

The `attachfile` package*

Scott Pakin
`scott+af@pakin.org`

November 11, 2015

Abstract

This package defines an `\attachfile` command that lets you attach arbitrary files to a PDF document. These files are embedded right in the PDF file, so they get transmitted along with it. The package also gives you control over the corresponding icon’s properties and various other associated metadata.

1 Introduction



PDF, Adobe’s Portable Document Format, is a common way to distribute documents that look the same on all platforms and output devices. Beginning with PDF version 1.3, PDF supports “file attachment annotations”. These are arbitrary auxiliary files that get embedded directly into the PDF document, just like attachments in an e-mail message.

The `attachfile` package gives pdfL^AT_EX users the ability to add these attachments to their documents automatically. And because L^AT_EX is a markup language, not a WYSIWYG tool, the user has precise control over the location of the file attachment icons. If an icon representing an attached spreadsheet file is placed next to a figure, the icon will move along with the figure whenever the document is modified. Furthermore, it is possible to define global properties for all the file attachments in a document. With one command, a user can change the properties of all the icons in the entire document.

Finally, one nifty feature that `attachfile` supports is the ability to use your own icons, which can be text, graphics, tables, mathematics—you name it! With this feature, a PDF file can, for example, instruct the reader to click on a formula to extract the Mathematica notebook that derived it. Or to click on a graph to extract the Microsoft Excel spreadsheet that contains all the data that was plotted. The possibilities are endless.

Okay, let’s get down to business. Here are some sample file attachments so you can see if your PDF viewer is able to handle them:

*This file has version number v1.8, last revised 2015/11/11.

Icon:  (Should resemble this: )
L^AT_EX text: `attachfile.bib` (Should resemble this: `attachfile.bib`)

Each of the above points to the B_IB_T_EX bibliography (a plain text file) for the document you’re reading now. Try extracting the attachment. In Adobe Acrobat, this is achieved by right-clicking on the icon and choosing “Save Embedded File to Disk. . .” (or in older versions of Adobe Acrobat, “Extract File. . .”). You can also double-click to open the file immediately. If you’re unable to access the attached file, or you observe miscellaneous strange behavior, your PDF viewer might not be capable of handling file attachments properly. See Section 5 for some PDF viewer problems I encountered while testing `attachfile`.

2 Usage

Load `attachfile` by putting a `\usepackage{attachfile}` in your document’s preamble. `attachfile` implicitly loads a variety of other packages. Section 5 presents the complete list.

`attachfile` v1.8 does not have any of its own package options; any options that get passed to `attachfile` are forwarded to `hyperref`. Because `hyperref` works best when loaded as one of the last packages in the document the same holds true for `attachfile`.

3 Commands

The following are the commands that `attachfile` makes available for attaching files, customizing the icon appearance, and changing various file attachment metadata.

`\attachfile` [*options*] {*filename*}

The `\attachfile` macro has only one required argument: the name of the file to attach. `\attachfile` will insert an icon at the current point in the document to represent the attachment. *options* is a list of optional parameters for describing the icon and other assorted metadata. It is described in Section 4.

`\noattachfile` [*options*]

When writing instructions, it is sometimes convenient to describe what a file attachment icon looks like without actually attaching a file. That’s what `\noattachfile` is for. All it does is insert the image of a file attachment icon into the document. *options* is a list of optional parameters for describing the icon and other assorted metadata. It is described in Section 4. In particular, note that if the `print` option is set to `false` then `\noattachfile` will output empty space of the same size as the icon image.

`\textattachfile [options] {filename} {text}`

`\textattachfile` is just like `\attachfile`, except that instead of using one of the predefined PDF icons, it lets you use an arbitrary piece of text to represent the attachment. The *text* parameter is not limited to text; it can contain any arbitrary horizontal material. The following are all legal uses of `\textattachfile`:

- You can `\textattachfile{myfile.cc}{extract my source code}` if your PDF viewer supports file annotations.
- It is intuitively obvious to even the most casual observer that

```
\textattachfile{derivation.m}{
$$\frac{\partial E_p}{\partial w_{ji}^h} = -\sum_k (y_{pk} - o_{pk}) f_k^{(o)'}(\text{net}_{pk}^o) w_{kj}^o f_j^{(h)'}(\text{net}_{pj}^h) x_{pi}$$
}
```

- `\textattachfile{earningsdata.csv}{\includegraphics{earnings}}`

`\notextattachfile [options] {text}`

Just as `\noattachfile` is a dummy version of `\attachfile`, so `\notextattachfile` is a dummy version of `\textattachfile`. All `\notextattachfile` does is insert *text* into the document according to *options* (described in Section 4). In particular, note that if the `print` option is set to `false` then `\notextattachfile` will output empty space of the same size as *text*.

`\attachfilesetup {options}`

If you find yourself passing the same set of options to multiple `\attachfile` calls in your document, you can use `\attachfilesetup` to specify default option values. `\attachfilesetup`'s *options* parameter is the same as that used by `\attachfile` and is described in Section 4. Some noteworthy points are:

1. `\attachfilesetup` can be called as many times as desired. Any options specified replace the previous value of those options. All unspecified options are left alone.
2. Options passed to `\attachfile` take precedence over those specified by `\attachfilesetup`. This lets you define default values for all file attachments and selectively override them on a per-attachment basis.
3. Options set by `\attachfilesetup` are local to the current scope. This lets you assign defaults to a group of file attachments without affecting the global defaults. To define options that apply to the entire document, `\attachfilesetup` should be called at the top-level scope (which includes the document's prologue).

4 Options

`attachfile` gives the user a great deal of control over the way files are attached to a document. All the commands in Section 3 accept the same set of options, which are entered as comma-separated, $\langle key \rangle = \langle value \rangle$ pairs. Options can be specified in any order. Case is significant. Only the options you want to change need to be specified; the others will retain their previous value (or the default, if no previous value was specified).

4.1 List of available options

The following are the options `attachfile` accepts, in alphabetical order.

`appearance= $\langle boolean \rangle$`

The `attachfile` package normally embeds the file attachment’s icon explicitly with each file attachment annotation. (In PDF-speak, it includes an appearance dictionary in the `FileAttachment` object.) The advantages to doing this are to ensure that:

- The file attachment icons look the same in all PDF viewers.
- T_EX knows exactly how much space to allocate, instead of just guessing based on the size of the Adobe Acrobat icons.
- Pre-1.3 PDF viewers don’t regress to showing an “unknown annotation type” graphic.

However, the problems with embedding the icon graphic are:

- It adds a bit of extra bulk to the PDF file.
- It takes flexibility away from the PDF viewer, which can no longer choose for itself how best to render a file attachment icon.

The `appearance` option gives the author the ability to prevent the icon’s appearance from being specified explicitly in the PDF file. By setting `appearance=false`, it will be left up to the PDF viewer to decide how to display the icon.

`author= $\langle text \rangle$`

The metadata associated with a file attachment annotation includes the name of the person who attached the file. In Adobe Acrobat, this information is shown when one right-clicks on the file attachment icon and selects *Properties...* By default, no author is listed but specifying `author= $\langle name \rangle$` sets the author field to $\langle name \rangle$.

`color=<red> <green> <blue>`

The icons inserted by `\attachfile` and the text inserted by `\textattachfile` can be any color. The `color` option sets this color. Each of `<red>`, `<green>`, and `<blue>` must be a decimal number between 0 (darkest) and 1 (brightest). The default is `color=1 0.9255 0.7765`, which is a beige.

`created=<PDF date>`

Virtually all filesystems associate a file-creation timestamp with each file. Although \TeX provides no portable mechanism for determining the date and time a file was created the `created` option lets you manually specify these parameters for the reader's benefit. See Section 4.2 for more information about `attachfile` dates.

`date=<PDF date>`

Each annotation in a PDF file can have a timestamp indicating when the annotation was last modified. `attachfile` automatically adds a timestamp to file attachment annotations. It uses the date and time at which \LaTeX started processing your job (to minute precision because that's what \TeX 's `\time` command provides) and includes the timezone, if specified (using the `timezone` option, p. 7). Although it's unlikely you'll need to use it, the `date` option lets you override the annotation's modification date and time with a date and time of your choice. See Section 4.2 for more information about `attachfile` dates.

`description=<text>`

The metadata associated with a file attachment annotation can include a brief description of the file. In Adobe Acrobat, this information is shown when one right-clicks on the file attachment icon and selects *Properties...* Also, in later versions of Adobe Acrobat, the description field shows up as a tool tip when the user mouses over the attachment. By default, no description is included, but specifying `description=<text>` sets the description field to `<text>`.

`icon=<name>`

PDF 1.3 defines four icons that can be used for file attachments: `Graph`, `Paperclip`, `PushPin`, and `Tag`. These are shown in Table 1. If no icon name is specified, `PushPin` is assumed. While the PDF specifications say that, normally, a PDF viewer chooses how to display each of those, the `attachfile` package specifies the appearance explicitly. This is what Adobe Acrobat does, presumably because doing so ensures that viewers which don't support file attachment annotations can still display something reasonable. The tradeoff is that it slightly increases the size of the PDF file.





Graph	
Paperclip	
PushPin	
Tag	

Table 1: Valid file attachment icons

`mimetype=<type>`

It is considered good practice to specify the MIME type [2] of each attached file. That way, a PDF viewer can automatically launch an appropriate application to process the file. *<type>* should be the form “*<type>/<subtype>*”. For instance, a plain text file would be specified with “`mimetype=text/plain`”. An MPEG movie would be specified with “`mimetype=video/mpeg`”. The Internet Assigned Numbers Authority maintains a list of registered media types [3], so look there first to see what type to use for a given file.

`modified=<PDF date>`

Virtually all filesystems associate a last-modification timestamp with each file. Although \TeX provides no portable mechanism for determining the date and time a file was last modified the `modified` option lets you manually specify these parameters for the reader’s benefit. See Section 4.2 for more information about `attachfile` dates.

`print=<boolean>`

By default, file annotation icons print along with the rest of the document. By setting `print=false`, the icons will not print. Note that in Adobe Acrobat, annotations will *never* print unless the Annotations box is checked in the Print dialog.

`size=<integer>`

The `size` option tells the PDF viewer that the attached file is *<integer>* bytes long. Adobe Acrobat displays this size under the “Size” column in the Attachments pane but does not otherwise seem to use the *<integer>* value.

`subject=<text>`

The metadata associated with a file attachment annotation can include a brief comment about the subject of the attachment. In Adobe Acrobat, this information is shown when one right-clicks on the file attachment icon and selects *Properties*. By default, no subject is included, but specifying `subject=<text>` sets the subject field to `<text>`.

`timezone=<offset>`

Because \TeX doesn't make the current timezone available, `attachfile` is unable to include timezone information when it timestamps a file attachment. The `timezone` option lets you manually specify the timezone. `<offset>` is the offset from Universal Time (a.k.a. GMT) and should be in the format specified in the PDF reference manual [1, §3.8.3, "Dates"], namely:

`+<HH>'<mm>'` `<HH>` hours, `<mm>` minutes later than Universal Time
(i.e., east of Greenwich, England)

`-<HH>'<mm>'` `<HH>` hours, `<mm>` minutes earlier than Universal
Time (i.e., west of Greenwich, England)

`Z` Universal Time (i.e., at the same longitude as Green-
wich, England)

For example, U.S. Central Time would be specified with `timezone=-06'00'`.

`zoom=<boolean>`

Normally, when a reader magnifies or reduces the view of the PDF document, the file annotation icons change size proportionally with the text. By setting `zoom=false`, the icon size does not scale.

The defaults for all of the options described above are summarized in Table 2.

4.2 Date usage

Section 4.1 presents three timestamp-related options: `date`, `created`, and `modified`. The `date` option specifies the annotation date—the date and time the given file was attached to the PDF file—and should usually be left unspecified. (It defaults to the date and time at which \LaTeX started processing your job.) The annotation date is displayed in Adobe Acrobat by right-clicking on the annotation, choosing *Properties...* from the menu, and clicking on the *General* tab. The `modified` option specifies the file's modification date—the date and time the given file was last modified. Adobe Acrobat displays the modification date under the "Modified" column in the Attachments pane but does not otherwise appear to

Option	Default setting
<code>appearance</code>	<code>true</code>
<code>author</code>	<code>none</code>
<code>color</code>	<code>1 0.9255 0.7765</code>
<code>created</code>	<code>none</code>
<code>date</code>	<code>automatic</code>
<code>description</code>	<code>none</code>
<code>icon</code>	<code>PushPin</code>
<code>mimetype</code>	<code>none</code>
<code>modified</code>	<code>none</code>
<code>print</code>	<code>true</code>
<code>size</code>	<code>none</code>
<code>subject</code>	<code>none</code>
<code>timezone</code>	<code>none</code>
<code>zoom</code>	<code>true</code>

Table 2: Default values for all options

use the modification date. Finally, the `created` option specifies the file’s creation date—the date and time the given file was first written to disk. As of this writing, Adobe Acrobat does not appear to use or even display the creation date; perhaps future versions or other PDF viewers will.

Dates should be specified in the form “D: YYYYMMDDHHmmSSOHH’mm’” as described in the PDF reference manual [1, §3.8.3, “Dates”]. Note, however, that although the PDF reference manual clearly states that “viewer applications should be prepared to accept and display a string in any format” [1, Table 8.11, “Entries common to all annotation dictionaries”], Adobe Acrobat will ignore any timestamp that is not in the recommended format and will instead show “00/00/00 00:00:00” for the annotation date or “Unknown” for the modification date.

5 Caveats

Note that there are a few caveats you should be aware of:

1. `attachfile` requires either `pdfLATEX` version 0.14 or later or `LuaLATEX`. (Version 0.14 of `pdfLATEX` was released circa 1999 so it’s unlikely that you’re running an older version than that.) While there are many other ways to produce PDF files from `LATEX` source, `attachfile` v1.8 supports only `pdfLATEX` and `LuaLATEX`.
2. `attachfile` will not run unless the following `LATEX` packages are installed: `calc`, `keyval`, `color`, `hyperref`, and `ifpdf`. (Most `TEX` distributions include all of these.)
3. File attachments are a PDF 1.3 feature. They will not be visible in PDF viewers that don’t support PDF 1.3. (Version 4.0 of Adobe Acrobat is the

first version of that program which does.)

4. Even some viewers that purportedly support PDF 1.3 don't support file attachments. As far as I can tell, very old versions of Adobe Acrobat Reader (the free, view-only version of Adobe Acrobat) doesn't seem to support *any* annotations except text annotations.
5. Even some viewers that do support PDF 1.3 and file attachments don't support them under all circumstances. For instance, some Windows versions of Adobe Acrobat, when functioning as a Web-browser plug-in, give an error message¹ when a file attachment icon is activated.
6. Even in circumstances where file attachments are supported, the support may be flawed. For example, some Windows versions of Adobe Acrobat change a custom icon to the default icon when it's selected.

Also, while file-attachment icons with custom appearances printed fine in older versions of Adobe Acrobat, Adobe introduced a bug circa Adobe Acrobat 6.0 that prevents `attachfile`'s icons from printing. Unfortunately, because Adobe Acrobat lacks `attachfile`'s ability to create custom appearances for file-attachment icons it's unlikely that this bug will ever get fixed. Nevertheless, please consider sending a bug report to Adobe to let them know that you'd like to be able to print file-attachment icons with custom appearances.

Even given all of those caveats, file attachments can be a useful way to pass additional information along with a PDF file. The `attachfile` package makes file annotations automatic and easy.

6 Implementation

This section contains the complete source code for `attachfile`. Most users will not get much out of it, but it should be of use to those who need more precise documentation and those who want to extend the `attachfile` package.

6.1 Sanity checking

`attachfile` v1.8 requires either `Lua \LaTeX` or `pdf \LaTeX` (and at least version 0.14 of `pdf \LaTeX` , although `attachfile` no longer checks for that). (Future versions of `attachfile` may support `dvipdfm`, `dvips` with `pdfmarks`, `V \TeX` , etc.) Also, `pdf \LaTeX` /`Lua \LaTeX` must be in PDF-generating mode, not DVI-generating mode. So, to save the user some aggravation, we check for the correct backend right up front and give a warning if all is not well. Later, in Section 6.7, we replace all of the core `attachfile` macros with dummy versions so `\LaTeX` can at least run to completion.

```
1 \RequirePackage{ifpdf}
2 \ifpdf
3 \else
```

¹“Launching embedded files from within a browser environment is not allowed”.

```

4 \PackageWarningNoLine{attachfile}{%
5   attachfile works _only_ with pdfLaTeX and LuaLaTeX\MessageBreak
6   and _only_ in PDF-generating mode. For this run,\MessageBreak
7   placeholders will be substituted for all\MessageBreak
8   attachfile commands%
9 }
10 \fi

```

6.2 Preliminaries

We need to load `hyperref` to get our hands on that great `\pdfstringdef` macro. For now, we blindly pass all our package options directly to `hyperref`. In the future, it would be nice to do a `\setkeys{AtFi}` on our options.

```

11 \RequirePackage{keyval}
12 \RequirePackage{calc}
13 \RequirePackage{color}
14 \RequirePackageWithOptions{hyperref}

```

6.3 Adobe Acrobat icons

The following macros draw a representation of the various icons that Adobe Acrobat² inserts to represent what the PDF 1.3 specifications refer to as “Graph,” “Paperclip,” “PushPin,” and “Tag”. The `\parbox` dimensions are taken directly from the original graphics’ bounding box. However, I just eyeballed the `\raisebox` heights (intended to put shadows below the baseline).

`\atfi@acroGraph@data` Recreate Adobe Acrobat’s Graph icon.

```

15 \newcommand{\atfi@acroGraph@data}{%
16   q 0.5 g 1.1133 0 20.7202 18.2754 re f 1 g 0 G 0 i 0.5 w 4 M
17   0.25 1.6453 20.145 17.7715 re B 0 g 2.7319 4.1367 3.9571
18   13.8867 re f 8.7031 4.1367 3.9571 9.8867 re f 14.7471 4.1367
19   3.9571 11.8867 re f \atfi@color@rgb\space rg 1.689 3.0938
20   3.9571 13.8867 re f 7.6602 3.0938 3.9571 9.8867 re f 13.7041
21   3.0938 3.9571 11.8867 re f Q
22 }

```

`\atfi@acroGraph` Draw `\atfi@acroGraph@data` in a box of the appropriate size.

```

23 \DeclareRobustCommand{\atfi@acroGraph}{%
24   \raisebox{-1.5bp}{\parbox[b][20bp]{22bp}{%
25     \rule{0pt}{0pt}\pdfliteral{\atfi@acroGraph@data}}}%
26   }%
27 }

```

`\atfi@acroPaperclip@data` Recreate Adobe Acrobat’s Paperclip icon.

```

28 \newcommand{\atfi@acroPaperclip@data}{%
29   q 0.75 G 0 i 2.5 w 1 J 4 M 1.9619 11.7559 m 1.9619 3.3037

```

²I got these graphics specifically from the Windows version of Adobe Acrobat 4.0.

```

30 1.9619 2.5059 v 1.9619 1.707 4.0947 1.25 y 7.4141 1.25 l 9.4292
31 1.8223 9.4292 3.3066 v 9.4292 4.79 9.4292 16.8945 y 9.7852
32 18.1514 8.481 18.1514 v 7.1768 18.1514 5.1616 18.1514 y 3.8574
33 17.9209 3.8574 16.8945 v 3.8574 15.8652 3.8574 6.6172 y 4.3325
34 5.418 5.1025 5.418 v 5.8726 5.418 6.5845 5.418 y 7.6812 5.6455
35 7.6812 6.4736 v 7.6812 7.3027 7.6812 11.5264 y S 0 G 1.2495
36 12.4404 m 1.2495 3.9883 1.2495 3.1895 v 1.2495 2.3906 3.3833
37 1.9326 y 6.7026 1.9326 l 8.7178 2.5068 8.7178 3.9902 v 8.7178
38 5.4736 8.7178 17.5781 y 9.0732 18.834 7.769 18.834 v 6.4653
39 18.834 4.4497 18.834 y 3.146 18.6055 3.146 17.5781 v 3.146
40 16.5498 3.146 7.3018 y 3.6201 6.1016 4.3911 6.1016 v 5.1611
41 6.1016 5.873 6.1016 y 6.9692 6.3301 6.9692 7.1572 v 6.9692
42 7.9863 6.9692 12.21 y S \atfi@color@rgb\space RG 1 w
43 1.2495 12.4404 m 1.2495 3.9883 1.2495 3.1895 v 1.2495 2.3906
44 3.3833 1.9326 y 6.7026 1.9326 l 8.7178 2.5068 8.7178 3.9902 v
45 8.7178 5.4736 8.7178 17.5781 y 9.0732 18.834 7.769 18.834 v
46 6.4653 18.834 4.4497 18.834 y 3.146 18.6055 3.146 17.5781 v
47 3.146 16.5498 3.146 7.3018 y 3.6201 6.1016 4.3911 6.1016 v
48 5.1611 6.1016 5.873 6.1016 y 6.9692 6.3301 6.9692 7.1572 v
49 6.9692 7.9863 6.9692 12.21 y S Q
50 }

```

`\atfi@acroPaperclip` Draw `\atfi@acroPaperclip@data` in a box of the appropriate size.

```

51 \DeclareRobustCommand{\atfi@acroPaperclip}{%
52 \raisebox{-1.25bp}{\parbox[b][21bp]{12bp}{%
53 \rule{0pt}{0pt}\pdfliteral{\atfi@acroPaperclip@data}}%
54 }%
55 }

```

`\atfi@acroPushPin@data` Recreate Adobe Acrobat's PushPin icon.

```

56 \newcommand{\atfi@acroPushPin@data}{%
57 q \atfi@color@rgb\space rg 0 G 1 w 1 6 m 11 6 l 11 13 l 12
58 13 l 14 11 l 21 11 l 22 12 l 23 12 l 23 2 l 22 2 l 21 3 l 14 3
59 1 12 l 1 11 l 1 11 6 l B 0.5 G 0 7 m 10 7 l 10 8 l 1 8 l S 1 G
60 12 12 m 14 10 l 22 10 l 22 11 l S Q
61 }

```

`\atfi@acroPushPin` Draw `\atfi@acroPushPin@data` in a box of the appropriate size.

```

62 \DeclareRobustCommand{\atfi@acroPushPin}{%
63 \raisebox{-1.25bp}{\parbox[b][14bp]{24bp}{%
64 \rule{0pt}{0pt}\pdfliteral{\atfi@acroPushPin@data}}%
65 }%
66 }

```

`\atfi@acroTag@data` Recreate Adobe Acrobat's Tag icon.

```

67 \newcommand{\atfi@acroTag@data}{%
68 q 0.5 g 10.0542 14.9873 m 24.27 14.9873 l 25.252 14.0059 l
69 25.252 1.1455 l 24.1064 0 l 9.9609 0 l 6.0327 6.0088 l 6.0327
70 9.002 l 10.0542 14.9873 l 9.3994 9.376 m 8.5215 9.376 7.8096

```

```

71 8.5596 7.8096 7.5527 c 7.8096 6.5449 8.5215 5.7285 9.3994
72 5.7285 c 10.2778 5.7285 10.9897 6.5449 10.9897 7.5527 c 10.9897
73 8.5596 10.2778 9.376 9.3994 9.376 c h f
74 \atfi@color@rgb\space rg 0 G 0 i 0.5 w 4 M 1 j 8.5107
75 16.5313 m 22.7266 16.5313 l 23.7085 15.5488 l 23.7085 2.6895 l
76 22.563 1.543 l 8.4175 1.543 l 4.4893 7.5527 l 4.4893 10.5449 l
77 8.5107 16.5313 l 7.856 10.9199 m 6.978 10.9199 6.2661 10.1035
78 6.2661 9.0957 c 6.2661 8.0879 6.978 7.2715 7.856 7.2715 c
79 8.7344 7.2715 9.4463 8.0879 9.4463 9.0957 c 9.4463 10.1035
80 8.7344 10.9199 7.856 10.9199 c h B 1 w 12.3291 12.2656 m
81 21.1206 12.2656 l S 12.3291 9.1797 m 21.1206 9.1797 l S 12.3291
82 6.1875 m 21.1206 6.1875 l S 0 G 0.5 w 0 9.0488 m 6.2661 9.0957
83 l S 1.4028 5.2148 m 1.4028 9.6094 l 1.6831 10.6387 2.4316
84 10.6387 v 3.6475 10.6387 3.5542 9.0488 y S Q
85 }

```

`\atfi@acroTag` Draw `\atfi@acroTag@data` in a box of the appropriate size.

```

86 \DeclareRobustCommand{\atfi@acroTag}{%
87   \raisebox{-1.6bp}{\parbox[b][17bp]{25bp}{%
88     \rule{0pt}{0pt}\pdfliteral{\atfi@acroTag@data}}}%
89   }%
90 }

```

6.4 Helper routines

`\atfi@temp@string` This is the same as `\pdfstringdef`, except that it *locally* defines its argument. For those of you who like analogies, `\atfi@pdfstringdef` is to `\def` as `\pdfstringdef` is to `\gdef`.

```

91 \def\atfi@temp@string{}
92 \DeclareRobustCommand{\atfi@pdfstringdef}[2]{%
93   \pdfstringdef\atfi@temp@string{#2}%
94   \edef#1{\atfi@temp@string}%
95 }

```

`\c@atfi@tmp` Because \TeX provides only a limited number of counters, we recycle a single counter, `atfi@tmp`, throughout the entire package whenever the need to perform arithmetic arises.

`\theatfi@tmp`

```

96 \newcounter{atfi@tmp}
97 \renewcommand*{\theatfi@tmp}{\the\value{atfi@tmp}}

```

`\atfi@embedfile` If the given file has not yet been embedded, embed it as a PDF EmbeddedFile object, and store its object number in `\atfi@embedfile@{filename}`.

```

98 \DeclareRobustCommand{\atfi@embedfile}[1]{%
99   \expandafter\ifx\csname atfi@embed@file@#1\endcsname\relax
100   \immediate\pdfobj stream attr {
101     /Type /EmbeddedFile
102     \atfi@mimetype\space

```

```

103     \atfi@dysize\space
104     /Params <<
105     \atfi@credate\space
106     \atfi@moddate\space
107     \atfi@size\space
108     >>
109   } file {#1}%
110   \expandafter\xdef\csname atfi@embed@file@#1\endcsname{\the\pdfastobj}%
111   \fi
112 }

```

`\atfi@appearancewidth` Each PDF annotation can have an associated “appearance”. In the `attachfile` package, we store the appearance with the `\atfi@set@appearance` macro (below).
`\atfi@appearanceheight` As a side effect, `\atfi@set@appearance` stores the dimensions of its argument in
`\atfi@appearancedepth` `\atfi@appearancewidth`, `\atfi@appearanceheight`, and `\atfi@appearancedepth`
`\atfi@appearancebox` so that, later, we can allocate an appropriate amount of space for the file attachment icon to fit within. `\atfi@appearancebox` is a temporary storage location for the \TeX box that will get converted to an XObject.

```

113 \newlength{\atfi@appearancewidth}
114 \newlength{\atfi@appearanceheight}
115 \newlength{\atfi@appearancedepth}
116 \newsavebox{\atfi@appearancebox}

```

`\atfi@set@appearance` `\atfi@set@appearance` stores its argument as a PDF XObject for later referral by
`\atfi@appearance@obj` the file annotation’s appearance dictionary. This serves two purposes:

1. It enables a \TeX box with arbitrary contents to serve as the file attachment icon.
2. It enables (generally older) PDF viewers that don’t recognize the icon name to still display something meaningful.

```

117 \DeclareRobustCommand{\atfi@set@appearance}[1]{%
118   \savebox{\atfi@appearancebox}{#1}%
119   \settoheight{\atfi@appearanceheight}{\usebox{\atfi@appearancebox}}%
120   \settowidth{\atfi@appearancewidth}{\usebox{\atfi@appearancebox}}%
121   \settodepth{\atfi@appearancedepth}{\usebox{\atfi@appearancebox}}%
122   \immediate\pdfxform attr {
123     /Subtype /Form
124   } \atfi@appearancebox
125   \edef\atfi@appearanceobj{\the\pdfastxform}%
126 }

```

`\atfi@flags@to@int` Convert all our flag options from booleans into a single integer (`\atfi@flags`).

```

\atfi@flags 127 \DeclareRobustCommand{\atfi@flags@to@int}{%
128   \setcounter{atfi@tmp}{0}%
129   \ifatfi@print
130     \addtocounter{atfi@tmp}{4}%
131   \fi

```

```

132 \ifatfi@zoom
133 \else
134   \addtocounter{atfi@tmp}{8}%
135 \fi
136 \edef\atfi@flags{\theatfi@tmp}%
137 }

```

`\atfi@insert@file@annot` Insert a PDF FileAttachment annotation that refers to the object created by `\atfi@embedfile`. TeX doesn't normally "see" a `\pdfannot`, so we have to explicitly allocate space for it. `\atfi@insert@file@annot` takes one argument, the name of the file to attach. This should be the same value that was passed to `\atfi@embedfile`.

```

138 \DeclareRobustCommand{\atfi@insert@file@annot}[1]{%
139   \rule{Opt}{Opt}%
140   \bgroup\Hy@unicodedefalse
141     \atfi@pdfstringdef\atfi@file{#1}%
142     \edef\next{\egroup
143       \def\noexpand\atfi@file{\atfi@file}%
144     }\next
145   \filename@parse{\atfi@file}%
146   \edef\atfi@file{\filename@base.\filename@ext}%
147   \ifatfi@appearance

```

We currently use the same appearance for Normal, Rollover, and Down, although future versions of `attachfile` may provide support for different appearances. Although the PDF specification claims that R and D appearances default to the N appearance, experience dictates otherwise. Hence, we explicitly specify all three appearances.

```

148   \def\atfi@appearance@dict{%
149     /AP <<
150       /N \atfi@appearanceobj\space 0 R
151       /R \atfi@appearanceobj\space 0 R
152       /D \atfi@appearanceobj\space 0 R
153     >>%
154   }%
155 \fi%
156 \pdfannot width \atfi@appearancewidth
157           height \atfi@appearanceheight
158           depth \atfi@appearancedepth {
159   /Subtype /FileAttachment
160   \atfi@appearance@dict\space
161   \atfi@author\space
162   \atfi@color\space
163   \atfi@date\space
164   \atfi@description\space
165   \atfi@icon\space
166   \atfi@moddate\space
167   \atfi@subject\space
168   /F \atfi@flags\space

```

```

169 /FS <<
170 /Type /Filespec
171 /F (\atfi@file)
172 /EF <<
173 /F \csmame atfi@embed@file@#1\endcsmame\space 0 R
174 >>
175 >>
176 }%

```

Now, so \TeX can budget space for the annotation, we insert some zero-width rules into the document.

```

177 \rule{0pt}{\atfi@appearanceheight}%
178 \rule[-\atfi@appearancecedepth]{0pt}{\atfi@appearancecedepth}%
179 \rule{\atfi@appearancewidth}{0pt}%
180 }

```

`\atfi@attachfile` This macro does all the work of the `\attachfile` author command. `\attachfile` began a group in which most special characters are set to category code “other”. `\atfi@attachfile` reads the filename within this group, embeds the corresponding file into the generated PDF file, and places an icon at the current location. Then, it ends the group, thereby restoring the original category codes.

```

181 \def\atfi@attachfile#1#2{%
182   \setkeys{AtFi}{#1}%
183   \atfi@embedfile{#2}%
184   \atfi@set@appearance{\csmame atfi@acro\atfi@icon@icon\endcsmame}%
185   \atfi@flags@to@int%
186   \atfi@insert@file@annot{#2}%
187   \endgroup
188 }

```

`\atfi@textattachfile` All this macro does is evaluate its second argument (a filename) within the group begun by `\textattachfile` then pass control to `\atfi@textattachfile@i`, which does all the work. `\atfi@textattachfile` is needed to force the filename to be evaluated while special characters are set to use category code “other”.

```

189 \def\atfi@textattachfile#1#2{%
190   \endgroup
191   \atfi@textattachfile@i{#1}{#2}%
192 }

```

`\atfi@textattachfile@i` This macro does all the work of the `\textattachfile` author command. Given a filename, some arbitrary text, and an optional set of attachment options, embed the corresponding file into the generated PDF file, and use the text as the icon. We recycle the icon color for the text. Note that the `\strut` is a bug workaround; I don’t know whose fault this is, but the bottom point or so of the text seems to get cut off. Weird.

`\atfi@textcolor`

```

193 \def\atfi@textattachfile@i#1#2#3{%
194   \setkeys{AtFi}{#1}%
195   \atfi@embedfile{#2}%
196   \def\atfi@textcolor{##1 ##2 ##3}##4{%

```

```

197     \textcolor[rgb]{##1,##2,##3}{##4}%
198   \atfi@set@appearance{%
199     \expandafter\atfi@textcolor\expandafter
200     (\atfi@color@rgb){#3\strut}}%
201   \atfi@flags@to@int
202   \atfi@insert@file@annot{#2}%
203 \endgroup
204 }

```

`\atfi@pdf@slash` The PDF specification dictates that MIME types be specified not as strings (e.g., “(Hello)”) but rather as PDF names (e.g., “/Hello”). The catch is that the forward slash—required in all MIME types—cannot be part of a PDF name. The solution is to replace the MIME “/” with the hexadecimal sequence “#2f”. Unfortunately, pdfL^AT_EX replaces “#” with “##” in a `\pdfobj` but leaves “\#” as is. The solution is to play some games with T_EX to define `\atfi@pdf@slash` as a “#2f” sequence that can used within `\pdfobj`.

```

205 \bgroup
206   \lccode'\@=' \#
207   \lowercase{\gdef\atfi@pdf@slash{@2f}}
208 \egroup

```

`\atfi@split@mimetype` Split a MIME type (e.g., “image/jpeg”) into a type, `\atfi@mime@type` (e.g., “image”),
`\atfi@mime@type` and a subtype, `\atfi@mime@subtype` (e.g., “jpeg”).

```

\atfi@mime@subtype 209 \def\atfi@split@mimetype#1/#2/{%
210   \def\atfi@mime@type{#1}%
211   \def\atfi@mime@subtype{#2}%
212 }

```

6.5 Annotation option processing

We start by defining the various options that `\attachfile` accepts and their default values.

`\atfi@mimetype` Declare the MIME type of the attached file. For example, “text/plain” would specify that the attachment is an ordinary text file.

```

213 \def\atfi@mimetype{}
214 \define@key{AtFi}{mimetype}{%
215   \atfi@pdfstringdef\atfi@mimetype{#1}%
216   \atfi@split@mimetype#1/%
217   \edef\atfi@mimetype{%
218     /Subtype /\atfi@mime@type\atfi@pdf@slash\atfi@mime@subtype
219   }%
220 }

```

`\atfi@icon` Specify an icon to represent the attachment. This should be one of `Graph`, `Paperclip`, `PushPin` (the default), or `Tag`. `\atfi@icon` is an attribute/value pair that gets inserted directly into the file attachment object. `\atfi@icon@icon` is only

the icon name itself and is used to insert a static graphic that represents Adobe Acrobat's rendition of a file attachment icon.

```
221 \define@key{AtFi}{icon}{%
222   \def\atfi@icon{/Name /#1}%
223   \def\atfi@icon@icon{#1}%
224 }
225 \setkeys{AtFi}{icon=PushPin}
```

`\atfi@color` Specify the color of the attachment icon as an RGB triplet. For example, “0 0.3 0” would be a fairly dark green. `\atfi@color` is an attribute/value pair that gets inserted directly into the file attachment object. It defaults to the empty string, which means the PDF viewer gets to choose what color the icon should be. `\atfi@color@rgb` is only the RGB triplet itself and is used to insert a static graphic that represents Adobe Acrobat's rendition of a file attachment icon. It defaults to a beige color.

```
226 \define@key{AtFi}{color}{%
227   \def\atfi@color{/C [#1]}%
228   \def\atfi@color@rgb{#1}%
229 }
230 \setkeys{AtFi}{color=1 0.9255 0.7765}
```

`\atfi@author` Specify the author of the annotation. Adobe Acrobat shows this when you right-click on the annotation and choose *Properties*.

```
231 \def\atfi@author{}
232 \define@key{AtFi}{author}[]{}%
233 \edef\atfi@author{/T (#1)}%
234 }
```

`\atfi@pad@ii` Pad a number to exactly two digits. This is used by `\atfi@date` (below).

```
235 \def\atfi@pad@ii#1{%
236   \ifnum#1>9
237     #1%
238   \else
239     0#1%
240   \fi
241 }
```

`\atfi@timezone` Specify the timezone to attach to the file modification date. It would be awfully nice if \TeX had some way to produce this automatically. (Does it?)

```
242 \def\atfi@timezone{}
243 \define@key{AtFi}{timezone}{\def\atfi@timezone{#1}}
```

`\atfi@time` The date the annotation was last modified. It's unlikely you'd want to specify this explicitly in your \LaTeX document, but if you want to, you can. Seconds are hardwired to zero, and the time zone must be manually specified. (I don't believe \TeX makes either of those available.) Note that `\time` is stored in `\atfi@time` in case the minutes roll over during the time calculations. I was too lazy to do

the same for `\day`, `\month`, and `\year`, so don't process your L^AT_EX document at midnight if you want to get a correct timestamp.

```

244 \edef\atfi@time{\time}
245 \setcounter{atfi@tmp}{\atfi@time/60}
246 \edef\atfi@hours{\theatfi@tmp}
247 \setcounter{atfi@tmp}{\atfi@time-\atfi@hours*60}
248 \edef\atfi@minutes{\theatfi@tmp}
249 \def\atfi@date{%
250 /M (D:\the\year%
251     \expandafter\atfi@pad@ii\the\month
252     \expandafter\atfi@pad@ii\the\day
253     \atfi@pad@ii\atfi@hours
254     \atfi@pad@ii\atfi@minutes
255     00%
256     \atfi@timezone)%
257 }
258 \define@key{AtFi}{date}{%
259   \bgroup \Hy@unicodedefalse
260   \atfi@pdfstringdef\atfi@date{#1}%
261   \edef\next{\egroup
262     \def\noexpand\atfi@date{/M (\atfi@date)}%
263   }\next
264 }

```

`\atfi@description` Store the annotation's description. Adobe Acrobat shows this when you right-click on the annotation and choose *Properties*. It also shows it in the Annotations tab once you "Rescan Document".

```

265 \def\atfi@description{}
266 \define@key{AtFi}{description}{%
267   \atfi@pdfstringdef\atfi@description{#1}%
268   \edef\atfi@description{/Contents (\atfi@description)}%
269 }

```

`\atfi@subject` Store the annotation's subject. Adobe Acrobat shows this when you right-click on the annotation and choose *Properties*. It also shows it in the Annotations tab once you "Rescan Document".

```

270 \def\atfi@subject{}
271 \define@key{AtFi}{subject}{%
272   \atfi@pdfstringdef\atfi@subject{#1}%
273   \edef\atfi@subject{/Subj (\atfi@subject)}%
274 }

```

`\atfi@create` Store the annotation's creation date. Adobe Acrobat shows this when you right-click on the annotation and choose *Properties*. It also shows it in the Annotations tab once you "Rescan Document". Note that creation date is a PDF 1.5 feature.

```

275 \def\atfi@create{}
276 \define@key{AtFi}{created}{%
277   \bgroup \Hy@unicodedefalse
278   \atfi@pdfstringdef\atfi@create{#1}%

```

```

279 \edef\next{\egroup
280 \def\noexpand\atfi@credate{/CreationDate (\atfi@credate)}%
281 }\next
282 }

```

`\atfi@moddate` Store the annotation’s modification date. Adobe Acrobat shows this when you right-click on the annotation and choose *Properties*. It also shows it in the Annotations tab once you “Rescan Document”. Note that modification date is a PDF 1.5 feature.

```

283 \def\atfi@moddate{}
284 \define@key{AtFi}{modified}{%
285 \bgroup \Hy@unicodedefalse
286 \atfi@pdfstringdef\atfi@moddate{#1}%
287 \edef\next{\egroup
288 \def\noexpand\atfi@moddate{/ModDate (\atfi@moddate)}%
289 }\next
290 }

```

`\atfi@size` Store the annotation’s file size. Adobe Acrobat shows this when you right-click on the annotation and choose *Properties*. It also shows it in the Annotations tab once you “Rescan Document”. Note that file size is a PDF 1.5 feature.

```

291 \def\atfi@size{}
292 \def\atfi@dsize{}
293 \define@key{AtFi}{size}{%
294 \bgroup \Hy@unicodedefalse
295 \atfi@pdfstringdef\atfi@size{#1}%
296 \edef\next{\egroup
297 \def\noexpand\atfi@size{/Size \atfi@size}%
298 \def\noexpand\atfi@dsize{/DL \atfi@size}%
299 }\next
300 }

```

`\ifatfi@print` By default, file annotation icons print along with the rest of the document. (In Adobe Acrobat, that’s the case if and only if the Annotations box is checked in the Print dialog.) By setting `print=false`, the icons will not print.

```

301 \newif\ifatfi@print
302 \atfi@printtrue
303 \define@key{AtFi}{print}[true]{\csname atfi@print#1\endcsname}

```

`\ifatfi@zoom` By default, file annotation icons zoom along with the rest of the document. By setting `zoom=false`, the icons will remain at a constant size, regardless of magnification.

```

304 \newif\ifatfi@zoom
305 \atfi@zoomtrue
306 \define@key{AtFi}{zoom}[true]{\csname atfi@zoom#1\endcsname}

```

`\ifatfi@appearance` The attachfile package normally embeds an icon graphic in each file attachment annotation’s appearance dictionary. By setting `appearance=false`, no appearance dictionary is embedded.

ance dictionary will be added to a file attachment annotation; the PDF viewer will need to decide for itself how to display the icon.

```
307 \newif\ifatfi@appearance
308 \atfi@appearancetrue
309 \def\atfi@appearance@dict{}
310 \define@key{AtFi}{appearance}[true]{\csname atfi@appearance#1\endcsname}
```

6.6 Author commands

The commands described in this section are those available to the user writing a L^AT_EX document. If the macros seem too simple, it's because all the work is performed by the helper routines in Section 6.4 and the option-processing routines in Section 6.5.

`\attachfilesetup` Set default values for all the various annotation options.

```
311 \DeclareRobustCommand{\attachfilesetup}[1]{\setkeys{AtFi}{#1}}
```

`\attachfile` Given a filename and an optional set of attachment options, embed the corresponding file into the generated PDF file, and place an icon at the current location. The real work is performed by `\atfi@attachfile`. `\attachfile` merely sets up the category codes in such a way as to allow filenames to contain special characters such as underscores.

```
312 \DeclareRobustCommand{\attachfile}[1][]{%
313   \begingroup
314     \let\do\@makeother
315     \dospecials
316     \catcode'\=0\relax
317     \catcode'\{=1\relax
318     \catcode'\}=2\relax
319     \atfi@attachfile{#1}%
320 }
```

`\textattachfile` Given a filename, some arbitrary text, and an optional set of attachment options, embed the corresponding file into the generated PDF file, and use the text as the icon. After setting up the category codes to use for processing the filename, `\textattachfile` passes to control to `\atfi@textattachfile`, which resets the category codes, and then to `\atfi@textattachfile@i`, which does all the work. We define two groups: one for keeping the attachment options local and one for temporarily altering category codes.

```
321 \DeclareRobustCommand{\textattachfile}[1][]{%
322   \begingroup
323     \begingroup
324       \let\do\@makeother
325       \dospecials
326       \catcode'\=0\relax
327       \catcode'\{=1\relax
328       \catcode'\}=2\relax
329       \atfi@textattachfile{#1}%
```

```
330 }
```

`\noattachfile` Insert the same icon into the document that we would for an `\attachfile` call. This is useful for writing documentation that instructs a user on how to deal with file attachments. `\noattachfile` is fairly simple; is just calls `\setkeys` in order to get the latest values of `\atfi@icon@icon` and `\atfi@color@rgb`, and then it defers to one of `\atfi@acroGraph`, `\atfi@acroPaperclip`, `\atfi@acroPushPin`, or `\atfi@acroTag`, which do the actual rendering work.

```
331 \DeclareRobustCommand{\noattachfile}[1] [] {%
332   \begingroup
333     \setkeys{AtFi}{#1}%
334     \ifatfi@print
335     \csname atfi@acro\atfi@icon@icon\endcsname
336   \else
337     \setbox0=\hbox{\csname atfi@acro\atfi@icon@icon\endcsname}%
338     \makebox[\wd0]{}%
339   \fi
340 \endgroup
341 }
```

`\notextattachfile` Insert the same text into the document that we would for a `\textattachfile` call. This is useful for writing documentation that instructs a user on how to deal with file attachments.

```
342 \DeclareRobustCommand{\notextattachfile}[2] [] {%
343   \begingroup
344     \setkeys{AtFi}{#1}%
345     \ifatfi@print
346     \def\atfi@textcolor{##1 ##2 ##3}##4{%
347       \textcolor[rgb]{##1,##2,##3}{##4}}%
348     \expandafter\atfi@textcolor\expandafter
349       (\atfi@color@rgb){#2\strut}%
350   \else
351     \setbox0=\hbox{#2\strut}%
352     \makebox[\wd0]{}%
353   \fi
354 \endgroup
355 }
```

6.7 Dummy commands

If the author is not using pdf \LaTeX or Lua \LaTeX or not using it in PDF-generating mode, we replace the core `attachfile` commands with dummy versions so \LaTeX can at least run to completion.

```
356 \ifpdf
357 \else
```

`\atfi@dummy@pushpin` Define an empty space of approximately the same size as `\atfi@acroPushPin`.

```
358 \def\atfi@dummy@pushpin{%
```

```

359     \raisebox{-1.25bp}{\parbox[b][14bp]{24bp}{}}%
360   }

\textattachfile Define a dummy \textattachfile in terms of \notextattachfile.
361   \DeclareRobustCommand{\textattachfile}[3] [] {%
362     \notextattachfile[#1]{#3}%
363   }

\noattachfile Define a dummy \noattachfile in terms of \notextattachfile.
364   \DeclareRobustCommand{\noattachfile}[1] [] {%
365     \notextattachfile[#1]{\atfi@dummy@pushpin}%
366   }

\attachfile Define a dummy \attachfile in terms of the dummy \noattachfile.
367   \DeclareRobustCommand{\attachfile}[2] [] {%
368     \noattachfile[#1]%
369   }

370 \fi

```

7 Future work

The following are some avenues for future work on `attachfile`. First, `attachfile` supports only pdf \LaTeX and Lua \LaTeX for generating PDF files. It would be nice if it supported all the backends that `hyperref` supports: `dvipdfm`, `dvips` with `pdfmarks`, `VT \TeX` , and so forth. Along those same lines, a “draft” package option would be a welcome addition, for use when PDF is not the final output format.

Second, PDF supports platform-specific file attachments. That is, a file attachment icon can represent a different file when activated on Windows, Unix, or MacOS. It might be nice for `attachfile` to support that feature.

Finally, I’d like to see `attachfile` expand sometime to support *all* the various PDF annotations: Sound, Movie, Stamp, Ink, Popup, etc.

Of course, I make no promises that I’ll ever do *any* of the above. `attachfile` was just something I wrote in my spare time, and it’s unlikely I’ll be able to devote another large block of time to enhance it.

References

- [1] Adobe Systems Incorporated. *PDF Reference Version 1.6*. Adobe Press, fifth edition, December 3, 2004. ISBN 0321304748. Available from <http://partners.adobe.com/public/developer/en/pdf/PDFReference16.pdf>.
- [2] N. Freed and N. Borenstein. Multipurpose Internet Mail Extensions (MIME) part two: Media types. Request for Comments (RFC) 2046, Internet Engineering Task Force (IETF), Network Working Group, November 1996. Available from <http://www.rfc-editor.org/rfc/rfc2046.txt>.

- [3] Internet Assigned Numbers Authority. MIME media types. Available from <http://www.iana.org/assignments/media-types/>.

Change History

v1.0			
General: Initial version	1	
v1.1			
General: Completely restructured the .dtx file	1	
Wrote dummy versions of all the core macros to use in the absence of pdfL ^A T _E X/LuaL ^A T _E X running in PDF-generating mode.	21	
<code>\atfi@file</code> : Added explicit Rollover and Down appearances to work around browser bugs	..	14	
<code>\atfi@subject</code> : Added support for specifying the subject of an annotation	18	
<code>\noattachfile</code> : Modified to leave space on the page when <code>print=false</code> is passed as an option	21	
<code>\notextattachfile</code> : Created this function	21	
v1.1a			
General: Corrected a few stupid bugs	1	
v1.2			
General: Modified so as to enable filenames to contain special characters, e.g., underscores	...	1	
v1.2a			
<code>\atfi@mimetype</code> : Changed the MIME Subtype from a string to a name	16	
v1.3			
General: Incorporated Ross Moore's patches for making <code>attachfile</code> robust to running <code>hyperref</code> with <code>\Hy@unicodetrue</code> and for supporting the <code>Created</code> , <code>Modified</code> , and <code>Size</code> keys in the <code>EmbeddedFile's Params</code> dictionary	1	
			<code>\atfi@credate</code> : Added support for specifying the creation date of an annotation
		 18
			<code>\atfi@date</code> : Made robust to running <code>hyperref</code> with <code>\Hy@unicodetrue</code>
		 18
			<code>\atfi@dysize</code> : Added support for specifying the file size of an annotation
		 19
			<code>\atfi@embedfile</code> : Included a <code>Params</code> dictionary describing the file's date, modification date, and size
		 12
			<code>\atfi@file</code> : Made robust to running <code>hyperref</code> with <code>\Hy@unicodetrue</code>
		 14
			Modified to include the modification date in the <code>FileAttachment</code> dictionary
		 14
			<code>\atfi@moddate</code> : Added support for specifying the modification date of an annotation
		 19
v1.3a			
General: Corrected the formatting of the "not pdfL ^A T _E X" warning message	9	
v1.4			
<code>\c@atfi@tmp</code> : Incorporated Martin Münch's reduction of the number of counters that <code>attachfile</code> uses from five to one	12	
v1.5			
<code>\attachfile</code> : Reset "\ to category code 0 to enable the use of <code>\jobname</code> in the name of the file attachment (suggested by Felix Mueller-Sarnowski)	20	
v1.5a			
General: Modified the package to generate <code>attachfile.bib</code> automatically from the .dtx and .ins files	1	

v1.5b	<code>\textattachfile</code> : Reset “\” to category code 0 to enable the use of <code>\jobname</code> in the name of the file attachment for compatibility with <code>\attachfile</code> (reported by Uwe Bieling) 20	v1.7	<code>\theatfi@tmp</code> : Made the package robust to redefinitions of <code>\@arabic</code> , such as those made by the <code>babel</code> package (reported by Jonas Olson and fixed by Heiko Oberdiek) 12
v1.6	<code>\atfi@embedfile</code> : Don’t re-embed files that have already been embedded (feature proposed by Gareth Walker) 12	v1.8	<code>\atfi@file</code> : Strip path names from included files. Mikkel Futtrup reported that path names often confuse PDF readers on tablets and smartphones 14

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols	
<code>\#</code>	206
<code>\@makeother</code>	314, 324
<code>\{</code>	317, 327
<code>\}</code>	318, 328
A	
Adobe Acrobat	<i>2, 4–11, 17–19</i>
Reader	<i>9</i>
appearance (option)	<i>4, 8</i>
<code>\atfi@acroGraph</code>	<u>23</u>
<code>\atfi@acroGraph@data</code>	<u>15, 25</u>
<code>\atfi@acroPaperclip</code>	<u>51</u>
<code>\atfi@acroPaperclip@data</code>	<u>28, 53</u>
<code>\atfi@acroPushPin</code>	<u>62</u>
<code>\atfi@acroPushPin@data</code>	<u>56, 64</u>
<code>\atfi@acroTag</code>	<u>86</u>
<code>\atfi@acroTag@data</code>	<u>67, 88</u>
<code>\atfi@appearance@dict</code>	148, 160, <u>307</u>
<code>\atfi@appearance@obj</code>	<u>117</u>
<code>\atfi@appearancebox</code> <u>113</u> , 118–121, 124	
<code>\atfi@appearancedepth</code>	<u>113, 121, 158, 178</u>
<code>\atfi@appearancefalse</code>	<u>307</u>
<code>\atfi@appearanceheight</code>	<u>113, 120, 157, 177</u>
<code>\atfi@appearanceobj</code>	125, 150–152
<code>\atfi@appearancetrue</code>	<u>307</u>
<code>\atfi@appearancewidth</code>	<u>113, 119, 156, 179</u>
<code>\atfi@attachfile</code>	<u>181, 319</u>
<code>\atfi@author</code>	161, <u>231</u>
<code>\atfi@color</code>	162, <u>226</u>
<code>\atfi@color@rgb</code>	19, 42, 57, 74, 200, <u>226, 349</u>
<code>\atfi@create</code>	105, <u>275</u>
<code>\atfi@date</code>	163, <u>244</u>
<code>\atfi@description</code>	164, <u>265</u>
<code>\atfi@dysize</code>	103, <u>291</u>
<code>\atfi@dummy@pushpin</code>	<u>358, 365</u>
<code>\atfi@embedfile</code>	<u>98, 183, 195</u>
<code>\atfi@file</code>	<u>138</u>
<code>\atfi@flags</code>	<u>127, 168</u>
<code>\atfi@flags@to@int</code>	<u>127, 185, 201</u>
<code>\atfi@hours</code>	<u>244</u>
<code>\atfi@icon</code>	165, <u>221</u>
<code>\atfi@icon@icon</code>	184, <u>221, 335, 337</u>
<code>\atfi@insert@file@annot</code> <u>138</u> , 186, 202	
<code>\atfi@mime@subtype</code>	209, 218
<code>\atfi@mime@type</code>	209, 218
<code>\atfi@mimetype</code>	102, <u>213</u>
<code>\atfi@minutes</code>	<u>244</u>
<code>\atfi@moddate</code>	106, 166, <u>283</u>
<code>\atfi@pad@ii</code>	<u>235, 251–254</u>
<code>\atfi@pdf@slash</code>	<u>205, 218</u>

<code>\atfi@pdfstringdef</code>	91, 141, 215, 260, 267, 272, 278, 286, 295	<code>\filename@base</code>	146
<code>\atfi@printfalse</code>	301	<code>\filename@ext</code>	146
<code>\atfi@printtrue</code>	301	<code>\filename@parse</code>	145
<code>\atfi@set@appearance</code>	117, 184, 198	G	
<code>\atfi@size</code>	107, 291	GMT	7
<code>\atfi@split@mimetype</code>	209, 216	Graph	5, 6, 10, 16
<code>\atfi@subject</code>	167, 270	H	
<code>\atfi@temp@string</code>	91	<code>\Hy@unicodedefalse</code> 140, 259, 277, 285, 294	
<code>\atfi@textattachfile</code>	189, 329	hyperref (package)	2, 8, 10, 22, 23
<code>\atfi@textattachfile@i</code>	191, 193	I	
<code>\atfi@textcolor</code>	193, 346, 348	icon (option)	5, 8
<code>\atfi@time</code>	244	<code>\ifatfi@appearance</code>	147, 307
<code>\atfi@timezone</code>	242, 256	<code>\ifatfi@print</code>	129, 301, 334, 345
<code>\atfi@zoomfalse</code>	304	<code>\ifatfi@zoom</code>	132, 304
<code>\atfi@zoomtrue</code>	304	ifpdf (package)	8
attachfile (package)		Ink	22
.	1, 2, 4–9, 13, 14, 19, 21–23	K	
<code>\attachfile</code>	2, 312, 367	keyval (package)	8
<code>\attachfilesetup</code>	3, 311	L	
author (option)	4, 8	L ^A T _E X	1, 2, 5, 7–9, 17, 18, 20, 21
B		<code>\lccode</code>	206
babel (package)	24	<code>\lowercase</code>	207
B ^B L _A T _E X	2	LuaL ^A T _E X	8, 9, 21–23
C		M	
<code>\c@atfi@tmp</code>	96	MIME	6, 16, 23
calc (package)	8	mimetype (option)	5–6, 8
color (package)	8	Modified	23
color (option)	4–5, 8	modified (option)	6–8
Created	23	<code>\month</code>	251
created (option)	5, 7, 8	Movie	22
D		MPEG	6
date (option)	5, 7, 8	N	
<code>\day</code>	252	<code>\next</code>	142, 144, 261, 263, 279, 281, 287, 289, 296, 299
<code>\define@key</code>	214, 221, 226, 232, 243, 258, 266, 271, 276, 284, 293, 303, 306, 310	<code>\noattachfile</code>	2–3, 331, 364, 368
description (option)	5, 8	<code>\noexpand</code>	143, 262, 280, 288, 297, 298
DVI	9	<code>\notextattachfile</code>	3, 342, 362, 365
dvipdfm	9, 22	O	
dvips	9, 22	options	
E		appearance	4, 8
EmbeddedFile	12, 23	author	4, 8
F		color	4–5, 8
FileAttachment	4, 14, 23	created	5, 7, 8
		date	5, 7, 8
		default values	8

description	5, 8	<code>\RequirePackageWithOptions</code>	14
icon	5, 8	RGB	17
mimetype	5-6, 8	S	
modified	6-8	Size	23
print	6, 8	size (option)	6, 8
size	6, 8	Sound	22
subject	6-8	Stamp	22
timezone	5, 7-8	subject (option)	6-8
zoom	7-8	Subtype	23
P			
packages		T	
attachfile	1, 2, 4-9, 13, 14, 19, 21-23	Tag	5, 6, 10, 11, 16
babel	24	TEX	4-8, 12-17
calc	8	<code>\textattachfile</code>	2-3, 321, 361
color	8	<code>\textcolor</code>	197, 347
hyperref	2, 8, 10, 22, 23	<code>\theatfi@tmp</code>	96, 136, 246, 248
ifpdf	8	<code>\time</code>	244
keyval	8	timezone (option)	5, 7-8
<code>\PackageWarningNoLine</code>	4	U	
Paperclip	5, 6, 10, 16	Universal Time	7
Params	23	V	
PDF	1-10, 12-23	<code>\value</code>	97
<code>\pdfannot</code>	156	VTEX	9, 22
<code>\pdflastobj</code>	110	W	
<code>\pdflastxform</code>	125	WYSIWYG	1
pdfLATEX	1, 8, 9, 16, 21-23	X	
<code>\pdfliteral</code>	25, 53, 64, 88	XObject	13
pdfmarks	9, 22	Y	
<code>\pdfobj</code>	100	<code>\year</code>	250
<code>\pdfstringdef</code>	93	Z	
<code>\pdfxform</code>	122	zoom (option)	7-8
Popup	22		
print (option)	6, 8		
PushPin	5, 6, 10, 11, 16		
R			
<code>\RequirePackage</code>	1, 11-13		