

The pdfoverlay package

David Purton*

2022/08/27 v1.3

Abstract

This pdfoverlay package provides a simple interface to overlay text on to an existing PDF. The text to be overlaid is typeset and positioned normally as you would any other L^AT_EX document. Some or all of the pages of the PDF can be included and not all pages of the PDF need have overlaid text. It's also possible to include text between pages of the PDF.

Contents

1	Introduction	1
2	Bug Reports and Feature Requests	2
3	Documentation	2
3.1	Basic Usage	2
3.2	Main Interface	2
4	Implementation	3
4.1	Messages	3
4.2	Private Variables and Helper Functions	4
4.3	Public Expl3 Functions	8
4.4	Public L ^A T _E X Interface	11
	Change History	12
	Index	12

*Email: dcpurton@marshwiggles.net

1 Introduction

It's often desirable to take an existing PDF and easily add annotations or text overlaying the PDF. This might arise if you wish to add comments to a PDF, fill in a PDF form, or add text to a PDF where space has been left for notes.

This package provides a simple interface to do this without having to resort to inserting one page at a time. Some or all of the pages of the PDF can be included and not all pages of the PDF need have overlaid text. It's also possible to include text between pages of the PDF.

Another advantage of this package is that the overlaid text can be set as normal flowing from one page to another or with manual page breaks if you wish. It's also possible to use any standard method to position text at arbitrary places on a given page.

2 Bug Reports and Feature Requests

Bug reports and feature requests can be made at the `pdfoverlay` package GitHub repository. See <https://github.com/dcpurton/pdfoverlay>.

3 Documentation

3.1 Basic Usage

At it's simplest, the only thing required is to specify the PDF to overlay with text and then begin typing your document. The pages of the PDF will be inserted as you go automatically. If your text takes up more pages than the PDF then any remaining text is inserted on blank pages.

```
\documentclass{article}
\usepackage{pdfoverlay}
\pdfoverlaySetPDF{filename.pdf}
\begin{document}
...
\end{document}
```

The PDF will be centered on and scaled to fit the document page size while keeping a fixed aspect ratio.

3.2 Main Interface

<code>\pdfoverlaySetPDF</code>	<code>\pdfoverlaySetPDF {<PDF filename>}</code>
<code>\pdfoverlay_set_pdf:n</code>	

Specify the *<PDF filename>* to overlay text on to. This function must be called first. It can be used either in the preamble or at any point in the document body. It is possible to call this more than once to include multiple PDF files. If the specified *<PDF filename>* cannot be found an error is generated.

<code>\pdfoverlaySetGraphicsOptions</code>	<code>\pdfoverlaySetGraphicsOptions {⟨Options⟩}</code>
<code>\pdfoverlay_set_graphics_options:n</code>	

Set the *⟨Options⟩* to be passed to `\includegraphics` when each page of the PDF is output. See the `graphicx` package documentation for valid options.

Note that it is not necessary to specify the `page` option as this is appended automatically in the right format. The default options are: `keepaspectratio, width=\paperwidth, height=\paperheight`.

<code>\pdfoverlayIncludeToPage</code>	<code>\pdfoverlayIncludeToPage {⟨page number⟩}</code>
<code>\pdfoverlay_include_to_page:n</code>	

Output all pages in the PDF file up until the specified *⟨page number⟩*. If the specified *⟨page number⟩* does not evaluate to an integer or does not exist in the PDF an error is generated.

<code>\pdfoverlayIncludeToLastPage</code>	<code>\pdfoverlayIncludeToLastPage</code>
<code>\pdfoverlay_include_to_last_page:</code>	

Output all remaining pages in the PDF file.

<code>\pdfoverlaySkipToPage</code>	<code>\pdfoverlaySkipToPage {⟨page number⟩}</code>
<code>\pdfoverlay_skip_to_page:n</code>	

Skip to the specified *⟨page number⟩*. This can be before or after the current page in the PDF file. If the specified *⟨page number⟩* does not evaluate to an integer or does not exist in the PDF an error is generated.

<code>\pdfoverlayPauseOutput</code>	<code>\pdfoverlayPauseOutput</code>
<code>\pdfoverlay_pause_output:</code>	

Pause outputting pages from the PDF file. Any subsequent text will be output on blank pages.

<code>\pdfoverlayResumeOutput</code>	<code>\pdfoverlayResumeOutput</code>
<code>\pdfoverlay_resume_output:</code>	

Resume outputting pages from the PDF file.

4 Implementation

```
1 ⟨*package⟩
2 ⟨@@=pdfoverlay⟩
3 \NeedsTeXFormat{LaTeX2e}[2020-10-01]
4 \ProvidesExplPackage{pdfoverlay}{2022/08/27}{1.3}
5   {Overlay text on an existing PDF document (DCP)}
```

The `graphicx` package is required.

```
6 \RequirePackage{graphicx}
```

Call `__pdfoverlay_output_pdf_page:` on every page.

```
7 \hook_gput_code:nnn { shipout/background } { pdfoverlay } { \__pdfoverlay_output_pdf_page: }
```

Add an empty `\hbox:n` to the end of the document if an action is pending. This is required to ensure that the last requested page from the PDF file is output even if there isn't any other content on the page.

```
8 \hook_gput_code:nnn { enddocument } { pdfoverlay } {
```

```

9   \bool_if:NT \g__pdfoverlay_action_pending_bool
10   {
11     \hbox:n { }
12   }
13 }

```

4.1 Messages

Error message when specified PDF file cannot be found.

#1: PDF file name

```

14 \msg_new:nnnn { pdfoverlay } { file-not-found }
15   { PDF~file~'#1'~not~found. }
16   { Unable~to~find~the~file~'#1'. \\\
17     Check~that~the~file~exists~and~you~have~spelt~it~correctly. }

```

Error message when no PDF file has been set.

```

18 \msg_new:nnnn { pdfoverlay } { file-not-set }
19   { PDF~file~not~set. }
20   { You~have~not~specified~a~PDF~file. \\\
21     Set~a~PDF~file~using~\pdfoverlaySetPDF. }

```

Error message when the requested page number cannot be found in the PDF file.

#1: PDF file name

#2: Requested page number

#3: Total number of pages in PDF

```

22 \msg_new:nnnn { pdfoverlay } { page-not-found }
23   { Page~not~found~in~PDF. }
24   { PDF~file~'#1'~does~not~contain~page~#2. \\\
25     Specify~a~page~between~1~and~#3. }

```

Error message when the requested page number to include to is less than the current page number being output from the PDF file.

#1: Requested page number

#2: Current page number in PDF

#3: Total number of pages in PDF

```

26 \msg_new:nnnn { pdfoverlay } { page-too-low }
27   { Requested~page~less~than~current~page~in~PDF. }
28   { You~have~requested~to~include~to~page~#1, \\\
29     but~the~current~page~is~already~at~page~#2. \\\
30     Specify~a~page~between~#2~and~#3. }

```

Error message when running in DVI mode. The pdfoverlay package does not support DVI mode for either pdf_latex or lua_latex.

```

31 \msg_new:nnnn { pdfoverlay } { dvi-mode }
32   { DVI~mode~not~supported. }
33   { DVI~mode~of~#1~is~not~supported. \\\
34     You~must~use~PDF~mode. }

```

Error message for unsupported engine. Only pdf_tex, lua_tex, and xet_ex are supported.

```

35 \msg_new:nnnn { pdfoverlay } { unsupported-engine }
36   { #1~not~supported. }
37   { The~#1~engine~is~not~supported. \\\
38     Use~one~of~pdftex,~luatex,~or~xetex. }

```

4.2 Private Variables and Helper Functions

<code>\g_pdfoverlay_pdf_file_name_tl</code>	Store the PDF file name. 39 <code>\tl_new:N \g_pdfoverlay_pdf_file_name_tl</code> (End definition for <code>\g_pdfoverlay_pdf_file_name_tl</code> .)
<code>\g_pdfoverlay_page_count_int</code>	Store the total number of pages in the PDF file. 40 <code>\int_new:N \g_pdfoverlay_page_count_int</code> (End definition for <code>\g_pdfoverlay_page_count_int</code> .)
<code>\g_pdfoverlay_page_int</code>	Keep track of the current page in the PDF file. 41 <code>\int_new:N \g_pdfoverlay_page_int</code> (End definition for <code>\g_pdfoverlay_page_int</code> .)
<code>\g_pdfoverlay_output_active_bool</code>	Flag to store whether to output pages from the PDF file or not. 42 <code>\bool_new:N \g_pdfoverlay_output_active_bool</code> (End definition for <code>\g_pdfoverlay_output_active_bool</code> .)
<code>\g_pdfoverlay_action_pending_bool</code>	Flag to store whether an action (<code>\pdfoverlay_include_to_page:n</code> or <code>\pdfoverlay_set_page:n</code>) is pending. If an action is pending, these functions will start a new page. 43 <code>\bool_new:N \g_pdfoverlay_action_pending_bool</code> (End definition for <code>\g_pdfoverlay_action_pending_bool</code> .)
<code>\g_pdfoverlay_graphics_options_clist</code>	Hold the options passed to <code>\includegraphics</code> when including a page from the PDF. The default options scale the PDF page to fit the document page while keeping the original PDF page aspect ratio. 44 <code>\clist_new:N \g_pdfoverlay_graphics_options_clist</code> 45 <code>\clist_set:Nn \g_pdfoverlay_graphics_options_clist</code> 46 <code>{</code> 47 <code> keepaspectratio ,</code> 48 <code> width = \paperwidth ,</code> 49 <code> height = \paperheight ,</code> 50 <code> page = \int_use:N \g_pdfoverlay_page_int</code> 51 <code>}</code> (End definition for <code>\g_pdfoverlay_graphics_options_clist</code> .)
<code>\g_pdfoverlay_pdf_page_coffin</code>	The current PDF page is set in this coffin which can then be positioned on the page. 52 <code>\coffin_new:N \g_pdfoverlay_pdf_page_coffin</code> (End definition for <code>\g_pdfoverlay_pdf_page_coffin</code> .)
<code>_pdfoverlay_output_pdf_page:</code>	Main function to output the current page of the PDF file. This is called before every page is shipped using the <code>shipout/background</code> hook. 53 <code>\cs_new_protected:Nn _pdfoverlay_output_pdf_page:</code> 54 <code>{</code>

Check if we are currently outputting pages, a PDF file has been set, and the requested page exists in the PDF file.

```

55   \bool_lazy_all:nT
56   {
57     { \bool_if_p:N \g__pdfoverlay_output_active_bool }
58     { \bool_not_p:n
59       { \tl_if_empty_p:N \g__pdfoverlay_pdf_file_name_tl } }
60     { \int_compare_p:n
61       { \c_zero_int <= \g__pdfoverlay_page_int
62         < \g__pdfoverlay_page_count_int } }
63     }
64     {

```

Increment the current page counter.

```

65     \int_gincr:N \g__pdfoverlay_page_int

```

Place requested page from the PDF file in the background of the document page with the specified format.

```

66     \__pdfoverlay_format_pdf_page:
67     \__pdfoverlay_place_pdf_page:

```

Set action pending flag to false.

```

68     \bool_gset_false:N \g__pdfoverlay_action_pending_bool
69   }
70 }

```

(End definition for __pdfoverlay_output_pdf_page:.)

`__pdfoverlay_format_pdf_page:` Format the PDF page with options from `\g__pdfoverlay_graphics_options_clist` and place it in `\g__pdfoverlay_pdf_page_coffin`.

```

71 \cs_new_protected:Nn \__pdfoverlay_format_pdf_page:
72 {
73   \hcoffin_gset:Nn \g__pdfoverlay_pdf_page_coffin
74   {
75     \use:x
76     {
77       \exp_not:N \includegraphics
78       [ \clist_use:Nn \g__pdfoverlay_graphics_options_clist { , } ]
79       { \g__pdfoverlay_pdf_file_name_tl }
80     }
81   }
82 }

```

(End definition for __pdfoverlay_format_pdf_page:.)

`__pdfoverlay_place_pdf_page:` Place the PDF page in the centre of the document page.

Although marked as private, this function could be redefined to position the PDF page at a different location on the document page.

This macro is called from within the `shipout/background` hook, which adds a picture environment into the background of the page with the (0, 0) coordinate in the top-left corner using a `\unitlength` of 1pt. It should therefore only receive `\put` commands or other commands suitable in a picture environment and the vertical coordinate values would normally be negative.

The PDF page to be placed is available as the coffin `\g__pdfoverlay_pdf_page_coffin`.

```

83 \cs_new_protected:Nn \__pdfoverlay_place_pdf_page:
84 {
85   \put ( 0.5 \paperwidth, -0.5 \paperheight )
86   {
87     \coffin_typeset:Nnnnn \g__pdfoverlay_pdf_page_coffin
88     { hc } { vc } { Opt } { Opt }
89   }
90 }

```

(End definition for __pdfoverlay_place_pdf_page:.)

__pdfoverlay_count_pdf_pages: Count the number of pages in the current \g__pdfoverlay_pdf_file_name_tl and store the result in \g__pdfoverlay_pdf_page_count_int. If no PDF has been set with \pdfoverlay_set_pdf then \g__pdfoverlay_pdf_page_count_int is set to zero.

```

91 \cs_new_protected:Nn \__pdfoverlay_count_pdf_pages:
92 {
93   \int_gzero:N \g__pdfoverlay_page_count_int
94   \tl_if_empty:NTF \g__pdfoverlay_pdf_file_name_tl
95   {
96     \msg_error:nn { pdfoverlay } { file-not-set }
97   }
98   {
99     \sys_if_engine_xetex:TF
100     {
101       \int_gset:Nn \g__pdfoverlay_page_count_int
102       {
103         \XeTeXpdfpagecount " \g__pdfoverlay_pdf_file_name_tl "
104       }
105     }
106     {
107       \str_if_exist:NF \c_sys_backend_str
108       {
109         \sys_load_backend:n { }
110       }
111       \sys_if_output_pdf:TF
112       {
113         \sys_if_engine_pdftex:TF
114         {
115           \pdfximage { \g__pdfoverlay_pdf_file_name_tl }
116           \int_gset_eq:NN
117           \g__pdfoverlay_page_count_int
118           \pdflastximagepages
119         }
120         {
121           \sys_if_engine luatex:TF
122           {
123             \saveimageresource
124             { \g__pdfoverlay_pdf_file_name_tl }
125             \int_gset_eq:NN
126             \g__pdfoverlay_page_count_int
127             \lastsavedimageresourcepages
128           }
129           {
130             \msg_error:nxx { pdfoverlay } { unsupported-engine }

```

```

131             { \c_sys_engine_str }
132         }
133     }
134 }
135 {
136     \msg_error:nnx { pdfoverlay } { dvi-mode }
137     { \c_sys_engine_str }
138 }
139 }
140 }
141 }

```

(End definition for `_pdfoverlay_count_pdf_pages:.`)

`_pdfoverlay_check_page_p:n` Check if a PDF file has been set and the requested page is valid. If not, generate suitable error messages.

`_pdfoverlay_check_page:nTF`

```

142 \prg_new_conditional:Nnn \_pdfoverlay_if_page_exists:n { p, T, F, TF }
143 {
144     \tl_if_empty:NTF \g__pdfoverlay_pdf_file_name_tl
145     {
146         \msg_error:nn { pdfoverlay } { file-not-set }
147         \prg_return_false:
148     }
149     {
150         \int_compare:nTF
151         {
152             \c_one_int <= #1 <= \g__pdfoverlay_page_count_int
153         }
154         {
155             \prg_return_true:
156         }
157         {
158             \msg_error:nnxxx { pdfoverlay } { page-not-found }
159             { \g__pdfoverlay_pdf_file_name_tl }
160             { \int_eval:n { #1 } }
161             { \int_use:N \g__pdfoverlay_page_count_int }
162             \prg_return_false:
163         }
164     }
165 }

```

(End definition for `_pdfoverlay_check_page:nTF.`)

4.3 Public Expl3 Functions

`\pdfoverlay_set_pdf:n` Specify a PDF file to overlay text on to.

```

166 \cs_new_protected:Nn \pdfoverlay_set_pdf:n
167 {

```

Test if the file exists and generate a suitable error if it doesn't.

```

168     \file_if_exist:nTF { #1 }
169     {
170         \tl_gset:Nn \g__pdfoverlay_pdf_file_name_tl { #1 }

```


Find the number of pages in the PDF file.

```
171     \_pdfoverlay_count_pdf_pages:
```

Initialise variables.

```
172     \int_gzero:N \g__pdfoverlay_page_int
173     \bool_gset_true:N \g__pdfoverlay_output_active_bool
174     \bool_gset_false:N \g__pdfoverlay_action_pending_bool
175   }
176   {
177     \msg_error:nnx { pdfoverlay } { file-not-found } { #1 }
178   }
179 }
```

(End definition for `\pdfoverlay_set_pdf:n`. This function is documented on page 2.)

`\pdfoverlay_set_graphics_options:n` Set the options as a comma separated list to be passed to `\includegraphics` when outputting each page from the PDF file. The `page` option is appended automatically in the correct format.

```
180 \cs_new_protected:Nn \pdfoverlay_set_graphics_options:n
181   {
182     \clist_gset:Nn \g__pdfoverlay_graphics_options_clist { #1 }
183     \clist_gput_right:Nn \g__pdfoverlay_graphics_options_clist
184       {
185         page = \int_use:N \g__pdfoverlay_page_int
186       }
187   }
```

(End definition for `\pdfoverlay_set_graphics_options:n`. This function is documented on page 2.)

`\pdfoverlay_include_to_page:n` Output all pages in the PDF file up until the specified page. Since the actual PDF page is inserted by `_pdfoverlay_output_pdf_page:` using the `shipout/background` hook, we just insert the required number of blank pages into the document.

```
188 \cs_new_protected:Nn \pdfoverlay_include_to_page:n
189   {
```

Check if a PDF file is set and the requested page exists.

```
190     \_pdfoverlay_if_page_exists:nT { #1 }
191     {
```

Check if the requested page is greater than the current page.

```
192         \int_compare:nTF
193           {
194             #1 >= \g__pdfoverlay_page_int + 1
195           }
196           {
```

Check if an action is pending and start a new page if so.

```
197         \bool_lazy_all:nT
198           {
199             { \bool_if_p:n { \g__pdfoverlay_action_pending_bool } }
200             { \int_compare_p:n { \g__pdfoverlay_page_int <
201               \g__pdfoverlay_page_count_int - 1 } }
202             { \int_compare_p:n { #1 != \g__pdfoverlay_page_int + 1 } }
203           }
204         {
```

```

205         \hbox:n { }
206         \clearpage
207     }

```

Loop until the specified page, inserting blank pages.

```

208     \int_while_do:nNnn { \g__pdfoverlay_page_int } < { #1 - 1 }
209     {
210         \hbox:n { }
211         \clearpage
212     }

```

Set the action pending flag, so final page is output even if no text is overlaid on it.

```

213     \bool_gset_true:N \g__pdfoverlay_action_pending_bool
214 }
215 {
216     \msg_error:nxxxx { pdfoverlay } { page-too-low }
217     { \int_eval:n { #1 } }
218     { \int_eval:n { \g__pdfoverlay_page_int + 1 } }
219     { \int_use:N \g__pdfoverlay_page_count_int }
220 }
221 }
222 }

```

(End definition for \pdfoverlay_include_to_page:n. This function is documented on page 2.)

\pdfoverlay_include_to_last_page: Output all remaining pages in the PDF file.

```

223 \cs_new_protected:Nn \pdfoverlay_include_to_last_page:
224 {
225     \pdfoverlay_include_to_page:n { \g__pdfoverlay_page_count_int }
226 }

```

(End definition for \pdfoverlay_include_to_last_page:.. This function is documented on page 3.)

\pdfoverlay_skip_to_page:n Skip to the specified page number in the PDF file.

```

227 \cs_new_protected:Nn \pdfoverlay_skip_to_page:n
228 {
229     \__pdfoverlay_if_page_exists:nT { #1 }
230     {

```

Check if an action is pending and start a new page if so.

```

231         \bool_if:nT { \g__pdfoverlay_action_pending_bool }
232         {
233             \hbox:n { }
234             \clearpage
235         }

```

Set the page number for the next page to be output.

```

236         \int_gset:Nn \g__pdfoverlay_page_int { #1 - 1 }

```

Set the action pending flag, so final page is output even if no text is overlaid on it.

```

237     \bool_gset_true:N \g__pdfoverlay_action_pending_bool
238 }
239 }

```

(End definition for \pdfoverlay_skip_to_page:n. This function is documented on page 3.)

`\pdfoverlay_pause_output:` Pause outputting pages from the PDF file.

```
240 \cs_new_protected:Nn \pdfoverlay_pause_output:
241 {
242   \bool_gset_false:N \g__pdfoverlay_output_active_bool
243 }
```

(End definition for \pdfoverlay_pause_output:. This function is documented on page 3.)

`\pdfoverlay_resume_output:` Resume outputting pages from the PDF file.

```
244 \cs_new_protected:Nn \pdfoverlay_resume_output:
245 {
246   \bool_gset_true:N \g__pdfoverlay_output_active_bool
247 }
```

(End definition for \pdfoverlay_resume_output:. This function is documented on page 3.)

4.4 Public L^AT_EX Interface

`\pdfoverlaySetPDF` Specify a PDF file to overlay text on to.

```
248 \NewDocumentCommand \pdfoverlaySetPDF { m }
249 {
250   \pdfoverlay_set_pdf:n { #1 }
251 }
```

(End definition for \pdfoverlaySetPDF. This function is documented on page 2.)

`\pdfoverlaySetGraphicsOptions` Set the options as a comma separated list to be passed to `\includegraphics` when outputting each page from the PDF file. The `page` option is appended automatically in the right format.

```
252 \NewDocumentCommand \pdfoverlaySetGraphicsOptions { m }
253 {
254   \pdfoverlay_set_graphics_options:n { #1 }
255 }
```

(End definition for \pdfoverlaySetGraphicsOptions. This function is documented on page 2.)

`\pdfoverlayIncludeToPage` Output all pages in the PDF file up until the specified page.

```
256 \NewDocumentCommand \pdfoverlayIncludeToPage { m }
257 {
258   \pdfoverlay_include_to_page:n { #1 }
259 }
```

(End definition for \pdfoverlayIncludeToPage. This function is documented on page 2.)

`\pdfoverlayIncludeToLastPage` Output all remaining pages in the PDF file.

```
260 \NewDocumentCommand \pdfoverlayIncludeToLastPage { }
261 {
262   \pdfoverlay_include_to_last_page:
263 }
```

(End definition for \pdfoverlayIncludeToLastPage. This function is documented on page 3.)

`\pdfoverlaySkipToPage` Skip ahead to the specified page number in the PDF file.

```
264 \NewDocumentCommand \pdfoverlaySkipToPage { m }
265 {
266   \pdfoverlay_skip_to_page:n { #1 }
267 }
```

(End definition for `\pdfoverlaySkipToPage`. This function is documented on page 3.)

`\pdfoverlayPauseOutput` Pause outputting pages from the PDF file.

```
268 \NewDocumentCommand \pdfoverlayPauseOutput { }
269 {
270   \pdfoverlay_pause_output:
271 }
```

(End definition for `\pdfoverlayPauseOutput`. This function is documented on page 3.)

`\pdfoverlayResumeOutput` Resume outputting pages from the PDF file.

```
272 \NewDocumentCommand \pdfoverlayResumeOutput { }
273 {
274   \pdfoverlay_resume_output:
275 }
```

(End definition for `\pdfoverlayResumeOutput`. This function is documented on page 3.)

```
276 \endpackage
```

Change History

v1.0	counting pages	1
General: First public release		1
v1.1	v1.2b	
General: Fix deprecated macros	General: Protect unexpandable macros	1
v1.2	v1.3	
General: Update to use <code>l3hooks</code>	General: Allow PDF file name to be	
v1.2a	passed in as macro	1
General: Ensure backend loaded before		

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	
<code>\</code>	<code>\bool_gset_true:N</code> .. 173, 213, 237, 246
	<code>\bool_if:NTF</code> 9
	<code>\bool_if:nTF</code> 231
	<code>\bool_if_p:N</code> 57
	<code>\bool_if_p:n</code> 199
	<code>\bool_lazy_all:nTF</code> 55, 197
B	
bool commands:	
<code>\bool_gset_false:N</code> 68, 174, 242

<code>\bool_new:N</code>	42, 43	<code>\msg_error:nnn</code>	130, 136, 177
<code>\bool_not_p:n</code>	58	<code>\msg_error:nnnnn</code>	158, 216
		<code>\msg_new:nnnn</code> ...	14, 18, 22, 26, 31, 35
C			
<code>\clearpage</code>	206, 211, 234		
clist commands:			
<code>\clist_gput_right:Nn</code>	183		
<code>\clist_gset:Nn</code>	182		
<code>\clist_new:N</code>	44		
<code>\clist_set:Nn</code>	45		
<code>\clist_use:Nn</code>	78		
coffin commands:			
<code>\coffin_new:N</code>	52		
<code>\coffin_typeset:Nnnnn</code>	87		
cs commands:			
<code>\cs_new_protected:Nn</code>	53, 71,		
	83, 91, 166, 180, 188, 223, 227, 240, 244		
E			
exp commands:			
<code>\exp_not:N</code>	77		
F			
file commands:			
<code>\file_if_exist:nTF</code>	168		
H			
hbox commands:			
<code>\hbox:n</code>	3, 11, 205, 210, 233		
hcoffin commands:			
<code>\hcoffin_gset:Nn</code>	73		
hook commands:			
<code>\hook_gput_code:nnn</code>	7, 8		
I			
<code>\includegraphics</code>	2, 5, 8, 11, 77		
int commands:			
<code>\int_compare:nTF</code>	150, 192		
<code>\int_compare_p:n</code>	60, 200, 202		
<code>\int_eval:n</code>	160, 217, 218		
<code>\int_gincr:N</code>	65		
<code>\int_gset:Nn</code>	101, 236		
<code>\int_gset_eq:NN</code>	116, 125		
<code>\int_gzero:N</code>	93, 172		
<code>\int_new:N</code>	40, 41		
<code>\int_use:N</code>	50, 161, 185, 219		
<code>\int_while_do:nNnn</code>	208		
<code>\c_one_int</code>	152		
<code>\c_zero_int</code>	61		
L			
<code>\lastsavedimageresourcepages</code>	127		
M			
msg commands:			
<code>\msg_error:nn</code>	96, 146	<code>\NeedsTeXFormat</code>	3
		<code>\NewDocumentCommand</code>	
		248, 252, 256, 260, 264, 268, 272
N			
P			
<code>\paperheight</code>	2, 49, 85		
<code>\paperwidth</code>	2, 48, 85		
<code>\pdflastximagepages</code>	118		
pdfoverlay commands:			
<code>\pdfoverlay_include_to_last_-</code>			
<code>page:</code>	3, 223, 223, 262		
<code>\pdfoverlay_include_to_page:n</code> ...			
.....	2, 5, 188, 188, 225, 258		
<code>\pdfoverlay_pause_output:</code>			
.....	3, 240, 240, 270		
<code>\pdfoverlay_resume_output:</code>			
.....	3, 244, 244, 274		
<code>\pdfoverlay_set_graphics_-</code>			
<code>options:n</code>	2, 180, 180, 254		
<code>\pdfoverlay_set_page:n</code>	5		
<code>\pdfoverlay_set_pdf</code>	6		
<code>\pdfoverlay_set_pdf:n</code>	2, 166, 166, 250		
<code>\pdfoverlay_skip_to_page:n</code>			
.....	3, 227, 227, 266		
pdfoverlay internal commands:			
<code>\g__pdfoverlay_action_pending_-</code>			
<code>bool</code>	9, 43, 68, 174, 199, 213, 231, 237		
<code>__pdfoverlay_check_page:nTF</code> ..	142		
<code>__pdfoverlay_check_page_p:n</code> ..	142		
<code>__pdfoverlay_count_pdf_pages:</code> ..			
.....	91, 91, 171		
<code>__pdfoverlay_format_pdf_page:</code> ..			
.....	66, 71, 71		
<code>\g__pdfoverlay_graphics_options_-</code>			
<code>clist</code>	6, 44, 78, 182, 183		
<code>__pdfoverlay_if_page_exists:n</code> ..	142		
<code>__pdfoverlay_if_page_exists:nTF</code>			
.....	190, 229		
<code>\g__pdfoverlay_output_active_-</code>			
<code>bool</code>	42, 57, 173, 242, 246		
<code>__pdfoverlay_output_pdf_page:</code> ..			
.....	3, 9, 7, 53, 53		
<code>\g__pdfoverlay_page_count_int</code> ...			
.....	40, 62, 93,		
	101, 117, 126, 152, 161, 201, 219, 225		
<code>\g__pdfoverlay_page_int</code>			
.....	41, 50, 61, 65,		
	172, 185, 194, 200, 202, 208, 218, 236		

<code>\g_pdfoverlay_pdf_file_name_tl</code>	6, 39 , 59 , 79 , 94 , 103 , 115 , 124 , 144 , 159 , 170		
<code>\g_pdfoverlay_pdf_page_coffin</code>	6, 52 , 73 , 87		
<code>\g_pdfoverlay_pdf_page_count_int</code>	6		
<code>__pdfoverlay_place_pdf_page:</code>	67, 83 , 83		
<code>\pdfoverlayIncludeToLastPage</code>	3, 260		
<code>\pdfoverlayIncludeToPage</code>	2, 256		
<code>\pdfoverlayPauseOutput</code>	3, 268		
<code>\pdfoverlayResumeOutput</code>	3, 272		
<code>\pdfoverlaySetGraphicsOptions</code>	2, 252		
<code>\pdfoverlaySetPDF</code>	2, 21 , 248		
<code>\pdfoverlaySkipToPage</code>	3, 264		
<code>\pdfximage</code>	115		
prg commands:			
<code>\prg_new_conditional:Nnn</code>	142		
<code>\prg_return_false:</code>	147, 162		
<code>\prg_return_true:</code>	155		
<code>\ProvidesExplPackage</code>	4		
<code>\put</code>	6, 85		
		R	
<code>\RequirePackage</code>	6		
			S
		<code>\saveimageresource</code>	123
		str commands:	
		<code>\str_if_exist:NTF</code>	107
		sys commands:	
		<code>\c_sys_backend_str</code>	107
		<code>\c_sys_engine_str</code>	131, 137
		<code>\sys_if_engine luatex:TF</code>	121
		<code>\sys_if_engine pdftex:TF</code>	113
		<code>\sys_if_engine xetex:TF</code>	99
		<code>\sys_if_output_pdf:TF</code>	111
		<code>\sys_load_backend:n</code>	109
			T
		tl commands:	
		<code>\tl_gset:Nn</code>	170
		<code>\tl_if_empty:NTF</code>	94, 144
		<code>\tl_if_empty_p:N</code>	59
		<code>\tl_new:N</code>	39
			U
		<code>\unitlength</code>	6
		use commands:	
		<code>\use:n</code>	75
			X
		<code>\XeTeXpdfpagecount</code>	103