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 LaTeX 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

*Email: dcpurton@marshwiggle.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

\pdfoverlaySetPDF
\pdfoverlay_set_pdf:n

\pdfoverlaySetPDF {⟨*PDF filename*⟩}

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.

\pdfoverlaySetGraphicsOptions
\pdfoverlay_set_graphics_options:n

\pdfoverlaySetGraphicsOptions {⟨*Options*⟩}

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`.

\pdfoverlayIncludeToPage
\pdfoverlay_include_to_page:n

\pdfoverlayIncludeToPage {⟨*page number*⟩}

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.

\pdfoverlayIncludeToLastPage
\pdfoverlay_include_to_last_page:

\pdfoverlayIncludeToLastPage

Output all remaining pages in the PDF file.

\pdfoverlaySkipToPage
\pdfoverlay_skip_to_page:n

\pdfoverlaySkipToPage {⟨*page number*⟩}

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.

\pdfoverlayPauseOutput
\pdfoverlay_pause_output:

\pdfoverlayPauseOutput

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

\pdfoverlayResumeOutput
\pdfoverlay_resume_output:

\pdfoverlayResumeOutput

Resume outputting pages from the PDF file.

4 Implementation

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

The graphicx package is required.

```
\RequirePackage{graphicx}
```

Call `\__pdfoverlay_output_pdf_page:` on every page.

```
\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.

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

```
  \bool_if:NT \g__pdfoverlay_action_pending_bool
    {
      \hbox:n { }
    }
}
```

4.1 Messages

Error message when specified PDF file cannot be found.
#1: PDF file name

```
\msg_new:nnnn { pdfoverlay } { file-not-found }
  { PDF~file~‘#1’~not~found. }
  { Unable~to~find~the~file~‘#1’. \\
    Check~that~the~file~exists~and~you~have~spelt~it~correctly. }
```

Error message when no PDF file has been set.

```
\msg_new:nnnn { pdfoverlay } { file-not-set }
  { PDF~file~not~set. }
  { You~have~not~specified~a~PDF~file. \\
    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

```
\msg_new:nnnn { pdfoverlay } { page-not-found }
  { Page~not~found~in~PDF. }
  { PDF~file~‘#1’~does~not~contain~page~#2. \\
    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

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

Error message when running in DVI mode. The pdfoverlay package does not support DVI mode for either `pdflatex` or `lualatex`.

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

Error message for unsupported engine. Only `pdftex`, `luatex`, and `xetex` are supported.

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

4.2 Private Variables and Helper Functions

\g__pdfoverlay_pdf_file_name_tl Store the PDF file name.

```
\tl_new:N \g__pdfoverlay_pdf_file_name_tl
```

(*End definition for* `\g__pdfoverlay_pdf_file_name_tl`.)

\g__pdfoverlay_page_count_int Store the total number of pages in the PDF file.

```
\int_new:N \g__pdfoverlay_page_count_int
```

(*End definition for* `\g__pdfoverlay_page_count_int`.)

\g__pdfoverlay_page_int Keep track of the current page in the PDF file.

```
\int_new:N \g__pdfoverlay_page_int
```

(*End definition for* `\g__pdfoverlay_page_int`.)

\g__pdfoverlay_output_active_bool Flag to store whether to output pages from the PDF file or not.

```
\bool_new:N \g__pdfoverlay_output_active_bool
```

(*End definition for* `\g__pdfoverlay_output_active_bool`.)

\g__pdfoverlay_action_pending_bool Flag to store whether an action (`\pdfoverlay_include_to_page:n` or `\pdfoverlay_set_page:n`) is pending. If an action is pending, these functions will start a new page.

```
\bool_new:N \g__pdfoverlay_action_pending_bool
```

(*End definition for* `\g__pdfoverlay_action_pending_bool`.)

\g__pdfoverlay_graphics_options_clist Hold the options passed to `\includegraphics` 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.

```
\clist_new:N \g__pdfoverlay_graphics_options_clist
\clist_set:Nn \g__pdfoverlay_graphics_options_clist
  {
    keepaspectratio ,
    width  = \paperwidth ,
    height = \paperheight ,
    page = \int_use:N \g__pdfoverlay_page_int
  }
```

(*End definition for* `\g__pdfoverlay_graphics_options_clist`.)

\g__pdfoverlay_pdf_page_coffin The current PDF page is set in this coffin which can then be positioned on the page.

```
\coffin_new:N \g__pdfoverlay_pdf_page_coffin
```

(*End definition for* `\g__pdfoverlay_pdf_page_coffin`.)

__pdfoverlay_output_pdf_page: Main function to output the current page of the PDF file. This is called before every page is shipped using the `shipout/background` hook.

```
\cs_new_protected:Nn \__pdfoverlay_output_pdf_page:
  {
```

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

```
\bool_lazy_all:nT
  {
    { \bool_if_p:N \g__pdfoverlay_output_active_bool }
    { \bool_not_p:n
        { \tl_if_empty_p:N \g__pdfoverlay_pdf_file_name_tl } }
    { \int_compare_p:n
        { \c_zero_int <= \g__pdfoverlay_page_int
          < \g__pdfoverlay_page_count_int } }
  }
  {
```

Increment the current page counter.

```
    \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.

```
    \__pdfoverlay_format_pdf_page:
    \__pdfoverlay_place_pdf_page:
```

Set action pending flag to false.

```
    \bool_gset_false:N \g__pdfoverlay_action_pending_bool
  }
}
```

(*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`.

```
\cs_new_protected:Nn \__pdfoverlay_format_pdf_page:
  {
    \hcoffin_gset:Nn \g__pdfoverlay_pdf_page_coffin
      {
        \use:x
          {
            \exp_not:N \includegraphics
              [ \clist_use:Nn \g__pdfoverlay_graphics_options_clist { , } ]
              { \g__pdfoverlay_pdf_file_name_tl }
          }
      }
  }
```

(*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`.

```
\cs_new_protected:Nn \__pdfoverlay_place_pdf_page:
  {
    \put ( 0.5 \paperwidth, -0.5 \paperheight )
      {
        \coffin_typeset:Nnnnn \g__pdfoverlay_pdf_page_coffin
          { hc } { vc } { 0pt } { 0pt }
      }
  }
```

(*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.

```
\cs_new_protected:Nn \__pdfoverlay_count_pdf_pages:
  {
    \int_gzero:N \g__pdfoverlay_page_count_int
    \tl_if_empty:NTF \g__pdfoverlay_pdf_file_name_tl
      {
        \msg_error:nn { pdfoverlay } { file-not-set }
      }
      {
        \sys_if_engine_xetex:TF
          {
            \int_gset:Nn \g__pdfoverlay_page_count_int
              {
                \XeTeXpdfpagecount " \g__pdfoverlay_pdf_file_name_tl "
              }
          }
          {
            \str_if_exist:NF \c_sys_backend_str
              {
                \sys_load_backend:n { }
              }
            \sys_if_output_pdf:TF
              {
                \sys_if_engine_pdftex:TF
                  {
                    \pdfximage { \g__pdfoverlay_pdf_file_name_tl }
                    \int_gset_eq:NN
                      \g__pdfoverlay_page_count_int
                      \pdflastximagepages
                  }
                  {
                    \sys_if_engine_luatex:TF
                      {
                        \saveimageresource
                          { \g__pdfoverlay_pdf_file_name_tl }
                        \int_gset_eq:NN
                          \g__pdfoverlay_page_count_int
                          \lastsavedimageresourcepages
                        }
                      {
                        \msg_error:nnx { pdfoverlay } { unsupported-engine }
```

```
                        { \c_sys_engine_str }
                    }
                }
            }
            {
              \msg_error:nnx { pdfoverlay } { dvi-mode }
                { \c_sys_engine_str }
            }
        }
    }
}
```

(*End definition for* `\__pdfoverlay_count_pdf_pages:`.)

`\__pdfoverlay_check_page_p:n`
`\__pdfoverlay_check_page:nTF`

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

```
\prg_new_conditional:Nnn \__pdfoverlay_if_page_exists:n { p, T, F, TF }
  {
    \tl_if_empty:NTF \g__pdfoverlay_pdf_file_name_tl
      {
        \msg_error:nn { pdfoverlay } { file-not-set }
        \prg_return_false:
      }
      {
        \int_compare:nTF
          {
            \c_one_int <= #1 <= \g__pdfoverlay_page_count_int
          }
          {
            \prg_return_true:
          }
          {
            \msg_error:nnxxx { pdfoverlay } { page-not-found }
              { \g__pdfoverlay_pdf_file_name_tl }
              { \int_eval:n { #1 } }
              { \int_use:N \g__pdfoverlay_page_count_int }
            \prg_return_false:
          }
      }
  }
```

(*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.

```
\cs_new_protected:Nn \pdfoverlay_set_pdf:n
  {
```

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

```
    \file_if_exist:nTF { #1 }
      {
        \tl_gset:Nn \g__pdfoverlay_pdf_file_name_tl { #1 }
```

Find the number of pages in the PDF file.

```
      \__pdfoverlay_count_pdf_pages:
```

Initialise variables.

```
      \int_gzero:N \g__pdfoverlay_page_int
      \bool_gset_true:N \g__pdfoverlay_output_active_bool
      \bool_gset_false:N \g__pdfoverlay_action_pending_bool
    }
    {
      \msg_error:nnx { pdfoverlay } { file-not-found } { #1 }
    }
}
```

(*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.

```
\cs_new_protected:Nn \pdfoverlay_set_graphics_options:n
  {
    \clist_gset:Nn \g__pdfoverlay_graphics_options_clist { #1 }
    \clist_gput_right:Nn \g__pdfoverlay_graphics_options_clist
      {
        page = \int_use:N \g__pdfoverlay_page_int
      }
  }
```

(*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.

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

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

```
    \__pdfoverlay_if_page_exists:nT { #1 }
      {
```

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

```
        \int_compare:nTF
          {
              #1 >= \g__pdfoverlay_page_int + 1
          }
          {
```

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

```
            \bool_lazy_all:nT
              {
                { \bool_if_p:n { \g__pdfoverlay_action_pending_bool } }
                { \int_compare_p:n { \g__pdfoverlay_page_int <
                                     \g__pdfoverlay_page_count_int - 1 } }
                { \int_compare_p:n { #1 != \g__pdfoverlay_page_int + 1 } }
              }
              {
```

```
            \hbox:n { }
            \clearpage
          }
```

Loop until the specified page, inserting blank pages.

```
          \int_while_do:nNnn { \g__pdfoverlay_page_int } < { #1 - 1 }
            {
              \hbox:n { }
              \clearpage
            }
```

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

```
          \bool_gset_true:N \g__pdfoverlay_action_pending_bool
        }
        {
          \msg_error:nnxxx { pdfoverlay } { page-too-low }
            { \int_eval:n { #1 } }
            { \int_eval:n { \g__pdfoverlay_page_int + 1 } }
            { \int_use:N \g__pdfoverlay_page_count_int }
        }
    }
}
```

(*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.

```
\cs_new_protected:Nn \pdfoverlay_include_to_last_page:
  {
      \pdfoverlay_include_to_page:n { \g__pdfoverlay_page_count_int }
  }
```

(*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.

```
\cs_new_protected:Nn \pdfoverlay_skip_to_page:n
  {
    \__pdfoverlay_if_page_exists:nT { #1 }
      {
```

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

```
        \bool_if:nT { \g__pdfoverlay_action_pending_bool }
          {
            \hbox:n { }
            \clearpage
          }
```

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

```
        \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.

```
        \bool_gset_true:N \g__pdfoverlay_action_pending_bool
      }
  }
```

(*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.

```
\cs_new_protected:Nn \pdfoverlay_pause_output:
  {
    \bool_gset_false:N \g__pdfoverlay_output_active_bool
  }
```

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

\pdfoverlay_resume_output: Resume outputting pages from the PDF file.

```
\cs_new_protected:Nn \pdfoverlay_resume_output:
  {
    \bool_gset_true:N \g__pdfoverlay_output_active_bool
  }
```

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

4.4 Public LaTeX Interface

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

```
\NewDocumentCommand \pdfoverlaySetPDF { m }
  {
    \pdfoverlay_set_pdf:n { #1 }
  }
```

(*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.

```
\NewDocumentCommand \pdfoverlaySetGraphicsOptions { m }
  {
    \pdfoverlay_set_graphics_options:n { #1 }
  }
```

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

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

```
\NewDocumentCommand \pdfoverlayIncludeToPage { m }
  {
    \pdfoverlay_include_to_page:n { #1 }
  }
```

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

\pdfoverlayIncludeToLastPage Output all remaining pages in the PDF file.

```
\NewDocumentCommand \pdfoverlayIncludeToLastPage { }
  {
    \pdfoverlay_include_to_last_page:
  }
```

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

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

```
\NewDocumentCommand \pdfoverlaySkipToPage { m }
  {
    \pdfoverlay_skip_to_page:n { #1 }
  }
```

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

\pdfoverlayPauseOutput Pause outputting pages from the PDF file.

```
\NewDocumentCommand \pdfoverlayPauseOutput { }
  {
    \pdfoverlay_pause_output:
  }
```

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

\pdfoverlayResumeOutput Resume outputting pages from the PDF file.

```
\NewDocumentCommand \pdfoverlayResumeOutput { }
  {
    \pdfoverlay_resume_output:
  }
```

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

⟨/package⟩

Change History

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.

R

S

T

U

X