228

leaderoffset in `create_annotation()` 228

leading
in `set_text_option()`, `fit/info_textline()`,
`fill_textblock()` and `add/create_textflow()` 78
keyword in `info_textflow()` 111

left option in `add_nameddest()` and suboption for
destination in `create_action()`,
`create_annotation()`, `create_bookmark()` and
`begin/end_document()` 242

leftindent in `add/create_textflow()` 98

leftlinex, **leftliney** keywords in `info_textflow()` 111

license in `set_option()` 27

licensefile in `set_option()` 27

lighting in `create_3dview()` 255

limitcheck in `begin_document()` 48

line
keyword in `add_path_point()` 152
suboption for stroke in `fit_table()` 120

linearize in `begin_document()` 46

linearunit suboption for georeference 261

linecap
in `add_path_point()` 153
in `create_gstate()` 140
in several methods 136

linegap
in `info_font()` 74
in `load_font()` 68

lineheight suboption for wrap in `fit_textflow()`
110

linejoin
in `add_path_point()` 153
in `create_gstate()` 140
in several methods 137

linespreadlimit in `fit_textflow()` 107

linewidth
in `add_path_point()` 153
in `create_annotation()` 226
in `create_field/group()` 234
in `create_gstate()` 140
in several methods 137

ListNumbering in `begin_item()` and the tag option
268

locale in `add/create_textflow()` 99

locked
in `create_annotation()` 226
in `create_field/group()` 234

lockedcontents in `create_annotation()` 226

lockmode in `create_field/group()` 234

logging in `set_option()` 27

luminosity keyword for the type suboption of
`softmask` in `create_gstate()` 141

M

macro option list macro definition in
`fit_textflow()` 102

maingid in `info_font()` 74

major keyword in `get_option()` 30

mappoints suboption for georeference 261

margin
in `add_table_cell()` 114
in `fit_textline()` 126
suboption for `matchbox` 133

marginbottom, **marginleft**, **marginright**,
martop in `add_table_cell()` 114

mark in `add/create_textflow()` 100

mask in `load_image()` 174

masked in `load_image()` 174

masterpassword in `begin_document()` 50

matchbox
in `add_table_cell()` and suboption for caption
116
in `add/create_textflow()` 100
in several methods 126
suboption for `createlastindent` in
`fit_textflow()` 106

matrix keyword for transform in
`begin_pattern_ext()`, `begin_template_ext()`,
`shading_pattern()` and `open_pdi_page()` 191

maxchar in `create_field/group()` 234

maxcode in `info_font()` 74

maxlinewidth keyword in `info_textflow()` 111

maxlines in `fit_textflow()` 107

maxliney keyword in `info_textflow()` 111

maxspacing in `add/create_textflow()` 99

maxvunicode in `info_font()` 74

mediabox in `begin/end_page_ext()` 57

menuname in `create_action()` 239

metadata 277
in `begin_item()` and the tag option 265
in `begin/end_document()` 46
in `begin/end_page_ext()` 57
in `load_font()` 68
in `load_image()`, `load_graphics()`,
`open_pdi_page()` and `begin_template_ext()`
190
keyword in `info_graphics()` 185

mimetype
in `load_asset()` and suboption for other
methods 249
in `load_asset()` with `type=rendition` 247

minfontsize in `fit_textflow()` 107, 126

mingapwidth in `fit_textflow()` 107

minlinecount in `add/create_textflow()` 98

minlinelength keyword in `info_textflow()` 111

minliney keyword in `info_textflow()` 111

minor keyword in `get_option()` 30

minrowheight in `add_table_cell()` 114

minspacing in `add/create_textflow()` 99

minvunicode in `info_font()` 74

mirroringx, mirroringy
 keywords in `info_image()` 178
 keywords in `info_pdi_page()` 204

missingglyphs keyword in `info_textline()` 94

miterlimit
 in `add_path_point()` 153
 in `create_gstate()` 140
 in several methods 137

modeltree suboption for `3Dactivate` in `create_annotation()` 259

monitor in `load_asset()` 247

move keyword in `add_path_point()` 152

multiline in `create_field/group()` 234

multiselect in `create_field/group()` 234

N

N in `shading()` and suboption of the shading graphics appearance option 167

name
 in `add_path_point()` 153
 in `create_3dview()` 255
 in `create_annotation()` 226
 in `info_font()` 72, 73
 in `load_asset()` 247
 in `load_asset()` and suboption for other methods 249
 keyword in `info_matchbox()` 134
 suboption for `matchbox` 133
 suboption for `targetpath` in `create_action()` 241
 suboption for viewports in `begin/end_page_ext()` 260

namelist in `create_action()` 239

names in `create_devicen()` 165

namespace in `begin_item()` and the tag option 265

nametree in `load_asset()` 247

network in `set_option()` 27

newwindow in `create_action()` 239

nextline in `add/create_textflow()` 100

nextparagraph in `add/create_textflow()` 100

nodenamelist in `begin_document()` 48

nofitlimit in `add/create_textflow()` 99

nonfullscreenpagemode suboption for viewerpreferences in `begin/end_document()` 53

normal suboption for template in `create_annotation()` 227

normalize
 in `set_text_option()`, `fit/info_textline()`, `fill_textblock()` and `add/create_textflow()` 77

numcolorglyphs in `info_font()` 74

numcopies suboption for viewerpreferences in `begin/end_document()` 53

numglyphs in `info_font()` 74

numpoints keyword in `info_path()` 156

numusableglyphs in `info_font()` 74

numusedglyphs in `info_font()` 74

O

objectheight, objectwidth keywords in `info_*`() 130

objectstreams in `begin_document()` 46

offset
 suboption for shadow in `add/create_textflow()` 78
 suboption for wrap in `fit_textflow()` 110

offsetbottom, offsetleft, offsetright, offsettop
 suboptions for `matchbox` 133

onpanel in `define_layer()` 62

opacity
 in `create_annotation()` 226
 in `load_asset()` 247
 suboption for `rendermode` in `create_3dview()` 257

opacityfill
 in `create_annotation()` 226
 in `create_gstate()` 141

opacitystroke
 in `create_gstate()` 141

opaque keyword for the backdropcolor suboption of `softmask` in `create_gstate()` 141

open
 in `create_annotation()` 226
 in `create_bookmark()` 220

openmode in `begin/end_document()` 46

openrect suboption for `matchbox` 133

operation in `create_action()` 240

optimize in `begin_document()` 46

optimizeinvisible in `load_font()` 68

orientate
 in `create_annotation()` 226
 in `create_field/group()` 234
 in `fit_textflow()` 107
 in various methods 126

outlineformat in `info_font()` 74

outputblockname suboption of `block` in `process_pdi()` 206

outputintents in `begin/end_page_ext()` 57

over suboption for the window suboption of `PDF_load_asset()` 249

overline in `set_text_option()`, `fit/info_textline()`, `fill_textblock()` and `add/create_textflow()` 78

P

page
 in `load_image()` 174
 option in `add_nameddest()` and suboption for destination in `create_action()`, `create_annotation()`, `create_bookmark()` and `begin/end_document()` 242

pageelement in `define_layer()` 62

pageheight, pagewidth
 keywords in `get_option()` 30
 keywords in `info_pdi_page()` 204

pagelayout in `begin/end_document()` 46

pagenumber
 in `begin_page_ext()` 57
 in `resume_page()` 60
 suboption for labels in `begin/end_document()` and label in `begin/end_page_ext()` 52
 suboption for `targetpath` in `create_action()` 241
 suboption of block in `process_pdi()` 206

pages suboption for `separationinfo` in `begin/end_page_ext()` 57

painttype in `begin_pattern_ext()` 169

parent
 in `begin_item()` and the tag option 265
 in `create_bookmark()` 220
 in `set_layer_dependency()` 63

parentlayer in `open_pdi_document` 194

parentname in `create_annotation()` 226

parenttitle in `open_pdi_document` 194

parindent in `add/create_textflow()` 98

passthrough in `load_image()` 174

password
 in `create_field/group()` 234
 in `load_asset()` and suboption for other methods 249
 in `open_pdi_document` 194
 suboption for network option lists 38

path
 in `add_path_point()` 153
 in `add_table_cell()` and suboption for caption 116
 suboption for `textpath` in `fit_textline()` 91
 suboption of paths suboption of wrap in `fit_textflow()` 110

pathlength
 keyword in `info_path()` 156
 keyword in `info_textline()` 94

pathref keyword in `add_path_point()` 152

paths suboption for wrap in `fit_textflow()` 110

pdfa in `begin_document()` 48

pdftagset in `begin_document()` 49

pdfua in `begin_document()` 48

pdfvt
 in `begin_document()` 48
 in `load_image()`, `load_graphics()`,
`open_pdi_page()` and `begin_template_ext()` 190

pdfx in `begin_document()` 48

pdi keyword in `get_option()` 30

pdiimage in `add_table_cell()` and suboption for caption 116

pdiusebox in `open_pdi_page()` 200

permissions
 in `begin_document()` 50
 in `load_asset()` 248

perpendiculardir keyword in `info_textline()` 94

picktraybypdfsize suboption for
 viewerpreferences in `begin/end_document()` 53

Placement in `begin_item()` and the tag option 268

polar in `add_path_point()` 154

polygons suboption for wrap in `fit_textflow()` 110

polylinelist in `create_annotation()` 226

portfolio in `end_document()` 47

position
 in `create_field/group()` 234
 in various methods 127
 suboption for template in
`create_annotation()` 227

prefix
 suboption for fieldlist in `add_portfolio_file/folder()` 280
 suboption for labels in `begin/end_document()` and label in `begin/end_page_ext()` 52

presentation suboption for the activate
 suboption of richmedia in
`create_annotation()` 252

preservepua in `load_font()` 68

preserveradio in `create_action()` 240

PrintFieldRole in `begin_item()` and the tag option 269

printpagerange suboption for viewerpreferences
 in `begin/end_document()` 53

printscaling suboption for viewerpreferences in
`begin/end_document()` 53

printsubtype in `define_layer()` 62

process in `create_device()` 165

properties in `begin_mc()` and `mc_point()` 272

proxy suboption for network option lists 38

px, py keywords in `info_path()` 156

R

ro, r1 in `shading()` and suboption of the shading
 graphics appearance option 167

radians in `add_path_point()` 154

radius in `add_path_point()` 154

readfeatures in `load_font()` 68

readkerning in `load_font()` 68

readonly
 in `create_annotation()` 226
 in `create_field/group()` 234

readselectors in `load_font()` 68

readshaping in `load_font()` 68

readverticalmetrics in `load_font()` 69

recordlevel in `begin_document()` 48

recordsize in `begin_document()` 51

rect keyword in `add_path_point()` 152

rectangle keyword in `info_matchbox()` 134

rectdiff in `create_annotation()` 226

rectify
 in `add_path_point()` 154
 in `elliptical_arc()` 149

recursive in `poca_delete()` 40

ref in `begin_item()` and the tag option 266

refpoint
 in `fit_graphics()`, `fill_*block()` and `info_path()` 127
 in `info_path()` 157
 suboption of paths suboption of `wrap` in `fit_textflow()` 110

relation suboption for `targetpath` in `create_action()` 241

relationship in `load_asset()` and suboption for other methods 250

relative in `add_path_point()` 154

remotestructdest in `create_action()` 240

remove suboption for logging in `set_option()` 18

removeonsuccess suboption for logging in `set_option()` 18

removeunused in `define_layer()` 62

rendercolor suboption for `rendermode` in `create_3dview()` 257

renderingintent
 in `create_gstate()` 141
 in `load_image()` 174

rendermode in `create_3dview()` 255

rendition in `create_action()` 240

repair in `open_pdi_document()` 195

repeatcontent in `add_table_cell()` 114

replacedchars in `info_textline()` 94

replacementchar
 in `info_font()` 74
 in `load_font()` 69

replyto in `create_annotation()` 227

required in `create_field/group()` 234

requiredmode in `open_pdi_document()` 195

resetfont in `add/create_textflow()` 100

resolution suboption for in `create_field/group()` 236

resourcefile in `set_option()` 27

resourcenumber in `get_option()` 30

restore in `add/create_textflow()` 100

resx, **resy** keywords in `info_image()` 178

return
 in `add_table_cell()` 115
 in `add/create_textflow()` 100

returnatmark in `fit_textflow()` 107

returnreason
 keyword in `info_table()` 121
 keyword in `info_textflow()` 111

revision keyword in `get_option()` 30

rewind
 in `fit_table()` 119
 in `fit_textflow()` 108

richmedia in `create_annotation()` 227

richmediaargs in `create_action()` 240

richtext in `create_field/group()` 234

right option in `add_nameddest()` and suboption for destination in `create_action()`, `create_annotation()`, `create_bookmark()` and `begin/end_document()` 243

rightindent
 in `add/create_textflow()` 98
 suboption for `createlastindent` in `fit_textflow()` 106

rightlinex, **rightliney** keywords in `info_textflow()` 111

righttoleft in `info_textline()` 94

rolemap in `begin_document()` 49

rollover suboption for `template` in `create_annotation()` 227

rotate
 in `begin/end_page_ext()` 57
 in `create_annotation()` 227
 in `fit_textflow()` 108
 in various methods 127
 keyword for `transform` in `begin_pattern_ext()`, `begin_template_ext()`, `shading_pattern()` and `open_pdi_page()` 192
 keyword in `info_pdi_page()` 204

round
 in `add_path_point()` 154
 in `draw_path()` 155
 in `fit_table()` 119
 suboption for `matchbox` 133

rowcount keyword in `info_table()` 121

rowheight in `add_table_cell()` 115

rowheightdefault in `fit_table()` 120

rowjoiningroup in `add_table_cell()` 115

rowscalegroup in `add_table_cell()` 115

rowspan in `add_table_cell()` 115

RowSpan in `begin_item()` and the tag option 269

rowsplit keyword in `info_table()` 121

ruler in `add/create_textflow()` 98

S

save in `add/create_textflow()` 100

saveresources in `set_option()` 27

scale
 in various methods 127
 keyword for `transform` in `begin_pattern_ext()`, `begin_template_ext()`, `shading_pattern()` and `open_pdi_page()` 192

schema suboption for `portfolio` in `end_document()` 281

scope
 keyword in `get_option()` 30
 suboption for `pdfvt` in `load_image()`, `load_graphics()`, `open_pdi_page()` and `begin_template_ext()` 192

Scope in `begin_item()` and the tag option 269

script
 in `create_action()` 240
 in `fit/info_textline()`, `fill_textblock()` and `add/create_textflow()` 93
 in `info_font()` 73
 in `load_3d()` 258

scriptlist keyword in `info_textline()` 94

scriptname in `create_action()` 240

scripts suboption for the activate suboption of
richmedia in *create_annotation()* 252

scrollable in *create_field/group()* 234

search in *begin/end_document()* 47

searchpath in *set_option()* 28

selector
 in *info_font()* 71
keyword in *info_font()* 74

selectorlist *keyword* in *info_font()* 74

separationinfo in *begin_page_ext()* 57

shading in several methods 137

shadow in *fit/info_textline()*, *fill_textblock()* and
add/create_textflow() 78

shaping in *fit/info_textline()*, *fill_textblock()* and
add/create_textflow() 93

Short in *begin_item()* and the tag option 269

showborder
 in *fit_textflow()* 108
 in various methods 127

showcaption in *create_annotation()* 228

showcells in *fit_table()* 120

showgrid in *fit_table()* 120

showtabs in *fit_textflow()* 108

shrinklimit
 in *add/create_textflow()* 99
 in *fit_textline()* 128

shrug in *open_pdi_document* 195

shutdownstrategy in *set_option()* 28

simplefont in *load_font()* 69

singfont in *info_font()* 74

skew *keyword* for *transform* in
begin_pattern_ext(), *begin_template_ext()*,
shading_pattern() and *open_pdi_page()* 192

skipembedding in *load_font()* 69

softmask in *create_gstate()* 141

sort suboption for *portfolio* in *end_document()*
 281

sorted in *create_field/group()* 234

source in *fetch_data()* 37

space in *add/create_textflow()* 100

spellcheck in *create_field/group()* 234

split
keyword in *info_textflow()* 111
 suboption for *portfolio* in *end_document()*
 281

spotcolor suboption for *separationinfo* in *begin/*
end_page_ext() 57

spotcolorlookup in *set_option()* 163

spotname suboption for *separationinfo* in *begin/*
end_page_ext() 57

sreadlimit in *add/create_textflow()* 99

sslcertdir suboption for *network* option lists 38

sslcertfile suboption for *network* option lists 38

sslverifyhost suboption for *network* option lists
 38

sslverifypeer suboption for *network* option lists
 38

stamp
 in *fit_textflow()* 108
 in various methods 128

standardfont in *info_font()* 74

start
 suboption for labels in *begin/*
end_document() and label in *begin/*
end_page_ext() 52
 suboption of the shading graphics
 appearance option 167

startcolor in *shading()* and suboption of the
 shading graphics appearance option 167

startoffset suboption for *textpath* in *fit_textline()*
 91

startx, starty *keywords* in *info_textline()* 94

state in *create_annotation()* 227

statemodel in *create_annotation()* 227

stopcolors in *shading()* and suboption of the
 shading graphics appearance option 167

stretch in *begin_font()* 86

strict suboption for *metadata* 277

strikeout in *set_text_option()*, *fit/info_textline()*,
fill_textblock() and *add/create_textflow()* 78

strikeselect in *load_font()* 69

stringformat in *set_option()* 28

strips *keyword* in *info_image()* 178

stroke
 in *add_path_point()* 154
 in *draw_path()* 155
 in *fit_table()* 120

strokeadjust in *create_gstate()* 141

strokecolor
 in *add_path_point()* 153
 in *create_field/group()* 234
 in *set_text_option()*, *fit/info_textline()*,
fill_textblock() and *add/create_textflow()* 79
 in several methods 137
 suboption for *shadow* in *add/*
create_textflow() 78

strokewidth
 in *set_text_option()*, *fit/info_textline()*,
fill_textblock() and *add/create_textflow()* 79
 suboption for *shadow* in *add/*
create_textflow() 78

structdest option in *add_nameddest()* and
 suboption for *destination* in *create_action()*,
create_annotation(), *create_bookmark()* and
begin/end_document() 243

structureassociatedfiles in *end_document()* 49

structuretype in *begin_document()* 49

style in *load_asset()* 248

style suboption for labels in *begin/*
end_document() and label in *begin/*
end_page_ext() 52

subject in *create_annotation()* 227

submitemptyfields in *create_action()* 240

submitname in *create_field/group()* 235

subpaths in *draw_path()* 156

subsetlimit in `load_font()` 69
subsetminsize in `load_font()` 69
subsetting in `load_font()` 69
Summary in `begin_item()` and the tag option 269
svgpath in `add_path_point()` 154
symbol in `create_annotation()` 227
symbolfont in `info_font()` 74
symbology suboption for barcode in `create_field/group()` 236

T

tabalignchar in `add/create_textflow()` 101
tabalignment in `add/create_textflow()` 98
tableheight, tablewidth keywords in `info_table()` 121
taborder
 in `begin/end_page_ext()` 57
 in `create_field/group()` 235
tag
 in `begin_document()` 49
 in `begin_item()` and the tag option 266
 in `fit_image(), fit_pdi_page(), fit_graphics(), fit_table(), fit_textline(), fit_textflow(), draw_path(), create_annotation(), fill_block(), create_field()` and suboption in `add_table_cell()` 271
tagbegin in `add/create_textflow()` 101
tagend in `add/create_textflow()` 101
tagged in `begin_document()` 50
tagname in `begin_item()` and the tag option 266
tagsets in `begin_document()` 50
tagtrailinghyphen in `set_text_option(), fit/info_textline()` and `fill_textblock()` 79
target in `create_action()` 240
targetpath
 in `create_action()` 240
 suboption for `targetpath` in `create_action()` 241
tempdirname in `begin_document()` 51
tempfilenames in `begin_document()` 51
template
 in `create_annotation()` 227
 suboption of `softmask` in `create_gstate()` 141
templateoptions
 in `load_graphics()` 182
 in `load_image()` 174
text
 in `add_table_cell()` and suboption for caption 116
 suboption for leader in `fit/info_textline()` and `add/create_textflow()` 92
textcolor in `create_bookmark()` 220
textendx, textendy keywords in `info_textflow()` 111
textflow
 in `add_table_cell()` and suboption for caption

116
 in `fill_textblock()` 214
 suboption for `createrichtext` in `create_annotation()` 224
textflowhandle in `fill_textblock()` 214
textformat
 in `set_option()` 28
 in `set_text_option(), fit/info_textline(), fill_textblock()` and `add/create_textflow()` 77
textheight
 keyword in `info_textflow()` 111
 keyword in `info_textline()` 94
textknockout in `create_gstate()` 141
textlen in `create_textflow()` 103
textpath in `fit/info_textline()` 91
textrendering
 in `set_text_option(), fit/info_textline(), fill_textblock()` and `add/create_textflow()` 79
 suboption for shadow in `add/create_textflow()` 78
textrise in `set_text_option(), fit/info_textline(), fill_textblock()` and `add/create_textflow()` 79
textstate in `get_option()` 30
textwidth
 keyword in `info_textflow()` 112
 keyword in `info_textline()` 94
textx, texty keywords in `get_option()` 30
thumbnail in `load_asset()` and suboption for other methods 250
tilingtype in `begin_pattern_ext()` 170
timeout suboption for network option lists 38
title
 in `begin_item()` and the tag option 266
 in `create_annotation()` 227
 keyword in `info_graphics()` 185
 suboption for the window suboption of `PDF_load_asset()` 249
toggle in `create_fieldgroup()` 235
tolerance suboption for `textpath` in `fit_textline()` 91
toolbar suboption for `3Dactivate` in `create_annotation()` 259
tooltip in `create_field/group()` 235
top option in `add_nameddest()` and suboption for destination in `create_action(), create_annotation(), create_bookmark()` and `begin/end_document()` 243
topdown
 in `begin_page_ext()` 58
 in `begin_pattern_ext()` and `begin_template_ext()` 187
topindex in `create_field/group()` 235
topeleveltag keyword in `info_pdi_page()` 204
topeleveltagcount keyword in `info_pdi_page()` 204
transform
 in `begin_template_ext(), load_graphics(), open_pdi_page(), and begin_pattern_ext()`

191
 in `create_devicen()` 165
 in `shading_pattern()` 168

transition
 in `begin/end_page_ext()` 58
 in `create_action()` 240

translate keyword for transform in
`begin_pattern_ext()`, `begin_template_ext()`,
`shading_pattern()` and `open_pdi_page()` 192

transparencygroup
 in `begin/end_page_ext()` 58
 in `load_image()`, `load_graphics()`,
`open_pdi_page()`, and `begin_template_ext()`
 191

transparent
 keyword for the `backdropcolor` suboption of
`softmask` in `create_gstate()` 141
 keyword in `info_image()` 178

trimbox in `begin/end_page_ext()` 58

truncatetrailingwhitespace in `fit_textflow()` 108

type
 in `create_3dview()` 255
 in `load_3d()` 258
 in `poca_insert()` 41
 in `shading()` and suboption of the shading
 graphics appearance option 167
 keyword in `info_graphics()` 185
 option in `add_nameddest()` and suboption for
 destination in `create_action()`,
`create_annotation()`, `create_bookmark()` and
`begin/end_document()` 243
 suboption for custom in `create_annotation()`
 225
 suboption for fieldlist in `add_portfolio_file/`
`folder()` 280
 suboption for properties in `begin_mc()` and
`mc_point()` 272
 suboption for rendermode in `create_3dview()`
 257
 suboption for the coords and `displaycoords`
 suboptions of `georeference` 261
 suboption of `softmask` in `create_gstate()` 141

U

U3Dpath in `create_3dview()` 255

underline in `set_text_option()`, `fit/info_textline()`,
`fill_textblock()` and `add/create_textflow()` 79

underlineposition in `set_text_option()`, `fit/`
`info_textline()`, `fill_textblock()` and `add/`
`create_textflow()` 79

underlinewidth in `set_text_option()`, `fit/`
`info_textline()`, `fill_textblock()` and `add/`
`create_textflow()` 79

unicode in `info_font()` 72, 74

unicodefont in `info_font()` 74

unicodemap in `load_font()` 69

unisonselect in `create_fieldgroup()` 235

unknownchars in `info_textline()` 94

unmappedchars in `info_textline()` 94

unmappedglyphs in `info_font()` 74

uri in `begin/end_document()` 47

url
 in `create_action()` 241
 in `load_asset()` 248
 suboption for network option lists 38

urls in `load_iccprofile()` 161

usage
 in `load_iccprofile()` 161
 in `poca_new()` 40

useactions
 in `fit_pdi_page()` 202
 in `open_pdi_document()` 195

useannots in `fit_pdi_page()` 202

useblackptcomp in `create_gstate()` 141

used keyword in `info_textflow()` 112

usedglyph in `info_font()` 74

useembeddedimage in `info_image()` 178

usefields in `fit_pdi_page()` 202

usehostfonts in `set_option()` 28

usehypertextencoding in `set_option()` 28

usejavascript in `open_pdi_document()` 195

uselayers in `open_pdi_document` 195

usematchbox
 in `create_annotation()` 228
 in `draw_path()` 156

usematchboxes suboption for wrap in
`fit_textflow()` 110

usercoordinates
 in `begin_item()` and the tag option 266
 in `create_annotation()` 228
 in `create_field/group()` 235
 in `set_option()` 28

userenditions in `open_pdi_document()` 195

userlog in `set_option()` 28

username suboption for network option lists 38

userpassword in `begin_document()` 50

userunit
 in `begin/end_page_ext()` 59
 suboption for `createrichtext` in
`create_annotation()` 224

usesttransparency in `begin_document()` 48

usetags
 in `open_pdi_document` 195
 in `open_pdi_page` 200

V

value
 in `poca_insert()` 42
 suboption for custom in `create_annotation()`
 225
 suboption for fieldlist in `add_portfolio_file/`
`folder()` 280
 suboption for properties in `begin_mc()` and
`mc_point()` 272

values in `poca_insert()` 42

vertboxgap keyword in `info_table()` 121

vertical
 in `info_font()` 74
 in `load_font()` 70

verticalalign in `fit_textflow()` 108

vertshrinking keyword in `info_table()` 121

vertshrinklimit in `fit_table()` 120

view suboption for the activate suboption of `richmedia` in `create_annotation()` 252

viewerpreferences in `begin_document()` and `end_document()` 47

viewports in `begin/end_page_ext()` 59

views
 in `load_3d()` 258
 suboption of `richmedia` in `create_annotation()` 251

volume in `load_asset()` 248

W

watermark in `begin_template_ext()` 187

weight
 in `begin_font()` 86
 in `info_font()` 75

wellformed keyword in `info_textline()` 95

width
 in `add_path_point()` 154
 in `begin_glyph_ext()` 87
 in `begin_template_ext()` 187
 in `begin/end_page_ext()` 59
 in `load_image()` 175
 keyword in `info_*`() 130
 keyword in `info_matchbox()` 134
 suboption for the window suboption of `PDF_load_asset()` 249

Width in `begin_item()` and the tag option 269

widthsonly in `begin_font()` 86

willembd in `info_font()` 75

willsubset in `info_font()` 75

window in `load_asset()` 248

wkt suboption for the `coords` and `displaycoords` suboptions of `georeference` 261

wordspacing in `set_text_option()`, `fit/info_textline()`, `fill_textblock()` and `add/create_textflow()` 79

worldpoints suboption for `georeference` 261

worldsystem suboption for `georeference` 261

wrap in `fit_textflow()` 109

writingdirx, writingdiry keywords in `info_textline()` 95

WritingMode in `begin_item()` and the tag option 269

X

x1, y1, ..., x4, y4
 keywords in `info_*`() 130
 keywords in `info_matchbox()` 134
 keywords in `info_textflow()` 112

xadvancedlist in `fit/info_textline()` 91

xheight
 in `info_font()` 75
 in `load_font()` 70
 keyword in `info_textline()` 95

xid
 keyword in `info_graphics()` 185
 keyword in `info_image()` 178
 keyword in `info_pdi_page()` 204
 suboption for `pdfvt` in `begin_template_ext()` 192

xrotate
 in `add_path_point()` 154
 in `elliptical_arc()` 149

xstep in `begin_pattern_ext()` 170

xsymheight, xsymwidth suboption for `barcode` in `create_field/group()` 236

xvertline keyword in `info_table()` 121

Y

ychorline keyword in `info_table()` 121

yposition suboption for `leader` in `fit/info_textline()` and `add/create_textflow()` 92

ystep in `begin_pattern_ext()` 170

Z

zoom
 in `add_nameddest()` and suboption for `destination` in `create_action()`, `create_annotation()`, `create_bookmark()` and `begin/end_document()` 243
 in `create_annotation()` 228
 in `define_layer()` 62

C Revision History

Date	Changes
January 25, 2022	▶ Updates for PDFlib 10.0.0
March 29, 2021	▶ Updates for PDFlib 9.3.1
July 14, 2020	▶ Updates for PDFlib 9.3.0
February 01, 2019	▶ Updates for PDFlib 9.2.0
February 01, 2018	▶ Updates for PDFlib 9.1.2
July 24, 2017	▶ Updates for PDFlib 9.1.1
December 15, 2016	▶ Updates for PDFlib 9.1.0
July 27, 2016	▶ Updates for PDFlib 9.0.7
November 23, 2015	▶ Updates for PDFlib 9.0.6
May 18, 2015	▶ Updates for PDFlib 9.0.5
December 16, 2014	▶ Updates for PDFlib 9.0.4
May 12, 2014	▶ Updates for PDFlib 9.0.3
December 17, 2013	▶ Updates for PDFlib 9.0.2
July 24, 2013	▶ Updates for PDFlib 9.0.1
March 11, 2013	▶ Updates for PDFlib 9.0.0
May 30, 2011	▶ Updates for PDFlib 8 VT Edition (internally 8.1.0)
May 30, 2011	▶ Various updates and corrections for PDFlib 8.0.3
December 09, 2010	▶ Various updates and corrections for PDFlib 8.0.2
September 22, 2010	▶ Various updates and corrections for PDFlib 8.0.1p7
April 13, 2010	▶ Various updates and corrections for PDFlib 8.0.1
December 04, 2009	▶ Updates for PDFlib 8.0.0
April 20, 2010	▶ Minor corrections for PDFlib 7.0.5
March 13, 2009	▶ Various updates and corrections for PDFlib 7.0.4
February 13, 2008	▶ Various updates and corrections for PDFlib 7.0.3
August 08, 2007	▶ Various updates and corrections for PDFlib 7.0.2
March 09, 2007	▶ Various updates and corrections for PDFlib 7.0.1
October 03, 2006	▶ Updates and restructuring for PDFlib 7.0.0; split the manual in Tutorial and API reference
February 15, 2007	▶ Various updates and corrections for PDFlib 6.0.4
February 21, 2006	▶ Various updates and corrections for PDFlib 6.0.3; added Ruby section
August 09, 2005	▶ Various updates and corrections for PDFlib 6.0.2
November 17, 2004	▶ Minor updates and corrections for PDFlib 6.0.1 ▶ introduced new format for language-specific function prototypes in chapter 8 ▶ added hypertext examples in chapter 3

Date	Changes
June 18, 2004	▶ Major changes for PDFlib 6
January 21, 2004	▶ Minor additions and corrections for PDFlib 5.0.3
September 15, 2003	▶ Minor additions and corrections for PDFlib 5.0.2; added block specification
May 26, 2003	▶ Minor updates and corrections for PDFlib 5.0.1
March 26, 2003	▶ Major changes and rewrite for PDFlib 5.0.0
June 14, 2002	▶ Minor changes for PDFlib 4.0.3 and extensions for the .NET binding
January 26, 2002	▶ Minor changes for PDFlib 4.0.2 and extensions for the IBM eServer edition
May 17, 2001	▶ Minor changes for PDFlib 4.0.1
April 1, 2001	▶ Documents PDI and other features of PDFlib 4.0.0
February 5, 2001	▶ Documents the template and CMYK features in PDFlib 3.5.0
December 22, 2000	▶ ColdFusion documentation and additions for PDFlib 3.0.3; separate COM edition of the manual
August 8, 2000	▶ Delphi documentation and minor additions for PDFlib 3.0.2
July 1, 2000	▶ Additions and clarifications for PDFlib 3.0.1
Feb. 20, 2000	▶ Changes for PDFlib 3.0
Aug. 2, 1999	▶ Minor changes and additions for PDFlib 2.0.1
June 29, 1999	▶ Separate sections for the individual language bindings ▶ Extensions for PDFlib 2.0
Feb. 1, 1999	▶ Minor changes for PDFlib 1.0 (not publicly released)
Aug. 10, 1998	▶ Extensions for PDFlib 0.7 (only for a single customer)
July 8, 1998	▶ First attempt at describing PDFlib scripting support in PDFlib 0.6
Feb. 25, 1998	▶ Slightly expanded the manual to cover PDFlib 0.5
Sept. 22, 1997	▶ First public release of PDFlib 0.4 and this manual

Index

Options are listed separately in Appendix B, page 287.

A

abbreviated tagging 271
action lists in option lists 12
All spot color name 163
Author field 276

B

Bézier curve 146
Boolean values in option lists 11

C

circles in option lists 16
CMYK color 13
cmk keyword 14
color functions 159
color in option lists 14
Creator field 276
curves in option lists 16

D

DeviceN color 13
devicen keyword 15
document and page functions 43
document information fields 275
document scope 17
Dublin Core 275

F

fast Web view 46
float and integer values in option lists 12
floats in option lists 11
font scope 17
fontsize in option lists 12

G

global options 25
glyph scope 17
graphics functions 135, 179
gray keyword 14

H

handles in option lists 12

I

ICC Profiles 161

ICC-based color 13
iccbased keyword 15
iccbasedcmyk keyword 15
iccbasedgray keyword 15
iccbasedrgb keyword 15
Ideographic Variation Sequences (IVS) 68
image functions 171
import functions for PDF (PDI) 193
info fields 275
inline option lists for Textflows 104
inner cell box for table cells 114
invisible text 79
IVS 68

K

Keywords field 276
keywords in option lists 11

L

Lab color 14
lab keyword 14
landscape mode 57
linearized PDF 46
lines in option lists 15
list values in option lists 8

M

metadata 277
method scopes 17

N

Nchannel color spaces 165
nested option lists 8
network option list 38
networking 37
None spot color name 163
numbers in option lists 11

O

object scope 17
option list syntax 8
outline text 79

P

page scope 17
page size formats 55
path painting and clipping 150

path scope 17
pattern color 14
pattern keyword 15
pattern scope 17
pCOS functions 193
pCOS methods 207
PDF import functions (PDI) 193
PDF Object Creation API (POCA) 39
PDF_activate_item() 270
PDF_add_nameddest() 242
PDF_add_path_point() 152
PDF_add_portfolio_folder() 278, 279
PDF_add_table_cell() 113
PDF_add_textflow() 96
PDF_align() 144
PDF_arc() 147, 148
PDF_arcn() 147
PDF_begin_document() 43
PDF_begin_dpart() 282
PDF_begin_font() 85
PDF_begin_glyph_ext() 86
PDF_begin_item() 263
PDF_begin_layer() 64
PDF_begin_mcl() 272
PDF_begin_page_ext() 55, 56
PDF_begin_pattern_ext 169
PDF_begin_template_ext() 186
PDF_circle() 147
PDF_clip() 151
PDF_close_font() 70
PDF_close_graphics() 183
PDF_close_image() 175
PDF_close_pdi_document() 196
PDF_close_pdi_page() 201
PDF_closepath_fill_stroke() 151
PDF_closepath_stroke() 150
PDF_closepath() 149
PDF_concat() 144
PDF_continue_text() 83
PDF_continue_text2() 83
PDF_convert_to_unicode() 21
PDF_create_3dview() 254
PDF_create_action() 237
PDF_create_annotation() 221
PDF_create_bookmark() 219
PDF_create_devicen() 164
PDF_create_field() 229
PDF_create_fieldgroup() 235
PDF_create_gstate() 140
PDF_create_pvf() 34
PDF_create_textflow() 102
PDF_curveto() 146
PDF_define_layer() 61
PDF_delete_dl() 33
PDF_delete_path() 157
PDF_delete_pvf() 35
PDF_delete_table() 122
PDF_delete_textflow() 112
PDF_delete() 33
PDF_download() 37
PDF_draw_path() 154
PDF_ellipse() 148
PDF_elliptical_arc() 148
PDF_encoding_set_char() 88
PDF_end_document() 44
PDF_end_dpart() 283
PDF_end_font() 86
PDF_end_glyph() 87
PDF_end_item() 269
PDF_end_layer() 64
PDF_end_mcl() 273
PDF_end_pattern() 170
PDF_end_template_ext() 188
PDF_endpath() 151
PDF_fill_graphicsblock() 217
PDF_fill_imageblock() 215
PDF_fill_pdfblock() 216
PDF_fill_stroke() 150
PDF_fill_textblock() 213
PDF_fill() 150
PDF_fit_graphics() 183
PDF_fit_image() 176
PDF_fit_pdi_page() 201
PDF_fit_table() 116
PDF_fit_textflow() 105
PDF_fit_textline() 89
PDF_get_apiname() 20
PDF_get_buffer() 54
PDF_get_errmsg() 20
PDF_get_errnum() 19
PDF_get_opaque() 20
PDF_get_option() 29
PDF_get_string() 31
PDF_info_font() 71
PDF_info_graphics() 184
PDF_info_image() 177
PDF_info_matchbox() 133
PDF_info_path() 156
PDF_info_pdi_page() 203
PDF_info_pvf() 35
PDF_info_table() 121
PDF_info_textflow() 109
PDF_info_textline() 92
PDF_lineto() 146
PDF_load_3ddata() 258
PDF_load_asset() 245
PDF_load_font() 65
PDF_load_graphics() 179
PDF_load_iccprofile() 161
PDF_load_image() 171
PDF_makespotcolor() 163
PDF_mc_point() 273
PDF_moveto() 146
PDF_new_dl() 32
PDF_new() 32
PDF_neww2() 32

PDF_open_pdi_callback() 196
PDF_open_pdi_document() 193
PDF_open_pdi_page() 198
PDF_pcos_get_number() 207
PDF_pcos_get_stream() 208
PDF_pcos_get_string() 207
PDF_poca_delete() 40
PDF_poca_insert() 41
PDF_poca_new() 39
PDF_poca_remove() 42
PDF_process_pdi() 205
PDF_rect() 149
PDF_restore() 140
PDF_resume_page() 59
PDF_rotate() 143
PDF_save() 139
PDF_scale() 143
PDF_set_graphics_option() 137
PDF_set_gstate() 142
PDF_set_info() 275
PDF_set_info2() 275
PDF_set_layer_dependency() 62
PDF_set_option() 25
PDF_set_text_option() 80
PDF_set_text_pos() 81
PDF_setcolor() 159
PDF_setfont() 81
PDF_setlinewidth() 139
PDF_setmatrix() 145
PDF_shading_pattern() 168
PDF_shading() 166
PDF_shfill() 168
PDF_show_xy() 82
PDF_show_xyz() 82
PDF_show() 82
PDF_show2() 82
PDF_skew() 144
PDF_stringwidth() 83
PDF_stringwidth2() 83
PDF_stroke() 150
PDF_suspend_page() 59
PDF_translate() 143
PDF/A or PDF/X output intent 206
PDFlib Personalization Server (PPS) 211
PDI (PDF import) 193
POCA (PDF Object Creation API) 39
polylines in option lists 15
PPS (PDFlib Personalization Server) 211
proxy configuration 38

R

raster image functions 171
rectangles in option lists 16
RGB color 13
rgb keyword 14
rich media 245

S

scopes 17
separation color space 13
setup functions 32
skewing 144
spot color (separation color space) 13
spot keyword 14
spotname keyword 15
standard page sizes 55
strings in option lists 10
Subject field 276
subscript 79
superscript 79
SVG 179
syntax of option lists 8

T

table formatting 113
template scope 17
text appearance options 89
text filter options 89
text functions 65
Textflow: inline option lists 104
Title field 276
Trapped field 276
Type 3 fonts 85

U

Unichar values in option lists 11
Unicode ranges in option lists 11
Unquoted string values in option lists 10

V

vector graphics functions 179

W

web-optimized PDF 46

X

XMP metadata 277

PDFlib GmbH

Franziska-Bilek-Weg 9
80339 München, Germany
www.pdflib.com
phone +49 • 89 • 452 33 84-0

Licensing contact

sales@pdflib.com

Support

support@pdflib.com (*please include your license number*)

