pdfcomment.sty

v3.0b

A user-friendly interface to PDF annotations

2026-06-15

Package author:
Ulrike Fischer/Josef Kleber

Introduction to Version 3.0

The original author of the package, Josef Kleber, died some time ago and starting with version 3.0 the package has a new maintainer: Ulrike Fischer.
In this version the code has been splitted:

- If the PDF management is *not* loaded, the original package code from version 2.4a is used (with a few small adaptions for the split and some bug report).
- If the PDF management is loaded, either with `\RequirePackage{pdfmanagement}` or with `\DocumentMetadata`, then the annotations are created with the code from the PDF management and (basic) tagging support has added.

  This is big change, it involved lots of clean up and code changes and quite probably not everything will work from the start.

  Please report bugs and issues at https://github.com/LaTeX-Package-Repositories/pdfcomment

Documentation

Abstract

For a long time `pdflatex` has offered the command `\pdfannot` for inserting arbitrary PDF annotations. However, the command is presented in a form where additional knowledge of the definition of the PDF format is indispensable. This package is an answer to the – occasional – questions in newsgroups, about how one could use the comment function of `Adobe Reader`. At least for the writer of LaTeX code, the package offers a convenient and user-friendly means of using `\pdfannot` to provide comments in PDF files. Since version v1.1, `pdfcomment.sty` also supports:

LaTeX → `dvips` → `ps2pdf`, LaTeX → `dvipdfmx`[1] XeLaTeX and LuaLaTeX.

Unfortunately, support of PDF annotations by PDF viewers may vary. The reference viewer for the development of this package is `Adobe Reader`.

If you can't see this annotation  you are definitely using the wrong PDF viewer!

Required packages for using `pdfcomment.sty`

`hyperref` (v6.76a [2007/04/09]), `zref` (v1.8 [2007/04/22]), `xkeyval`, `etoolbox`, `luatex85`, `ifpdf`, `iflatex`, `ifthen`, `calc`, `marginnote`, `datetime2`, `refcount`, `soulpos` and the packages loaded by them.

[1] only with style option `dvipdfmx`

Contents

Acknowledgment

I want to thank the following persons for contributions to the development of this package:

- **Javier Bezos** for the development of `soulpos`, which allows much better support of non standard text cases in `\pdfmarkupcomment`.
- **Alexander Grahn** for contributing a patch for other drivers

  LaTeX → `dvips` → `ps2pdf`, LaTeX → `dvipdfmx` and XeLaTeX.

- **Ulrike Fischer** for answering my stupid questions on d.c.t.t. for making the avatar and style system possible and much more.
- **Christian Feuersänger** for contributing new ideas in form of his package `pdfmarginpar.sty` and for solving the printing problem[2] of PDF annotations and popups.
- **Ross Moore** for contributing ideas and code for his own feature request to support `\pdfmarkupcomment` in math mode.
- **Heiko Oberdiek** also for answering my stupid questions on d.c.t.t., as well as for the development of dozens of very usefull packages, especially `hyperref.sty`, `hycolor.sty` and `zref.sty`, which made this package possible.
- **Herbert Voß** also for answering my stupid questions on d.c.t.t.

I also want to thank the following persons for bug reports, feature requests, ...: Til Birnstiel, Jannis von Buttlar, Gabriel Cardona, Thomas Feller, Florent Chervet, Jin-Hwan Cho, Marcel Dausend, Andrew Dawson, diabonas, Max Funk, Zvi Gilboa, Thomas König, Marc-André Michel, Guillaume Millet, Fritz Moore, Michael Niedermair, Stefan Pinnow, René Schwarz

[2] see section 4

1 Options

1.1 Global options

1.1.1 `final`

The option `final` will set the package to final mode. The PDF annotations will not be typeset and will not influence line breaking. Use the local option `disable` if you want to disable single PDF annotations. (see: 1.2.33)

1.1.2 `draft`

The option `draft` (default) will set the package to draft mode. Therefore, the PDF annotations will be typeset.

1.1.3 `dvipdfmx`

If you want to use the driver `dvipdfmx` for creating your documents, you have to use the option `dvipdfmx`. The other drivers are recognized automatically.
Version 3.0: If the PDF management is loaded the option is ignored. The backend has to be declared with `backend=dvipdfmx` when loading the PDF management.

1.1.4 `version`

Version v2.0 of `pdfcomment.sty` introduced a bugfix that removes unwanted whitespace before the comment commands. This may also change the reference point for options like `hoffset`. If you want to preserve the old behavior for older documents simply use `version=1`. Otherwise `version=2` is used by default!
Version 3.0: If the PDF management is loaded and the new code is used this key is ignored.

1.2 Local options

The following options are useable as options for the commands presented in sections 2 and 3, as well as style options. As style options they have global effect, whereas they have only local effect when used in commands.

1.2.1 `id`

You can use the option `id` to give a comment an unique identifier that can then be used by the option `replyto` of the `\pdfreply` command to reference the comment.

1.2.2 `replyto`

You can use the option `replyto` in the `\pdfreply` command. The value is a id that has been set on another comment with the `id` option.

1.2.3 `subject`

You can use the option `subject` for defining the subject of the PDF popup annotations.

1.2.4 author

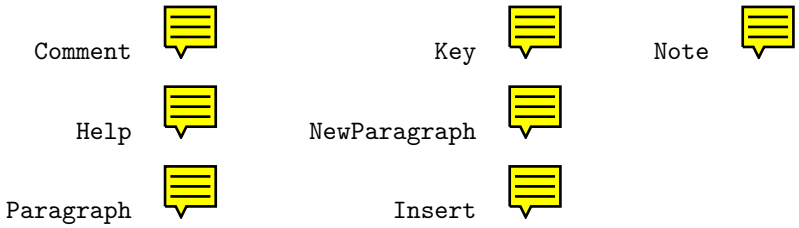
You can use the option `author` for defining the author of the PDF popup annotations.

1.2.5 opacity (1.0)

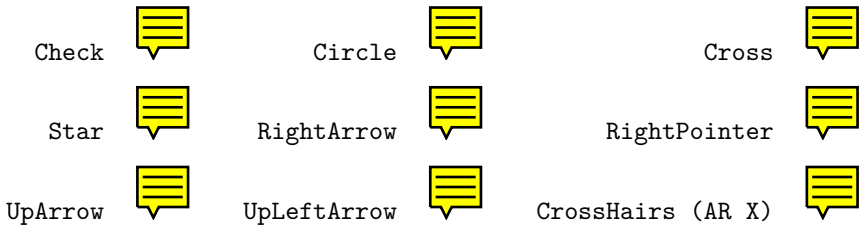
You can use the option `opacity` for defining the opacity of PDF annotations with values between 0 (transparent) and 1 (not transparent). If you want to print PDF popup annotations with transparency you will have to use the option `open=true`.

1.2.6 icon (Comment)

You can use the option `icon` for defining the graphic used for the PDF text annotations. The following icons are defined as mandatory by the PDF reference:



Furthermore the following icons are also supported by `Adobe Reader` and some other PDF viewers:



1.2.7 deadline

You can use the option `deadline` for defining a time limit until a problem has to be solved. The deadline will be added at the end of the comment, as well as in the list of comments, if an appropriate list style is chosen.

1.2.8 color ([0 0 1] (blue))

You can use the option `color` for defining the color of PDF annotations in the form {0.34 0.56 0.12} (RGB). If you are using the additional package `xcolor` you can use predefined color names, as well as the available optional color names. Furthermore you can use the command `\definecolor` to define your own named colors. Please take a look at the provided example files. They show the different possibilities of defining colors.
By definition the PDF specification allows four different color spaces with different numbers of color values:

#	color space
0	transparent
1	grey scale
3	RGB
4	CMYK

1.2.9 `icolor`

You can use the option `icolor` for defining the so called 'inner color', which is used by some PDF annotations, like arrowheads or the inner area of PDF circle annotation. For the rest, the provisions of the option `color` shall apply.

1.2.10 `open` (`true`,`false`)

You can use the option `open` for defining the opening status of the PDF popup annotations.
If you want to print the PDF popup annotations (with transparency) you will have to use the option `open=true`.

1.2.11 `hspace` (`0pt`)

You can use the option `hspace` for defining the horizontal space after the PDF text annotations, otherwise the PDF text annotations will overlay the text.

1.2.12 `type`

You can use the option `type` for defining the type of PDF annotation, if a comment command supports more than one PDF annotations:

comment command	possible types
`\pdffreetextcomment`	freetext, callout, typewriter
`\pdflinecomment`	line, polyline, polygon

1.2.13 `font` (`Helv`)

You can use the option `font` for defining the font of PDF freetext annotations, like all fonts installed in the system, which don't contain a space in their name. Although `Adobe Reader` shows these fonts as embedded, they are not embedded in reality, but the PDF file just contains a reference[3] to the font (default: Helvetica)! Therefore you should use this option quite carefully. In the case of document exchange between several authors you should just use fonts, which are available on all computer systems, like the so called 'standard 14 fonts'.

1.2.14 `fontsize` (`12bp`)

You can use the option `fontsize` for defining the fontsize of PDF freetext annotations.

[3] Adobe Acrobat offers the possibility to embed the fonts in the PS → PDF conversion

1.2.15 fontcolor ([0 0 0] (black))

You can use the option `fontcolor` for defining the font color in PDF freetext annotations, which must be a RGB color.

1.2.16 line

You can use the option `line` for defining the coordinates of lines in certain PDF annotations, e. g. in the form $\{x_1\ y_1\ x_2\ y_2\}$. The origin is in the bottom left page corner. The given numbers will be interpreted as Postscript points (LaTeX unit: bp (big points)), as usual in PDF documents.

type of line	# points
line	$n = 2$
polyline, polygon	$n > 2$
callout line	$n = 3$

In general, the points must be given from line begin to line end. For callout lines the points must be given from line end to line begin. Please note the example file `example.pdf`!
To avoid the method 'trail and error' you can load the PDF file with the Ghostscript viewer, which is capable of showing the cursor position in its status line.

1.2.17 linewidth (1bp)

You can use the option `linewidth` for defining the line width of PDF annoations.

1.2.18 linebegin (/None)

You can use the option `linebegin` for defining the arrow type a  line begin. The example file `example.pdf` shows all possible types of arrows.

1.2.19 lineend (/OpenArrow)

You can use the option `lineend` for defining the arrow type at the line end.

1.2.20 linesep (0.5cm)

You can use the option `linesep` for defining the horizontal space between text and line in the command `\pdfsidelinecomment`.

1.2.21 borderstyle (solid,dashed)

You can use the option `borderstyle` for defining the line style.

1.2.22 dashstyle ({3 3})

You can use the option `dashstyle` for defining the dash style, e. g. {5 3} (5 points line, 3 points space).

1.2.23 bse (none,cloudy)

You can use the option `bse` for defining the 'border style effect'.

1.2.24 bsei (1)

You can use the option bsei for defining the 'border style effect intensity' (size of cloud elements). The PDF reference suggests values between 1 and 2.

1.2.25 caption (none,inline,top)

You can use the option caption for defining the caption type of lines.
The options inline and top shows the comment 'inline' and on top of the line. While using none the comment will be shown in a PDF popup annotation.

1.2.26 captionhoffset (0pt)

You can use the option captionhoffset for defining a horizontal offset, that is a horizontal shift of the line caption.

1.2.27 captionvoffset (0pt)

You can use the option captionvoffset for definig a vertical offset of the line caption.

1.2.28 voffset (0pt)

You can use the option voffset for defining a vertical offset of the PDF annotations, that is a vertical shift for the given length.

1.2.29 hoffset (0pt)

You can use the option hoffset for defining a horizontal offset of the PDF annotations.

1.2.30 width (0pt)

You can use the option width for defining the width of PDF annotations, e. g. the width of FreeText annotations. PDF text annatations have a width of 0pt by definition.

1.2.31 height (0pt)

You can use the option height for definig the height of PDF annotations. PDF text annatations have a height of \normalbaselineskip by definition.

1.2.32 markup (Highlight,Underline,Squiggly,StrikeOut)

You can use the option markup for defining the type of the PDF text markup annotation.

1.2.33 disable (true,false)

You can use the option disable for switching off single PDF annotation. For switching off all PDF annotations use the global option final.

1.2.34 date

You can use the option `date` to assign a date to PDF annotations. If no explicit date is assigned the date and time of the creation of the PDF document is used. The input has to be in PDF date format, e.g.:

```
D:YYYYMMDDhhmmss+TZ
D:20101224153657+01'00'
```

1.2.35 timezone (+00'00')

You only have to use the option `timezone`, if you are using automatic date setting and XeLaTeX. Otherwise the `pdflatex` primitive \pdfcreationdate will be used. `timezone`=+01'00' sets the timezone to central european time CET (default: +00'00' (GMT)).

1.2.36 avatar

With the option `avatar` you can load the option lists, that were predefined with the command `\defineavatar` to avoid annoying typing.

1.2.37 style

With the option `style` you can also load predefined option lists for spliting personal and stylistic options, e. g. `avatar=Josef, style=MyComment`.

1.2.38 mathstyle (\textstyle)

With the option `mathstyle` you can correct the wrong size of PDF annotations produced with `\pdfmarkupcomment` or `\pdftooltip` in math mode by setting the math style to `\textstyle`, `\displaystyle`, `\scriptstyle` or `\scriptscriptstyle`. This option is only defined for `\pdfmarkupcomment` and `\pdftooltip`, so you can not set it globally!

1.2.39 printSOfinal (true,false)

With the option `printSOfinal` you can automatically delete PDF StrikeOut markup annotations including the text while using one of the options `final` or `disable`.

2 Environments

2.1 Comment environments

2.1.1 pdfsidelinecomment

`\begin{pdfsidelinecomment}`
`[`⟨*options*⟩`]`⟨*comment*⟩
…
`\end{pdfsidelinecomment}`

Possible options: `avatar`, `style`, `subject`, `author`, `color`, `icolor`, `opacity`, `linewidth`, `linebegin`, `lineend`, `linesep`, `borderstyle`, `dashstyle`, `caption`, `captionhoffset`, `captionvoffset`, `disable`, `date`, `timezone`

With the environment `pdfsidelinecomment` you can comment complete parts of a page in the form of two lines in the margins.

Limitations:

- The PDF sideline annotation must not be longer than one page, otherwise the recognition of the page break will fail.
- While using dvi files in the meantime you have to use a LaTeX distribution, which is using `pdf(e)latex` as engine in a version $\geq$ v1.40.0[4]!
- While using XeLaTeX you have to make sure that page dimensions are written to the xdvi file, e. g. with the option `pagesize` of the KoMa-Script classes, or as option of the package `typearea.sty` respectively.

3 Commands

3.1 Comment commands

You can use the following commands for commenting your documents.

3.1.1 `\pdfcomment`

`\pdfcomment` `[`⟨*options*⟩`]`⟨*comment*⟩

Possible options: `avatar`, `style`, `subject`, `author`, `icon`, `color`, `opacity`, `open`, `hspace`, `voffset`, `hoffset`, `disable`, `date`, `timezone`, `id`

`\pdfcomment` will typeset an annotation into the text at the current  position. Internally, the argument ⟨*comment*⟩ needs to be converted to PDFDocEncoding/PDFUnicode[5]. Some chars with special meaning in LaTeX (`&`, `%`, ...) must be escaped or replaced with their command form, e. g. `\_` or `\textunderscore`. Furthermore there are some commands for formatting:

`\textHT`	horizontal tab
`\textLF`	line feed
`\textCR`	carriage return

3.1.2 `\pdfmargincomment`

`\pdfmargincomment` `[`⟨*options*⟩`]`⟨*comment*⟩

Possible options: `avatar`, `style`, `subject`, `author`, `icon`, `color`, `opacity`, `open`, `hspace`, `voffset`, `hoffset`, `disable`, `date`, `timezone`, `id`

`\pdfmargincomment` will typeset an annotation into the margin. Please note the positioning of the annotation in this example. This documentation was written with the LaTeX class `ltxdoc`. Therefore the annotation is typeset into the left margin.

[4] older versions are not capable of providing x/y coordinates of the current position in dvi mode.

[5] `\hypersetup`{unicode}

3.1.3 \pdfmarkupcomment

\pdfmarkupcomment[⟨*options*⟩] ⟨*markup text*⟩⟨*comment*⟩

Possible options: avatar, style, subject, author, color, opacity, markup, mathstyle, disable, date, timezone, id

\pdfmarkupcomment will typeset so called PDF text markup annotations over the text specified in the argument ⟨*markup text*⟩. Possible types for the option markup are: Highlight, Underline, Squiggly and StrikeOut
Note: The use of *markup* in the name of the command is a bit wrong and confusing as the PDF specification calls nearly all comments provided by this package "markup" annotations. The annotations here are more correctly named *text markup annotations*.

Limitations:

- As the name PDF text markup annotation might lead to suggest, this form of comment is suitable for text only.
- The specified text with the argument ⟨*markup text*⟩ must not be longer than one paragraph.
- The PDF text markup annotation must not be longer than one page, otherwise the recognition of the page break will fail.
- In text mode the command uses the package soul, this means that not every command works in the argument.

Since version v1.6, \pdfmarkupcomment is also supported in math mode. So you can also comment formula like:

$$\text{Bernoulli Trials: } P(E) = \binom{n}{k} p^k (1-p)^{n-k}$$

Also see example_math_markup.pdf for a more detailed explanation of the possibilities!

3.1.4 \pdffreetextcomment

\pdffreetextcomment [⟨*options*⟩]⟨*comment*⟩

Possible options: avatar, style, subject, author, color, font, fontsize, fontcolor, opacity, line, linewidth, lineend, borderstyle, dashstyle, bse, bsei, type, height, width, voffset, hoffset, disable, date, timezone, id

\pdffreetextcomment (type=freetext) will typeset a comment in form of a freely positioned box on the wished spot of the page. By using the option type=callout the box will have an additional so called callout line, to bind the box to a certain element of the page. With the option type=typewriter you can write a comment everywhere on the page with a virtual 'typewriter'. These PDF freetext annotation don't have a frame but a transparent background for 'overwriting' arbitrary parts of the page. Please note the example file example.pdf!

3.1.5 \pdfsquarecomment

\pdfsquarecomment [⟨options⟩]⟨comment⟩

Possible options: avatar, style, subject, author, color, icolor, opacity, linewidth, borderstyle, dashstyle, bse, bsei, height, width, voffset, hoffset, disable, date, timezone, id

\pdfsquarecomment will typeset a rectangular box at the wished spot of a page (see: example.pdf).

3.1.6 \pdfcirclecomment

\pdfcirclecomment [⟨options⟩]⟨comment⟩

Possible options: avatar, style, subject, author, color, icolor, opacity, linewidth, borderstyle, dashstyle, bse, bsei, height, width, voffset, hoffset, disable, date, timezone, id

\pdfsquarecomment will typeset a comment in form of a circle or ellipse at the wished spot of the page (see: example.pdf).

3.1.7 \pdflinecomment

\pdflinecomment [⟨options⟩]⟨comment⟩

Possible options: avatar, style, subject, author, type, color, icolor, opacity, caption, captionhoffset, captionvoffset, linewidth, linebegin, lineend, line, borderstyle, dashstyle, disable, date, timezone, id

\pdflinecomment will typeset a comment in form of a line, polyline or polygon at the wished spot of the page (see: example.pdf).

3.1.8 \pdfreply

\pdfreply [⟨options⟩]⟨comment⟩

Possible options: replyto, avatar, style, subject, author, type, color, icolor, opacity, caption, captionhoffset, captionvoffset, linewidth, linebegin, lineend, line, borderstyle, dashstyle, disable, date, timezone, id

\pdfreply will add a comment as a reply to an existing comment. It should use the option replyto to identify the target. As an example

```
\pdfcomment[id=A]{Wow, that's nice!}%
\pdfreply[replyto=A]{Yes true}
```

The annotation must be created on the same page as the annotation is refers too! **Note:** The command already existed in version 2.4a but was undocumented (and broken). With version 2.4a the command works only with pdflatex and lualatex, with version 3.0 the other backends are supported too.

3.2 Tooltips

3.2.1 \pdftooltip

\pdftooltip[⟨options⟩] ⟨object⟩⟨comment⟩

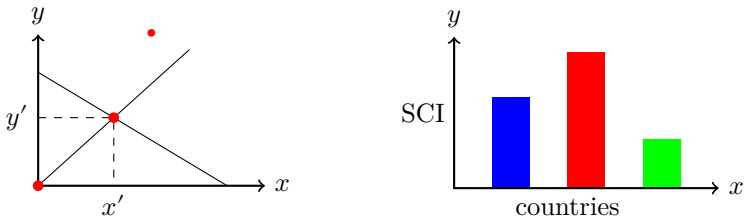
Possible options: disable, mathstyle

With the command \pdftooltip you can annotate any object with a tooltip - provided the the object can be set into a box to measure its size. Therefore you can smoothly use tooltips with words or in a chemical notation: H_2SO_4

Of course, it also works in equations:

$$\sum_{i=1}^{n} i = \frac{1}{2} n \cdot (n+1)$$

Please notice the comments about usage in math mode in section 3.1.3.
If you want to annotate parts of a graphic with tooltips, special methods are needed, because `\pdftooltip` can not measure the size (see: `example.pdf`).
`\pdftooltip` can not be influenced with the `final` option, as most users certainly want to have the tooltips in the final document. But they can be switched off with the `disable` option.



3.2.2 `\pdfsquaretooltip`

`\pdfsquaretooltip[`⟨*options*⟩`]`⟨*object*⟩⟨*comment*⟩

Tooltips are actually special Form fields and so not every PDF viewer can handle them. `\pdfsquaretooltip` are an alternative: this is a special invisible square annotation. If you prefer this variant, you can map it to the tool tip command with

```
\RenewCommandCopy\pdftooltip\pdfsquaretooltip
```

3.3 Misc. commands

3.3.1 `\pdfcommentsetup`

`\pdfcommentsetup [`⟨*options*⟩`]`⟨*options*⟩

With the command `\pdfcommentsetup` you can reset the global options at any time. By using the optional option `local` you can keep the reset local within a LaTeX group.
Version 3.0: With the new code used if PDF management is loaded, options are always set locally and the option `local` is ignored.

3.3.2 `\listofpdfcomments`

`\listofpdfcomments[`⟨*options*⟩`]`

The command `\listofpdfcomments` creates a list of the PDF comments to ease finding them.
Version 3.0: With the new code the command no longer forces one column mode.
The command offers the following options:

- `liststyle` specifies the list style. Possible option values are:
  - ⟨*AuthorSubject*⟩ (default)
  - ⟨*AuthorSubjectDeadline*⟩
  - ⟨*SubjectAuthorComment*⟩

- ⟨*SubjectAuthorCommentDeadline*⟩
- ⟨*AuthorComment*⟩
- ⟨*AuthorCommentDeadline*⟩
- ⟨*SubjectComment*⟩
- ⟨*SubjectCommentDeadline*⟩
- ⟨*Comment*⟩
- ⟨*CommentDeadline*⟩

Version 3.0: With the new code used if the PDF management is loaded the option can be also used in the optional argument of the various comment commands and then change the way they are added to list of pdf comments. An empty value suppresses the entry for this comment.

- `heading` specifies an alternative heading, e. g. `heading`={List of comments}.

3.3.3 \setliststyle

`\setliststyle`⟨*name*⟩ The command `\setliststyle` is necessary, if comments are used before `\listofpdfcomments` and you wish to have an alternative list style.

3.3.4 \definelistst yle

`\definelistst yle` ⟨*name*⟩⟨*definition*⟩ With the command `\definelistst yle` you can define new list styles, e. g.:

```
\makeatletter
\definelistst yle{MyListstyle}{\pc@lopt@subject\ \textcolor{green}
{(\pc@lopt@author)}:\ \pc@lopt@comment\ \textcolor{red}
{(\pc@lopt@deadline)}}
\makeatother
```

3.3.5 \defineavatar

`\defineavatar`⟨*name*⟩⟨*options*⟩ With the command `\defineavatar` you can create named predefined option lists, which can be later used in the comment commands with the option `avatar`. With this option, it's easy for several authors of the same document to switch between different avatars, that is their graphical representation.

3.3.6 \definestyle

`\definestyle`⟨*name*⟩⟨*options*⟩ With the command `\definestyle` you can split up lists in personal and stylistic option lists (see `example.pdf`). This option list can be loaded with the option `style`.

4 Printing comments and popups

Basically, `Adobe Reader` is able to print PDF annotations. Therefore, you have to choose 'Document and Markups' in the field 'Comments and Forms' of the print dialog. Additionally, you have to configure the menu 'Edit → Preferences... → Commenting → Print notes and pop-ups'. This entry exists since the release of `Adobe Reader X`.
Please note that only opened popups will be printed!

5 Tagging support

Tagging is enabled with `\DocumentMetadata{tagging=on}`. It is not clear yet, which is the best tagging and what it actually does for the reading. Please report problems! Improvements need tests and feedbacks from user!
Known restrictions are:

- `\pdfmarkupcomment` uses the soul package to get the dimension of the annotation. The soul package is not compatible with the tagging, and so tagging is suspended in the argument of the command.
- Inside a tikz-picture tagging is normally suspended (if you use the `alt` or `actualtext` key), and so also annotations are not tagged too. This is allowed by the standard (untagged annotations are artifacts) but some validator complain nevertheless.
- Markup annotations going over more than one line create more than one annation. These should probably be in one Annot structure, but currently everyone is tagged independantly.
- In PDF 2.0 annotations should actually have appearances, but this isn't done yet.