PDFlib Migration Guide

Latest PDFlib versions covered in this document: PDFlib 9.4 and 10.0

The PDFlib API generally remains compatible among major releases. Sometimes API methods, parameters or options are phased out and replaced with an improved successor which is more general or more powerful than the old method. In these situations the new PDFlib version declares the old method as deprecated without changing the functionality in any way. The new method is always preferable. New code should not use deprecated API methods; existing code should switch to the replacement method as soon as possible. Deprecated methods may be removed in a future release.

This Guide contains recommendations for users who migrate existing PDFlib application code which has been developed with an older PDFlib release. It explains how to identify deprecated API features in application code. Once identified the deprecated features should be replaced with the recommended newer ones.

PDFlib 10 removes several old features which have been deprecated in earlier versions. Legacy applications which still use these features must apply the replacements detailed in this document before they can switch to PDFlib 10. In most cases one-to-one replacement methods or options are available.

The majority of changes affects only very old applications which haven't been updated in many years.

Note *Since PDFlib 10 removes deprecated features it is strongly recommended to check your application for deprecated method calls and options with PDFlib 9.4 to ensure forward compatibility.*

1 Check for deprecated PDFlib API Features

Note *PDFlib 10 contains almost no deprecated features. You can use PDFlib 9.3 or above to check whether your application uses legacy features from earlier PDFlib versions.*

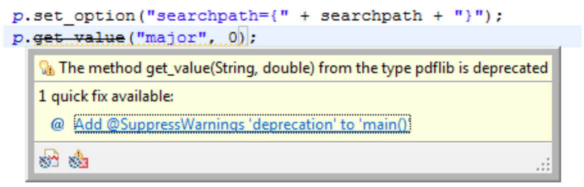
1.1 Identify deprecated API Method Calls at Compile Time

Deprecated API calls can be identified at compile time for some language bindings. Note that only warnings are emitted; the code can still be compiled successfully. Nevertheless it is strongly recommended to replace deprecated API methods with the recommended replacement method.

Java binding. The *pdflib.jar* module is created from source code which contains *@Deprecated* annotations and *@deprecated* Javadoc comments for all deprecated API methods so that the compiler can warn about the use of deprecated methods. The Java compiler emits a warning similar to the following:

```
javac -classpath pdflib.jar:.  image.java
Note: image.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
```

If you supply the compiler option *-Xlint:deprecation* the name and location of the deprecated API methods are shown:



```
p.set_option("searchpath={" + searchpath + "}");
p.~~get_value~~("major", 0);
```

The method get_value(String, double) from the type pdflib is deprecated
1 quick fix available:
@ Add @SuppressWarnings 'deprecation' to 'main()'

Fig. 1.1
Presentation of deprecated Java API methods in Eclipse

```
javac -classpath pdflib.jar:. -Xlint:deprecation  image.java
image.java:37: warning: [deprecation]
        get_value(String,double) in pdflib has been deprecated
        p.get_value( "major", 0.000000);
         ^
```

The Javadoc documentation for PDFlib lists all deprecated methods along with the recommended replacement method, e.g.

```
double get_value(java.lang.String key, double modifier) Deprecated.
Use PDF_get_option().
```

In Eclipse you must attach *pdflib.jar* as library to make information about deprecated API methods available to the integrated Java compiler. If deprecated API methods are found in your code, the name of the deprecated method is displayed with a strike-out, and a pop-up box informs about the method's status (see Figure 1.1). These messages are also listed in the *Problems* view.

.NET binding with Visual Studio. The PDFlib package marks all deprecated API methods so that calls to deprecated methods are marked with an underline, and IntelliSense pop-ups are shown with a warning for calls to deprecated API methods when editing the code.

The PDFlib assembly also contains »Obsolete« attributes for all deprecated API methods so that the compiler can warn about the use of deprecated API methods.

Visual Studio emits warnings for calls to deprecated API methods if warning level 2 is enabled in *Project, Properties, Build, Errors and warnings*. The warning includes the recommended replacement method:

```
image.cs(37,13): warning CS0618: 'PDFlib_dotnet.PDFlib.get_value(string, double)' is
obsolete: 'Deprecated, use PDF_get_option().'
```

.NET binding with command-line compiler. The *dotnet run* command also emits warnings for deprecated API methods in the compilation phase, provided the <*WarningLevel*> element in the project file is set to 2 or higher:

```
image.cs(37,13): warning CS0618: 'PDFlib.get_value(string, double)' is obsolete:
'Deprecated, use PDF_get_option().'
```

C++ binding with Visual Studio, GCC or Clang. The header file *pdflib.hpp* contains suitable attributes for all deprecated API methods so that the compiler can warn about the use of deprecated methods.

Visual Studio emits warnings for deprecated API calls if warning level 3 is enabled in *Project, Properties, Configuration Properties, C/C++, General/Warning Level: Level 3 (/W3)*:

```
image.cpp(33) : warning C4996: 'pdflib::basic_PDFlib<pstring,conv>::get_value':
was declared deprecated
        with
        [
            pstring=std::wstring,
            conv=pdflib::NoOpConverter<std::wstring>
        ]
```

With GCC and Clang the option *-Wdeprecated-declarations* must be supplied:

```
image.cpp:33:25: warning: 'double pdflib::basic_PDFlib<pstring, conv>::get_value(const
pstring&, double) [with pstring = std::__cxx11::basic_string<wchar_t>; conv =
pdflib::NoOpConverter<std::__cxx11::basic_string<wchar_t> >]' is deprecated
[-Wdeprecated-declarations]
  p.get_value(L"major", 0);
```

C binding with Visual Studio, GCC or Clang. The *pdflib.h* header file contains suitable attributes for all deprecated API methods so that the compiler can warn about the use of deprecated methods.

Visual Studio emits warnings for deprecated API calls if warning level 3 is enabled in *Project, Properties, Configuration Properties, C/C++, General/Warning Level: Level 3 (/W3):*

```
image.c(44): warning C4996: 'PDF_get_value': was declared deprecated
```

With GCC and Clang the option *-Wdeprecated-declarations* must be supplied:

```
image.c:44:2: warning: 'PDF_get_value' is deprecated [-Wdeprecated-declarations]
  PDF_get_value(p, "major", 0);
```

1.2 Identify deprecated API Method Calls at Runtime

Using the methods below you can create a list of deprecated API method calls at runtime. These API methods must be replaced in the application; see Section 2.1, »Deprecated PDFlib API Methods«, for details.

Logging for all language bindings. Calls to deprecated API methods are identified in the log file for all language bindings with logging class *api=1*. It can be set as follows (see PDFlib API Reference for more details on logging):

```
p.set_option("logging={filename=deprecated.log classes={api=1}}");
```

With this setting the log file contains a warning for calls to deprecated API methods, e.g.

```
PDF_get_value(p_0x2599c20, "major", 0.000000)
[PDF_get_value() is deprecated since PDFlib 9]
```

PHP language binding. The PDFlib extension module for PHP contains information about deprecated API methods so that the PHP interpreter can warn about deprecated methods. Depending on the PHP configuration warnings about deprecated method calls are written to the configured PHP error log file, the Web server log file, the Windows event log/system log, or *stderr/stdout*. The following directive in *php.ini* configures the log file:

```
error_log = /var/log/php.errors
```

The default is empty; see *www.php.net/manual/en/errorfunc.configuration.php* for details.
Whether or not warnings about deprecated calls are written to the log file depends on the *error_reporting* directive which must include *E_DEPRECATED,* e.g.

```
error_reporting = E_ALL
```

See *www.php.net/manual/en/function.error-reporting.php* for details. Once logging for deprecated calls is configured, PHP reports all deprecated calls as follows in the PHP error log file (not the PDFlib log file):

```
PHP Deprecated:  PDFlib::get_value(): Deprecated, use PDF_get_option(). in /home/bind/
php/image.php on line 37
```

Perl language binding. The PDFlib module for Perl emits warnings about deprecated API methods at runtime if the predefined warning category *deprecated* is enabled, which is true by default. Perl emits a warning similar to the following:

```
PDF_get_value(): Deprecated, use PDF_get_option(). at image.pl line 31.
```

Use the following instruction or the *-X* command-line option of the Perl interpreter to disable warnings about deprecated API methods on a module basis:

```
no warnings 'deprecated';
```

Warnings for deprecated API methods can be enabled on a module basis with the following instruction in the application code:

```
use warnings 'deprecated';
```

or by calling the Perl interpreter with the *-W* command-line option.

1.3 Identify deprecated Options

All language bindings. Deprecated options cannot be detected at compile time, but only at runtime. They can be identified in the logging output with the logging class *api=1* which can be set as follows:

```
p.set_option("logging={filename=deprecated.log classes={api=1}}");
```

With this setting the log file contains a message for calls with deprecated options, e.g.

```
PDF_create_annotation(p_0x1529c20, 100.000000, 400.000000, 400.000000, 500.000000,
"FileAttachment", "filename=foo.jpg mimetype=image/jpeg")
[Option "filename" is deprecated since PDFlib 9]
[Option "mimetype" is deprecated since PDFlib 9]
```

These options must be replaced in the application; see Section 2.2, »Deprecated Options and Keywords which are no longer available in PDFlib 10«, for details.

2 Deprecated PDFlib API Features

2.1 Deprecated PDFlib API Methods

Table 2.1 lists deprecated API methods which are no longer available in PDFlib 10. Applications must use the replacement methods shown in the table.

Table 2.1 Deprecated PDFlib API methods and their replacements

deprecated API method	deprecated since	replacement method

PDF_add_bookmark( )
PDF_add_bookmark2( )
	PDFlib 6	*PDF_create_bookmark( ); the* parent *and* open *parameters correspond to the same-named options*

PDF_add_launchlink( )	PDFlib 6	*PDF_create_action( ) with* type=Launch *and* *PDF_create_annotation( ) with* type=Launch
PDF_add_locallink( )	PDFlib 6	*PDF_create_action( ) with* type=GoTo *and* *PDF_create_annotation( ) with* type=Link

PDF_add_note( )
PDF_add_note2( )
	PDFlib 6	*PDF_create_annotation( ) with* type=Text

PDF_add_pdflink( )	PDFlib 6	*PDF_create_action( ) with* type=GoToR *and* *PDF_create_annotation( ) with* type=Link
PDF_add_thumbnail( )	PDFlib 9	*none; not functional since PDFlib 9*
PDF_add_weblink( )	PDFlib 6	*PDF_create_action( ) with* type=URI *and* *PDF_create_annotation( ) with* type=Link

PDF_attach_file( )
PDF_attach_file2( )
	PDFlib 6	*PDF_create_annotation( ) with* type=FileAttachment; *the* description *parameter corresponds to the* contents *option, the* author *parameter to the* title *option and the* icon *parameter to the* iconname *option.*

PDF_begin_glyph( )	PDFlib 9	*PDF_begin_glyph_ext( )*
PDF_begin_page( )	PDFlib 6	*PDF_begin_page_ext( )*
PDF_begin_pattern( )	PDFlib 9	*PDF_begin_pattern_ext( )*
PDF_begin_template( )	PDFlib 7	*PDF_begin_template_ext( )*
PDF_boot( ) (only C/C++)	PDFlib 7	*none; was never functional*
PDF_close( )	PDFlib 6	*PDF_end_document( )*
PDF_close_pdi( )	PDFlib 7	*PDF_close_pdi_document( )*
PDF_end_page( )	PDFlib 6	*PDF_end_page_ext( )*
PDF_end_template( )	PDFlib 8	*PDF_end_template_ext( )*
PDF_findfont( )	PDFlib 5	*PDF_load_font( )*

PDF_get_majorversion( )
PDF_get_minorversion( )
	PDFlib 4	*PDF_get_option( ) with keys* major/minor

PDF_get_parameter( )
PDF_get_value( )
	PDFlib 9	*For many keys the same-named option of PDF_get_option( ) can be used; see Table 2.3 for keys which require other replacement methods.*

PDF_get_pdi_parameter( )
PDF_get_pdi_value( )
	PDFlib 7	*PDF_pcos_get_string( ) and PDF_pcos_get_number( ) with pCOS paths according to Table 2.2*

PDF_initgraphics( )	PDFlib 9	*PDF_set_graphics_option( ) with option* initgraphicsstate
PDF_open_CCITT( )	PDFlib 5	*PDF_load_image( ) with CCITT-compressed TIFF images*
PDF_open_file( )	PDFlib 6	*PDF_begin_document( ) and PDF_end_document( )*

PDF_open_image( )
PDF_open_image_file( )
	PDFlib 5	*PDF_load_image( )*

PDF_open_mem( )	PDFlib 6	*PDF_begin_document( ) with empty* filename *and PDF_get_buffer( )*

Table 2.1 Deprecated PDFlib API methods and their replacements

deprecated API method	***deprecated since***	***replacement method***
PDF_open_pdi( )	*PDFlib 7*	*PDF_open_pdi_document( )*
PDF_place_image( )	*PDFlib 5*	*PDF_fit_image( ) with* `dpi=none`
PDF_place_pdi_page( )	*PDFlib 5*	*PDF_fit_pdi_page( )*
PDF_set_border_color( )	*PDFlib 6*	*PDF_create_annotation( ) with option* `annotcolor`
PDF_set_border_dash( )	*PDFlib 6*	*PDF_create_annotation( ) with option* `dasharray`
PDF_set_border_style( )	*PDFlib 6*	*PDF_create_annotation( ) with options* `borderstyle` *and* `linewidth`

PDF_set_parameter( )
PDF_set_value( )
	PDFlib 9	*For many keys the same-named option of PDF_set_option( ) can be used; see Table 2.3 for keys which require other replacement methods.*

PDF_setdash( ) *PDF_setdashpattern*	*PDFlib 9*	*PDF_set_graphics_option( ) and other methods with graphics appearance options with option* `dasharray` *and* `dashphase`
PDF_setflat( )	*PDFlib 9*	*PDF_set_graphics_option( ) or PDF_create_gstate( ) with option* `flatness`

PDF_setgray( )
PDF_setgray_fill( )
PDF_setgray_stroke( )
	PDFlib 4	*PDF_set_graphics_option( ) and other methods with color options*

PDF_setlinejoin( )	*PDFlib 9*	*PDF_set_graphics_option( ) or PDF_create_gstate( ) with option* `linejoin`
PDF_setlinecap( )	*PDFlib 9*	*PDF_set_graphics_option( ) or PDF_create_gstate( ) with option* `linecap`
PDF_setmiterlimit( )	*PDFlib 9*	*PDF_set_graphics_option( ) or PDF_create_gstate( ) with option* `miterlimit`
PDF_setpolydash( )	*PDFlib 5*	*PDF_set_graphics_option( ) and other methods with graphics appearance options with option* `dasharray`

PDF_setrgbcolor( )
PDF_setrgbcolor_fill( )
PDF_setrgbcolor_stroke( )
	PDFlib 4	*PDF_set_graphics_option( ) and other methods with color options*

PDF_show_boxed( )
PDF_show_boxed2( )
	PDFlib 6	*single lines: PDF_fit_textline( )*
		multi-line formatting: PDF_add/create_textflow( ) and PDF_fit_textflow( ) with options `minspacing=100% maxspacing=10000% nofitlimit=100% shrinklimit=100%` *to achieve similar formatting*
		If PDF_show_boxed( ) has been called with `mode=fulljustify` *use the following Textflow options:* `lastalignment=justify alignment=justify`

PDF_shutdown( ) *(only C/C++)*	*PDFlib 7*	*none; was never functional*

PDF_utf16_to_utf8( )
PDF_utf8_to_utf16( )
PDF_utf32_to_utf8( )
PDF_utf8_to_utf32( )
PDF_utf16_to_utf32( )
PDF_utf32_to_utf16( )
	PDFlib 8.1	*PDF_convert_to_unicode( ) with appropriate parameter* `inputformat` *and option* `outputformat`

PDF_xshow( ) (only C/C++)	*PDFlib 9*	*PDF_fit_textline( ) with option* `xadvancelist`

pCOS paths for deprecated PDF_get_pdi_value/parameter( ) methods. The deprecated (as of PDFlib 7) methods *PDF_get_pdi_value( )* and *PDF_get_pdi_parameter( )* use custom keys for addressing objects in a PDF documents. When switching to *PDF_pcos_get_number( )* and *PDF_pcos_get_string( )* these keys must be replaced with the corresponding pCOS paths according to Table 2.2 (see pCOS Path Reference for details).

Table 2.2 pCOS paths as replacements for deprecated keys of PDF_get_pdi_value( ) and PDF_get_pdi_parameter( )

key for PDF_get_pdi_value( ) and PDF_get_pdi_parameter( )	***pCOS path for PDF_pcos_get_number( ) and PDF_pcos_get_string( )***
`vdp/blockcount`	`length:pages[...]/blocks`
`/Root/Pages/Count`	`length:pages`
`width`, `height`	`pages[...]/width`, `pages[...]/height`
`/Rotate`	`pages[...]/Rotate`
`version`	`pdfversion`
`/CropBox`, `/BleedBox`, `/ArtBox`, `/TrimBox`, `/MediaBox`	`pages[...]/CropBox[0]` *for* `llx`, `pages[...]/CropBox[1]` *for* `lly`, `pages[...]/CropBox[2]` *for* `urx`, `pages[...]/CropBox[3]` *for* `ury` *etc.*
`vdp/Blocks/<name>/<property>`	`pages[...]/blocks/<name>/<property>`
`vdp/Blocks[...]/<property>`	`pages[...]/blocks[...]/<property>`
`vdp/Blocks/<name>/Custom/<property>`	`pages[...]/blocks/<name>/Custom/<property>` *or* `pages[...]/blocks[...]/Custom/<property>`
`isempty`	`pages[...]/isempty`
`filename`	`filename`
`/Info/<key>`	`/Info/<key>`
`tagged`	`tagged`
`pdfx`	`pdfx`

Replacements for deprecated global values and parameters. In most cases the deprecated methods *PDF_set/get_value( )* and *PDF_set/get_parameter( )* can be replaced with *PDF_set/get_option( )* and the previous parameter *key* as option name. Table 2.3 lists parameters which must be replaced with other methods and options.

Table 2.3 Deprecated global parameters for *PDF_set/get_value( ) and PDF_set/get_parameter( ) and their replacements*

deprecated parameter	*deprecated since*	*replacement method and option*
resource categories		
`Encoding`, `FontAFM`, `FontOutline`, `FontPFM`, `HostFont`, `ICCProfile`, `resourcefile`, `searchpath`	*PDFlib 9*	*same-named options of PDF_set_option( ); multiple entries can be set with a single call;* *PDF_get_option( ): use option* `resourcenumber` *to iterate over all values*
`StandardOutputIntent`	*PDFlib 9*	*none; use embedded or referenced ICC profile as output intent*
document properties		
`compatibility`	*PDFlib 6*	*PDF_begin_document( ) with option* `compatibility`
`userpassword`, `masterpassword` `permissions`	*PDFlib 6*	*PDF_begin_document( ) with options* `userpassword`, `masterpassword`, `permissions`
`pdfx`	*PDFlib 6*	*PDF_begin_document( ) with option* `pdfx`
`base`	*PDFlib 6*	*PDF_begin_document( ) with option* `uri`
`openmode`	*PDFlib 6*	*PDF_begin_document( ) with option* `openmode`

`openaction`	*PDFlib 6*	*PDF_begin/end_document( ) with option* `destination`*; the following deprecated option keywords must be replaced:*
		`fitbbox:` *use* `type=fitvisible`
		`fitheight:` *use* `type=fitheight left=0`
		`fitpage:` *use* `type=fitwindow`
		`fitwidth:` *use* `type=fitwidth top=10000`
		`retain:` *use* `type=fixed`

`hidetoolbar`, `hidemenubar`, `hidewindowui`, `fitwindow`, `centerwindow`, `displaydoctitle`, `nonfullscreenpagemode`, `direction`, `viewarea`, `viewclip`, `printarea`, `printclip`	*PDFlib 6*	*PDF_begin/end_document( ) with option* `viewer-preferences` *and respective suboptions;* *the suboptions* `viewarea`, `viewclip`, `printarea`, `printclip` *are deprecated in PDF 2.0 (see Table 2.5*)
page properties		
`pageheight`, `pagewidth` *in PDF_set_ parameter( )*	*PDFlib 6*	*PDF_begin/end_page_ext( ) with options* `pagewidth` *and* `pageheight`
`ArtBox`, `BleedBox`, `CropBox`, `TrimBox`	*PDFlib 6*	*PDF_begin/end_page_ext( ) with options* `artbox`, `bleed-box`, `cropbox`, `trimbox`
`duration`, `transition`	*PDFlib 6*	*PDF_begin/end_page_ext( ) with same-named options*
color		
`defaultgray`, `defaultrgb`, `defaultcmyk`	*PDFlib 6*	*PDF_begin/end_page_ext) with options* `defaultgray`, `defaultrgb`, `defaultcmyk`
`setcolor:iccprofilegray/rgb/cmyk`	*PDFlib 9*	*global options* `iccprofilegray/rgb/cmyk`
`preserveoldpantonenames`	*PDFlib 9.3*	*none; old-style PANTONE color names are no longer in use*

Table 2.3 Deprecated global parameters for PDF_set/get_value( ) and PDF_set/get_parameter( ) and their replacements

deprecated parameter	***deprecated since***	***replacement method and option***
font handling		
autosubsetting, subsetlimit, subsetminsize, fontstyle, unicodemap	*PDFlib 7*	*PDF_load_font( ) with same-named options*
ascender, ascenderfaked, capheight, capheightfaked, descender, descenderfaked, fontencoding, fontname, fontmaxcode, xheight, xheightfaked	*PDFlib 7*	*PDF_info_font( ) with same-named keywords; for the numerical values* fontsize=1 *must be supplied;* *parameter* fontencoding: *use keyword* encoding
autocidfont	*PDFlib 8*	*none; not functional since PDFlib 9*
monospace	*PDFlib 7*	*none; only for the obsolete concept of standard CJK fonts*
text output		
charref, charspacing, decorationabove, escapesequence, fakebold, font, fontsize, glyphcheck, horizscaling, italicangle, kerning, leading, overline, strikeout, strokewidth, textformat, textrendering, textrise, underline, underlineposition, underlinewidth, wordspacing	*PDFlib 9*	*PDF_set_text_option( ), PDF_fit/info_textline( ), PDF_fill_textblock( ) and PDF_add/create_textflow( )with same-named options*
images		
honoriccprofile	*PDFlib 8.2*	*PDF_load_image( ) with option* honoriccprofile
renderingintent	*PDFlib 8.2*	*PDF_load_image( ) with option* renderingintent
imagewidth, imageheight, image:iccprofile, orientation, resx, resy	*PDFlib 8*	*PDF_info_image( ) with same-named options*
graphics state		
fillrule *in PDF_set_parameter( )*	*PDFlib 9.3*	*PDF_set_graphics_option( ) with option* fillrule
inheritgstate	*PDFlib 6*	*none; not functional*
interactive elements		
bookmarkdest	*PDFlib 6*	*PDF_create_bookmark( ) with options* action, destination, fontstyle *and* textcolor
launchlink:parameters launchlink:operation launchlink:defaultdir	*PDFlib 6*	*PDF_create_action( ) with options* parameters, operation, *and* defaultdir; *however, these are deprecated in PDF 2.0 (see Table 2.5)*
miscellaneous		
flush	*PDFlib 6*	*PDF_begin_document_callback( ) with option* flush
pdiusebox	*PDFlib 6*	*PDF_open_pdi_page( ) with option* pdiusebox
iccwarning, fontwarning, glyphwarning, imagewarning, pdiwarning, warning	*PDFlib 7*	*corresponding API method with option* errorpolicy
openwarning	*PDFlib 6*	*PDF_begin_document( ) with option* errorpolicy

Table 2.3 Deprecated global parameters for *PDF_set/get_value( ) and PDF_set/get_parameter( ) and their replacements*

deprecated parameter	***deprecated since***	***replacement method and option***
`honorlang`	*PDFlib 8*	*PDF_set_option( ) with option* `filenamehandling=honor-lang`
`trace`, `tracefile`, `tracemsg`	*PDFlib 7*	*PDF_set_option( ) with option* `logging`
`logmsg`	*PDFlib 9*	*PDF_set_option( ) with option* `userlog`
`nodemostamp`	*PDFlib 9*	*PDF_set_option( ) with option* `avoiddemostamp`
`string`	*PDFlib 9*	*PDF_get_string( )*
`version`	*PDFlib 9*	*PDF_get_string( ) with option* `version` *and* `idx=-1`

2.2 Deprecated Options and Keywords which are no longer available in PDFlib 10

Table 2.4 lists API methods for which some options or keywords for an option are deprecated in PDFlib 9 and no longer available in PDFlib 10.

Table 2.4 Deprecated PDFlib API options or keywords which have been removed in PDFlib 10, and their replacements

deprecated option or keyword	***deprecated since***	***replacement option or action***
font and text		
PDF_create_textflow( ): textwarning	*PDFlib 7*	errorpolicy
PDF_fit_textline( ): locallink, pdflink, weblink	*PDFlib 7*	matchbox *in PDF_fit_textline( ) and PDF_create_annotation( ) to create links*
PDF_load_font( ): autocidfont	*PDFlib 9*	*none; not functional*
PDF_load_font( ): autosubsetting, subsetlimit	*PDFlib 10.0.1*	*These options don't work with the font subsetting implementation in PDFlib 10. They are still accepted for compatibility, but don't have any effect on subsetting.*
PDF_load_font( ), PDF_info_font( ) and PDF_fill_textblock( ): option and Block property monospace	*PDFlib 9*	*configure TrueType/OpenType fonts or use host fonts*
PDF_load_font( ): fontwarning	*PDFlib 7*	errorpolicy
PDF_load_font( ): kerning	*PDFlib 8*	readkerning
text option glyphwarning	*PDFlib 7*	glyphcheck
PDF_info_textline( ): keywords scalex, scaley	*PDFlib 9*	fitscalex/fitscaley
PDF_info_textline( ): keyword unmappedglyphs	*PDFlib 8*	unmappedchars
PDF_info_textflow( ): keyword remainchars	*PDFlib 8*	*none; worked reliably only under certain conditions*
PDF_fit/info_textline( ), PDF_add/create_textflow( ) and PDF_fill_textblock( ): option features *with keywords* vrt2 *and* vert	*PDFlib 9.3*	*none; these features are enabled automatically for fonts in vertical mode*
PDF_begin_font( ): colorize	*PDFlib 10*	*option* colorize *of PDF_begin_glyph_ext( )*
interactive elements		
PDF_create_action( ): actionwarning	*PDFlib 7*	errorpolicy
PDF_create_action( ): defaultdir, parameters *and* operation *for* type=Launch[1]	*PDFlib 9.3*	*none*
PDF_create_annotation( ): annotwarning	*PDFlib 7*	errorpolicy
PDF_create_annotation( ): filename *and* mimetype	*PDFlib 9*	attachment *and options* filename *and* mimetype *of PDF_load_asset( )*
PDF_create_annotation( ): popup	*PDFlib 10*	*none; the connection is created automatically if* parentname *is specified for the popup annotation.*
PDF_create_annotation( ): option template *with suboptions* normal/rollover/down: *keyword* viewer	*PDFlib 10*	*none; PDFlib always creates the required appearance stream*
PDF_create_field( ), PDF_create_fieldgroup( ): fieldwarning	*PDFlib 7*	*none; didn't have any effect*
PDF_create_field( ), PDF_create_fieldgroup( ): errorpolicy	*PDFlib 9.3*	*none; didn't have any effect*
PDF_create_bookmark( ): option destination with *suboptions* fontstyle *and* color	*PDFlib 6*	*options* fontstyle *and* textcolor *of PDF_create_bookmark( )*

Table 2.4 Deprecated PDFlib API options or keywords which have been removed in PDFlib 10, and their replacements

deprecated option or keyword	***deprecated since***	***replacement option or action***
Destination options for PDF_add_nameddest( ) and for the destination *option in PDF_create_action( ), PDF_create_annotation( ), PDF_create_bookmark(), and PDF_begin/end_document( ):* *options* fitbbox, fitheight, fitpage, fitwidth, retain	*PDFlib 7*	type=visible, fitheight, fitwindow, fitwidth, fixed
option filename *and* type=file		action *option with a* GoToR *action*
option name *and* type=nameddest		action *option with a* GoTo *action*
PDF_create_annotation( ): type=Movie[1]	*PDFlib 9.3*	type=RichMedia
PDF_create_action( ): type=Movie[1]	*PDFlib 9.3*	type=RichMediaExecute
image and PDF import		
PDF_load_image( ), PDF_fill_imageblock( ): template	*PDFlib 8*	templateoptions
PDF_load_image( ), PDF_fill_imageblock( ): createtemplate, iconname	*PDFlib 9*	templateoptions
PDF_load_image( ), PDF_fill_imageblock( ): downsamplemask	*PDFlib 9.2*	*none; rendering bug fixed in Acrobat X*
PDF_load_image( ), PDF_fill_imageblock( ): imagewarning	*PDFlib 7*	errorpolicy
PDF_open_pdi_document( ), PDF_open_pdi_page( ), PDF_process_pdi( ), PDF_fill_pdfblock( ): pdiwarning	*PDFlib 7*	errorpolicy
ICC profiles		
PDF_load_iccprofile( ): metadata	*PDFlib 9.3*	*none; XMP for profiles no longer supported*
PDF_load_iccprofile( ): iccwarning	*PDFlib 7*	errorpolicy
miscellaneous		
Common XObject options and PDF_begin_page_ext( ): transparencygroup *with suboptions* CS, I, K	*PDFlib 8.1*	*suboptions* colorspace, isolated, knockout
PDF_add_table_cell( ): checkwordsplitting	*PDFlib 8.1*	avoidwordsplitting
PDF_begin_document( ): moddate	*PDFlib 9.2*	*none; Acrobat Preflight bug fixed*
PDF_begin_item( ) and option tag *in many methods:* inline	*PDFlib 9.2*	*tagging option* direct
PDF_set_option( ) and many other API methods: errorpolicy=legacy	*PDFlib 8*	errorpolicy=return *or* errorpolicy=exception
PDF_set_option( ): topdown	*PDFlib 9*	*PDF_begin_page_ext( ), PDF_begin_template_ext( ) or PDF_begin_pattern_ext( ) with option* topdown
PDF_get_option( ): topdown	*PDFlib 9.3*	*none; applications must take care of options specified earlier*
PDF_set_option( ): filenamehandling=legacy	*PDFlib 9.3*	filenamehandling *with explicit encoding name or* honorlang
PDF_set_option( ): licence *and* licencefile	*(was never documented)*	*use corresponding options* license *and* licensefile

1. This feature is deprecated in PDF 2.0 according to ISO 32000-2.

2.3 Deprecated Features in PDF 2.0 and non-ISO Features

Deprecated PDF 2.0 features which are still available in PDF 1.x. PDF 2.0 according to ISO 32000-2 deprecates the features listed in Table 2.5. If any of these features is used, PDFlib 9.4 emits a warning (PDFlib 10: warning in PDF 1.x mode and an error in PDF 2.0 mode). It is recommended to avoid these features even in PDF 1.x mode.

Table 2.5 Features deprecated in PDF 2.0 which are still available in PDF 1.x mode

PDFlib features deprecated in PDF 2.0	***notes***
PDF_begin_document( ): `permissions` *with keyword* `noaccessible`	*Restricting document accessibility is deprecated in PDF 2.0.*
PDF_begin/end_document( ): `viewerpreferences`, *suboptions* `printarea`, `printclip`, `viewarea`, `viewclip`	*Prepress viewer preferences are deprecated in PDF 2.0.*
PDF_begin_page_ext( ) and PDF_end_page_ext( ): `separationinfo`	*Separation dictionaries are deprecated in PDF 2.0.*
PDF_create_annotation( ): `type=Movie`	*Use* `type=RichMedia`.
PDF_create_action( ): `type=Movie`	*Use* `type=RichMediaExecute`.
PDF_create_gstate( ): `blendmode` *accepts only a single keyword, but not a list with multiple values*	*Blendmode arrays are deprecated in PDF 2.0.*

Features outside of ISO 32000-1. The features listed in Table 2.6 were never part of ISO 32000-1. They work only in Acrobat, but not in third-party viewers. If any of these features is used, PDFlib 9.4 emits a warning (PDFlib 10: warning in PDF 1.x mode and an error in PDF 2.0 mode). It is recommended to avoid these features since they are not part of standard PDF.

Table 2.6 Features outside of ISO 32000 which are still available in PDF 1.x mode

features outside of ISO 32000	***notes***
PDF_create_field( ) and PDF_create_fieldgroup( ): `barcode`	*Barcode fields work only in the full version of Acrobat, but neither in Acrobat Reader nor third-party viewers.*
PDF_begin_document( ): `search`	*An attached search index works only in Acrobat, but not in third-party viewers.*
PDF_begin_template_ext( ): `watermark`	*Editable watermarks work only in Acrobat, but not in third-party viewers.*

2.4 Obsolete Concepts which have been removed from PDFlib 10

Some concepts are obsolete because there is no longer demand in today's workflows or they haven't been accepted in the marketplace. The items listed in have been removed in PDFlib 10.

Table 2.7 Obsolete PDF concepts which have been removed from PDFlib 10

deprecated concept	***deprecated since***	***notes***
PDF versions and standards		
PDF 1.3 (1999): PDF_begin_document( ) with option `compatibility=1.3`	*PDFlib 8.1*	*not relevant*
PDF/X-1:2001: PDF_begin_document( ) with option `pdfx=PDF/X-1:2001`	*PDFlib 7*	*superseded by PDF/X-4*
PDF/X-1a:2001, PDF/X-3:2002: PDF_begin_document( ) with option `pdfx=PDF/X-1a:2001` *and* `PDF/X-3:2002`	*PDFlib 8.1*	*superseded by PDF/X-4*
PDF/X-3 standard output intents without embedded ICC profile and resource category `StandardOutputIntent`	*PDFlib 9*	*superseded by PDF/X-3 and PDF/X-4 with embedded or referenced ICC profile*
PDF/X-5g, PDF/X-5pg: PDF_begin_document( ) with option `pdfx=PDF/X-5g` *and* `PDF/X-5pg`	*PDFlib 9.3*	*not accepted in the marketplace*
PDF/VT-2: PDF_begin_document( ) with option `pdfvt=PDF/VT-2`	*PDFlib 9.3*	*not accepted in the marketplace*
password security		
password encryption: PDF_begin_document( ) with option `userpassword` *or* `masterpassword` *and* `compatibility=1.4` *or* `1.5`	*PDFlib 9.3.1*	*This combination results in weak RC4 encryption which should be avoided. Create a newer PDF version, preferably PDF 1.7ext8.*
font handling		
Multiple Master PostScript Type 1 fonts	*PDFlib 9.3*	*not relevant*
resource fork PostScript Type 1 (LWFN) fonts on macOS	*PDFlib 9.3*	*not relevant*
SVG fonts, i.e. fonts in SVG files (not OpenType color fonts with SVG glyphs)	*PDFlib 9.3*	*deprecated in SVG 2.0 and unsupported in most browsers*
CEF fonts	*PDFlib 9.3*	*not used as stand-alone fonts*
standard CJK fonts and font option/Block property `monospace`	*PDFlib 9*	*Configure TrueType/OpenType fonts or use host fonts.*
CJK TrueType fonts without Unicode cmap table	*PDFlib 9.3*	*Use Unicode-compatible fonts.*
image handling		
images in GIF and CCITT formats	*PDFlib 9.4*	*outdated formats; use PNG or TIFF instead*
UCS-2 CMaps	*PDFlib 9.3*	*Use* `encoding=unicode`.
PANTONE spot colors		
PANTONE Goe coated/uncoated *PANTONE hexachrome coated/uncoated* *PANTONE solid in hexachrome coated* *PANTONE solid to process coated* *PANTONE solid to process coated EURO*	*PDFlib 9.1*	*no longer supported by Pantone, Inc.*

Table 2.7 Obsolete PDF concepts which have been removed from PDFlib 10

deprecated concept	deprecated since	notes
PANTONE color bridge CMYK PC PANTONE color bridge CMYK EURO PANTONE color bridge uncoated PANTONE solid matte PANTONE process coated EURO PANTONE process uncoated EURO PANTONE process coated PANTONE process uncoated	2010	no longer supported by Pantone, Inc. since 2010
mapping of deprecated PANTONE spot color name suffixes CV, CVV, CVU, CVC and CVP	PDFlib 9.3	old-style PANTONE spot color names are no longer in use
Interactive features		
Movie annotations[1]: PDF_create_annotation( ) with `type=Movie` and PDF_create_action( ) with `type=Movie`	PDFlib 9.3	Use `Screen` annotations and `Rendition` actions.
Flash[1]		
PDF_load_asset( ) with `type=Flash/Generic/JPEG/PNG` PDF_create_annotation( ) with `type=RichMedia` and option `configuration`, suboption `instances`, suboption `params` PDF_end_document( ) with option `portfolio` and suboptions `navigator` and `initialview=custom`	PDFlib 9.3	Flash in `RichMedia` annotations and Flash-based navigators are no longer supported in Acrobat since December 2020. Use `Screen` annotations or JavaScript.
other PDF features		
Reference XObjects: common XObject option `reference`	PDFlib 9.3	not accepted in the marketplace
page thumbnails: PDF_add_thumbnail( )	PDFlib 9	not functional since PDFlib 9.1; PDF viewers create page thumbnails automatically
Open Prepress Interface (OPI)[1]: common XObject options `OPI-1.3`, `OPI-2.0`	PDFlib 9.1	OPI is a concept of the 1980s which has very rarely been used in PDF.
PostScript XObjects[1]: PDF_begin_template_ext( ) with option `postscript`	PDFlib 9.3	not relevant

1. This feature is deprecated or not included in PDF 2.0 according to ISO 32000-2.

2.5 Deprecated Features which are still available in PDFlib 10

Table 2.8 lists deprecated features which are no longer documented, but are still available in PDFlib 10 for the benefit of legacy applications. Users should prepare for future removal of these features and adjust their applications accordingly. New applications should not use any of these features.

Table 2.8 Deprecated features which are still available in PDFlib 10, but shouldn't be used

deprecated feature	***deprecated since***	***recommended replacement option or action***
Encryption with PDF_begin_document( ) and the options `userpassword/masterpassword` *with compatibility*=`1.6`, `1.7` *or* `1.7ext3`	*PDFlib 10*	*Remove the password option or increase PDF output compatibility to 1.7ext8 or above.*
CMaps for CJK legacy encodings, i.e. loading a font with a predefined CMap (this affects only non-Unicode-aware language bindings); font option `keepnative`	*PDFlib 10*	*Switch to a Unicode workflow.*
PostScript Type 1 fonts	*PDFlib 9.3.1*	*Use TrueType or OpenType fonts.*
PDF/X-1a:2003: PDF_begin_document( ) with option `pdfx=PDF/X-1a:2003`	*PDFlib 9.3*	*Use PDF/X-4.*
PDF/X-3:2003: PDF_begin_document( ) with option `pdfx=PDF/X-3:2003`	*PDFlib 9.4*	*Use PDF/X-4.*
PDF_begin_font( ): option `colorized`	*PDFlib 10*	*Use option* `colorized` *of PDF_begin_glyph( ); the color status can now be specified for individual glyphs instead of only for all glyphs in the font.*
PDF_info_font( ): option `shapingsupport`	*PDFlib 9.3.1*	*This keyword couldn't be used reliably to test for shaping support of a font.*

3 Configuration Changes in PDFlib 10

3.1 Modified Default Values

The default values of some options have been modified according to Table 3.3. This allows shorter code, but in a few cases the options must be supplied with the desired value if the new defaults don't match your requirements.

Table 3.1 PDFlib features which require configuration changes

PDFlib feature	***required change if the new default doesn't match your requirements***
The default PDF compatibility has been increased from PDF 1.7 to PDF 1.7ext8 which supports stronger encryption, `RichMedia` *annotations and other features.*	*If you must create PDF 1.7 output (the default in PDFlib 9) use PDF_begin_document( ) with option* `compatibility=1.7.`
PDF_load_font( ) and implicit font loading: fonts are embedded by default	*Fonts are embedded by default. Use the option* `embedding=false` *to disable font embedding if you really want that or must use deprecated PostScript Type 1 fonts for which no embeddable outlines are available.* *The default of* `skipembedding` *is now* `{latincore}` *(except in PDF/A, PDF/X and PDF/UA mode where the default is still empty). This means that Standard Latin fonts without embeddable outlinesare silently processed without embedding instead of raising an exception.*
PDF_load_image( ) with RGB images: the sRGB ICC profile is now applied by default	*sRGB is useful for achieving more accurate color automatically. In rare cases where sRGB is not desired it can be disabled with* `iccprofile=none`*. This is required for the* `thumbnail` *option of PDF_load_asset( ).*
PDF_load_graphics( ) and option `templateoptions:` *templates are created by default*	*PDFlib now creates a Form XObject (template) for SVG by default. The previous behavior can be achieved with the* `inline` *option.*
Option `errorpolicy` *of PDF_set_option( ) and many other API methods: the new default is* `return`*.*	*Use* `errorpolicy=return` *and check error return values or set* `errorpolicy=exception` *if it is acceptable to give up the output document in case of a failed call.*

3.2 Modified Text Output

In some situations text output may look different compared to earlier versions. If you prefer the previous behavior you must adjust your code according to Table 3.2.

Table 3.2 PDFlib features which result in modified text output

PDFlib feature	***required change***
Text output with OpenType fonts and the `features` *option of PDF_fit/info_textline( ), PDF_add/create_textflow( ), and PDF_fill_textblock( ): the following OpenType features are enabled by default as required by the OpenType specification:* `calt`, `ccmp`, `clig`, `liga`, `locl`.	*Use one of the following methods to disable OpenType features:* ► *disable a particular feature, e.g.* `liga`, *with the typographic option* `features={noliga}` ► *disable all OpenType features for a text fragment with* `features={_none}` ► *completely ignore OpenType feature tables with the font option* `readfeatures=false`

3.3 Other Configuration Changes

In some situations you must adjust source and configuration according to Table 3.3.

Table 3.3 PDFlib features which require configuration changes

PDFlib feature	***required change***
PDF_begin_document( ) where the `pdfua` *option has a value different from* `none`	*The option* `lang` *is required in PDF/UA mode since PDFlib 9.2.0.*
PDF_create_annotation( ) with `type=Caret`, `FreeText`, `Line` *(only if* `showcaption=true`)	*Access to the font* `NotoSans-Regular` *must be configured unless the* `font` *option is supplied.*[1]
PDF_create_annotation( ) with `type=Stamp`	*Access to the font* `NotoSans-Bold` *must be configured unless the* `font` *option is supplied.*[1]
PDF_load_asset( ) with `type=Sound` *or* `Video`	*The option* `mimetype` *is now required.*
names of form fields	*Field names created with PDF_create_field( ) must be unique in the document. Fields with synchronized values can be created with PDF_create_fieldgroup( ) and a suitable* `fieldtype` *option.*
8-bit encodings	*The encoding definitions listed below are no longer available internally in PDFlib. While on Windows and Linux they are fetched from the operating system, they must be configured as external resource files on other systems (see PDFlib Tutorial):* `iso8859-2`, `iso8859-3`, `iso8859-4`, `iso8859-5`, `iso8859-6`, `iso8859-7`, `iso8859-8`, `iso8859-9`, `iso8859-10`, `iso8859-13`, `iso8859-14`, `iso8859-15`, `iso8859-16`, `cp1250`, `cp1251`, `cp1253`, `cp1254`, `cp1255`, `cp1256`, `cp1257`, `cp1258`
Standard Type 1 font `Courier`	*PDFlib 9 treated the Standard Type 1 font* `Courier` *in a special way to make certain glyphs beyond ISO 8859-1 available in Acrobat even without embedding. However, this technique isn't backed by ISO 32000 and is not supported by other PDF viewers.* *Workaround: configure an embeddable version of the* `Courier` *font.*
Host font names on macOS, i.e. retrieving fonts which are installed in the operating system	*Previously host fonts on macOS had to be retrieved with their old-style QuickDraw or PostScript names. This is no longer required since the standard TrueType/OpenType font name can be used (the »full font name« in the font's name table).* *For example, the semibold italic version of the font* `Warnock Pro` *previously had to be requested with the abbreviated name* »`Warnock Pro SmBd Ital`«. *PDFlib 10 accepts the full name* »`Warnock Pro Semibold Italic`«.
Transparency operations are not allowed in the definition of uncolorized glyphs in a Type 3 font per ISO 32000. PDFlib therefore rejects all uses of `opacityfill`, `opacitystroke`, `blendmode` *or* `softmask` *in such a glyph definition.*	*Transparency doesn't have any effect anyway in uncolorized Type 3 glyphs; remove all uses of transparency in the glyph description.*

1. *These fonts are included in the PDFlib distribution package. The PDFlib Tutorial describes several methods for font configuration, especially* `searchpath`. *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=Caret`, `FreeText`, `Line` *or* `Stamp`.

4 Binding-specific Changes in PDFlib 10

4.1 C Language Binding

Table 4.1 PDFlib features which require changes in C applications

PDFlib feature	***required change***
The `errorhandler` *callback for PDF_new2( ) no longer supports the* `errortype` *parameter which has been unused since PDFlib 8.*	*Change your custom* `errorhandler` *for PDF_new2( ) from* `typedef void (*errorproc_t) (PDF *p1, int errortype, const char *msg);` *to* `typedef void (*errorproc_t) (PDF *p1, const char *msg);`

4.2 C++ Language Binding

Table 4.2 PDFlib features which require changes in C++ applications

PDFlib feature	***required change***
Removed the `PDFCPP_USE_PDFLIB_NAMESPACE` *macro which disabled namespace support for PDFlib 7 compatibility.*	*Use the following in your application code:* `using namespace pdflib;`
Removed custom string types and the `PDFCPP_PDFLIB_WSTRING` *macro which disabled* `wstring` *support in favor of* `string` *for PDFlib 7 compatibility.*	*Use one of the standard C++ string types* `wstring`, `u8string`, `u16string`, `u32string`, *or* `string`
The C++ binding based on `std::wstring` *cannot be used on IBM Z systems since the compiler no longer supports conversion of EBCDIC string literals to UTF-32 (previously this was possible with the compiler option* `CONVLIT(,UNICODE)`).	*Use the new interfaces which are based on* `std::u16string` *and* `std::u32string`.

4.3 COM Binding

The COM binding is no longer supported. We recommend to use the .NET binding instead.

4.4 Objective-C Language Binding for macOS and iOS

Table 4.3 PDFlib features which require changes in Objective-C applications

PDFlib feature	***required change***
The name of the PDFlib framework is now identical for macOS and iOS.	*iOS applications only: change the framework name* `PDFlib_ios` *to* `PDFlib` *in the* `#import` *statement.*

4.5 Perl Language Binding

Table 4.4 PDFlib features which require changes in Perl applications

PDFlib feature	***required change***
PDF_set_option( ): the default value of the `stringformat` *option is now* `utf8`.	*No change required for the common case of Unicode-based applications which already set* `stringformat=utf8`. *Applications which work with 8-bit encodings must set* `stringformat=legacy`.

4.6 PHP Language Binding

Table 4.5 PDFlib features which require changes in PHP applications

PDFlib feature	***required change***
PDF_set_option( ): the default value of the `stringformat` *option is now* `utf8`.	*No change required for the common case of Unicode-based applications which already set* `stringformat=utf8`. *Applications which work with 8-bit encodings must set* `stringformat=legacy`.
The functional PDFlib API (deprecated since PDFlib 8) has been removed in favor of the object-oriented interface.	*Convert application code to the object-oriented interface.*
The delete( ) method (which was no-op anyway) has been removed.	*Remove all calls to PDFlib->delete( ) from application code.*

4.7 Ruby Language Binding

Table 4.6 PDFlib features which require changes in Ruby applications

PDFlib feature	***required change***
PDF_set_option( ): the default value of the `stringformat` *option is now* `utf8`.	*No change required for the common case of Unicode-based applications which already set* `stringformat=utf8`. *Applications which work with 8-bit encodings must set* `stringformat=legacy`.

5 PDFlib History

Table 5.1 Release history of PDFlib, PDFlib+PDI and PPS

PDFlib version	***release date***
PDFlib 4	*2001*
PDFlib 5	*2003*
PDFlib 6	*2004*
PDFlib 7	*2006*
PDFlib 8	*2009*
PDFlib 8.1	*2011*
PDFlib 9	*2013*
PDFlib 9.1	*2016*
PDFlib 9.2	*2019*
PDFlib 9.3	*2020*
PDFlib 9.3.1	*2021*
PDFlib 9.4	*2022*
PDFlib 10	*2022*