

The embedfile package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2007/11/25 v2.3

Abstract

This package embeds files to a PDF document. Currently the only supported driver is pdfTeX >= 1.30 in PDF mode.

Contents

1	Documentation	2
1.1	Introduction	2
1.1.1	Future development	2
1.2	User interface	2
1.3	Collection support (PDF 1.7)	3
1.4	Export of object references	5
1.4.1	Example	5
1.5	Examples	5
1.5.1	plain-TeX	5
1.5.2	Collection example	6
1.6	Package dtx-attach	7
2	Implementation	7
2.1	Reload check and package identification	7
2.2	Catcodes	8
2.3	Tools	9
2.4	Check for recent pdfTeX in PDF mode	9
2.5	Strings	10
2.6	Switches	11
2.7	Key value definitions	11
2.8	Embed the file	18
3	Test	22
3.1	Catcode checks for loading	22
3.2	Simple test	24
4	Installation	24
4.1	Download	24
4.2	Bundle installation	24
4.3	Package installation	25
4.4	Refresh file name databases	25
4.5	Some details for the interested	25
5	References	26

6 History	26
[2006/08/16 v1.0]	26
[2007/04/11 v1.1]	26
[2007/09/09 v1.2]	26
[2007/10/28 v2.0]	26
[2007/10/29 v2.1]	26
[2007/11/11 v2.2]	26
[2007/11/25 v2.3]	27
7 Index	27

1 Documentation

1.1 Introduction

The PDF format ([3]) allows the inclusion of files inside the PDF document. The included files can be bound to an annotation on a page. Or they can be recorded in a sorted list of embedded files. The packages `attachfile` or `attachfile2` follow the first approach, this package uses the latter method.

1.1.1 Future development

My dream is a large package that merges the features of all these packages mentioned before:

- Files can be attached to a page.
- Files can be attached to the document.
- An easy user interface for simple, common tasks and beginners.
- An interface for the advanced users that want to setup every detail.
- Support of many drivers (`pdftex`, `dvips`, `dvipdfm`, ...).
- ...

However, I have not managed to take the time for this project. Instead:

- First I experimented with package `attachfile`, adding driver support, fixing bugs, The result is currently named as `attachfile2`. It uses an external script to get file properties (size, date, checksum, ...).
- In order to avoid an external program for getting basic file properties I provided a patch “EscapeAndOther” for pdfTeX that was accepted for version 1.30.
- This package closes a gap left by the packages for attaching files and allows the embedding of files to the document. Also it makes use of the new primitives of pdfTeX.

1.2 User interface

This package `embedfile` can be used with both \LaTeX and plain- \TeX . See [subsubsection 1.5.1](#) that explains the use with plain- \TeX by an example. In \LaTeX the package is loaded as usually. There are no options.

```
\usepackage{embedfile}
```

`\embedfile` [*options*] {*file*}

The macro `\embedfile` includes file *file* and attaches it to the PDF document. At the end of the document the sorted list of embedded files are written. Thus you can safely use `\embedfile` before `\end{document}`. Embedding files using `\AtEndDocument` will only work, if `\AtEndDocument` is called before loading the package `embedfile`.

The *options* are give as key value pairs. The following keys are supported:

filespec This allows to override the file name that appears in the PDF file. If you are using other than simple file names (8bit, path separators, ...), look into the PDF specification ([3]). There are rules how these file names must be written/encoded.

filesystem This sets the entry `/FS` in the file specification dictionary, see PDF specification ([3]). Example: `filesystem=URL`.

mimetype This sets the mime type ([4]) of the file, see [subsection 1.5.1](#) for examples and [5] for a list of officially registered types.

desc The description for the file.

stringmethod The package must convert the values of the keys `filespec` and `desc` into a PDF string. If `hyperref` is found, then its `\pdfstringdef` will be used, otherwise pdfTeX's `\pdfescapestring` is used. Value `psd` forces the use of `\pdfstringdef`, value `escape` the use of `\pdfescapestring`.

<key>.value Sets the value of a collection item property, see [section 1.3](#).

<key>.prefix Sets the prefix of a collection item property, see [section 1.3](#).

id The value must be an unique name. Macros `\embedfileifobjectexists` and `\embedfilegetobject` are using this name later.

`\embedfilefinish`

The list of all embedded files must be added as data structure in the PDF file. In case of L^AT_EX this is automatically done. The package uses `\AtEndDocument`. Then the list of all files should be known. However, plain-T_EX does not know about `\AtEndDocument`. Thus the user must call `\embedfilefinish` at the end of the document after the last file is embedded.

`\embedfilesetup` {*options*}

Options for `\embedfile` and collection support can be set in `\embedfilesetup`.

1.3 Collection support (PDF 1.7)

Since PDF 1.7 the embedded files can form a *collection* (sometimes referred as *package*), the main document is called *cover sheet*. See PDF specification 8.2.4 “Collections” and 3.10.5 “Collection items” [3].

Usually Acrobat Reader 7 or 8 shows the embedded files in a table at the bottom with the following columns:

Name	Description	Modified	Size
...

If the files form a collection, then they are displayed in a table left or top (depending on option `view`, see `\embedfilesetup`).

Collection support is enabled automatically, if it is used.

`\embedfilesetup {<options>}`

The following options are supported in addition to options for `\embedfile`:

view If the PDF file contains a collection, then Acrobat Reader 8 shows a line at the top below the menu bar and the toolbar. It shows the current selected file, icons for changing the view mode, an options menu. The initial mode how the collection is presented is set by this option `view`. The following modes/values are supported, the default is `details`:

details The full collection table is displayed at the top below the collection bar.

tile The files of the collection are shown in tile mode on the left.

hidden The collection table is not shown.

initialfile Selects the file that is initially presented. Especially useful for an embedded PDF file that is then shown instead of the cover document.

`\embedfilefield {<key>} {<options>}`

Macro `\embedfilefield` defines a column/field in the collection table. The name of the field is `<key>`.

type sets the type of the field. The supported values are:

text A text field. Its value is set in `\embedfile` by option `<key>.value`.

date A date field. Its value is set in `\embedfile` by option `<key>.value`. A special format is required, see “3.8.3 Dates” [\[3\]](#).

number A field with an integer or float number. Its value is set in `\embedfile` by option `<key>.value`.

file The file name of the embedded file.

desc The description text of the embedded file. It is set in `\embedfile` by option `desc`.

moddate The modification date of the embedded file.

size The size of the embedded file.

All types allow the use of a prefix that is disregarded by sorting. The prefix for this field is set in `\embedfile` by option `<key>.prefix`.

title sets the column title.

visible controls whether the column is presented:

true shows the column.

false hides the column.

Default: `true`

edit Allows the editing of field values. Does not seem to have an effect for Acrobat Reader.

true enables the feature, if available (depends on the PDF viewer).

false disables the feature.

Default: `false`

The order of `\embedfilefield` statements defines the order of the columns.

`\embedfilesort {⟨key-sort-list⟩}`

The sort order of the embedded files are controlled by macro `\embedfilesort`. `⟨key-sort-list⟩` defines the sort order. The key is a field name defined by `\embedfilefield`. Its value is either `ascending` or `descending`. The default is `ascending`.

1.4 Export of object references

Caution: This feature is still experimental. It may be even removed in future versions. Therefore feedback would be nice, if someone has a useful application for this feature.

Object numbers are saved, if `id` is given in `\embedfile`. The following objects are supported:

- `EmbeddedFile`
- `Filespec`

`\embedfileifobjectexists {⟨id⟩} {⟨type⟩} {⟨then⟩} {⟨else⟩}`

Macro `\embedfileifobjectexists` tests whether object of `⟨type⟩` is available for the embedded file identified by `⟨id⟩`.

`\embedfilegetobject {⟨id⟩} {⟨type⟩}`

Macro `\embedfilegetobject` expands to the full object reference object of `⟨type⟩` for the embedded file identified by `⟨id⟩`.

1.4.1 Example

```
\embedfile[id={foo}]{foo.pdf}
\embedfileifobjectexists{foo}{Filespec}{%
  \typeout{%
    FileSpec object for 'foo': %
    \embedfilegetobject{foo}{Filespec}%
  }%
}{%
  \typeout{No Filespec object for 'foo'}%
}
```

1.5 Examples

1.5.1 plain-TeX

The package can be used with plain-TeX. It can be used with or without help from `miniltx.tex`.

If additionally package `keyval` (`graphicx`) is needed, load it first. Then package `embedfile` avoids a duplicate loading of package `keyval`.

Because plain-TeX does not provide a hook at end of the document, you have to call `\embedfilefinish` manually at the end after the last embedded file.

```
1 ⟨*exampleplain⟩
2 %<<END
3 % Load packages
4 \input miniltx
5 % \def\Gin@driver{pdftex.def}
6 % \input graphicx.sty
```

```

7 \input embedfile.sty
8 \resetatcatcode
9
10 % default setting
11 \embedfilesetup{
12   mimetype=text/plain
13 }
14
15 % Embed files
16 \embedfile[
17   filespec=example.tex,
18   desc={Source code (plain-TeX) of this example}
19 ]{embedfile-example-plain.tex}
20
21 \embedfile[
22   desc={Source of package 'embedfile'}
23 ]{embedfile.dtx}
24
25 \embedfile[
26   mimetype=application/pdf,
27   desc={Documentation of package 'embedfile'}
28 ]{embedfile.pdf}
29
30 % Some text
31 This example document contains three embedded files.
32
33 % End of document
34 \embedfilefinish % don't forget
35 \bye
36 %END
37 </exampleplain>

```

1.5.2 Collection example

```

38 (*examplecollection)
39 %<<END
40 \NeedsTeXFormat{LaTeX2e}
41 \documentclass{article}
42 \usepackage[bookmarks=false]{hyperref}
43 % provides \pdfstringdef that is then used by 'title' and
44 % other keys.
45 \usepackage{embedfile}[2007/11/25]
46 \embedfilesetup{
47   view=details,
48   initialfile=embedfile.pdf
49 }
50 \embedfilefield{file}{
51   type=file,
52   title={File name}
53 }
54 \embedfilefield{description}{
55   type=desc,
56   title={Description}
57 }
58 \embedfilefield{date}{
59   type=moddate,
60   title={Date}
61 }
62 \embedfilefield{size}{
63   type=size,
64   title={Size}
65 }
66 \embedfilefield{type}{

```

```

67  type=text,
68  title={Type},
69  visible=false
70 }
71 \embedfilesort{
72  type,
73  date=descending
74 }
75 \begin{document}
76 An example for embedded files as collection.
77 You need Acrobat Reader 8 or higher.
78
79 \embedfile[
80  desc={Source file of package 'embedfile'},
81  description.prefix={Package: },
82  type.value={DTX}
83 ]{embedfile.dtx}
84
85 \embedfile[
86  desc={Documentation of package 'embedfile'},
87  description.prefix={Package: },
88  type.value={PDF}
89 ]{embedfile.pdf}
90
91 \embedfile[
92  desc={The source for this example},
93  description.prefix={Example: },
94  type.value={TEX}
95 ]{\jobname.tex}
96
97 \end{document}
98 %END
99 \</examplecollection>

```

1.6 Package dtx-attach

Package `dtx-attach` is just a small application of package `embedfile`. I am using it for the CTAN documentation of my packages in [CTAN:macros/latex/contrib/oberdiek/](#). It also serves as small example for the use of the package with \LaTeX .

```

100 (*dtxattach)
101 \NeedsTeXFormat{LaTeX2e}
102 \ProvidesPackage{dtx-attach}
103 [2007/11/25 v2.3 Embed \string\jobname.dtx (HO)]%
104 \RequirePackage{embedfile}[2007/11/25]
105 \embedfile[%
106  stringmethod=escape,%
107  mimetype=plain/text,%
108  desc={LaTeX docstrip source archive for package '\jobname'}%
109 ]{\jobname.dtx}
110 \</dtxattach>

```

2 Implementation

```

111 (*package)

```

2.1 Reload check and package identification

Reload check, especially if the package is not used with \LaTeX .

```

112 \begingroup
113  \catcode44 12 % ,
114  \catcode45 12 % -
115  \catcode46 12 % .

```

```

116 \catcode58 12 % :
117 \catcode64 11 % @
118 \expandafter\let\expandafter\x\csname ver@embedfile.sty\endcsname
119 \ifcase 0%
120 \ifx\x\relax % plain
121 \else
122 \ifx\x\empty % LaTeX
123 \else
124 1%
125 \fi
126 \fi
127 \else
128 \catcode35 6 % #
129 \catcode123 1 % {
130 \catcode125 2 % }
131 \expandafter\ifx\csname PackageInfo\endcsname\relax
132 \def\x#1#2{%
133 \immediate\write-1{Package #1 Info: #2.}%
134 }%
135 \else
136 \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
137 \fi
138 \x{embedfile}{The package is already loaded}%
139 \endgroup
140 \expandafter\endinput
141 \fi
142 \endgroup
Package identification:
143 \begingroup
144 \catcode35 6 % #
145 \catcode40 12 % (
146 \catcode41 12 % )
147 \catcode44 12 % ,
148 \catcode45 12 % -
149 \catcode46 12 % .
150 \catcode47 12 % /
151 \catcode58 12 % :
152 \catcode64 11 % @
153 \catcode123 1 % {
154 \catcode125 2 % }
155 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
156 \def\x#1#2#3[#4]{\endgroup
157 \immediate\write-1{Package: #3 #4}%
158 \xdef#1{#4}%
159 }%
160 \else
161 \def\x#1#2[#3]{\endgroup
162 #2[{#3}]%
163 \ifx#1\relax
164 \xdef#1{#3}%
165 \fi
166 }%
167 \fi
168 \expandafter\x\csname ver@embedfile.sty\endcsname
169 \ProvidesPackage{embedfile}%
170 [2007/11/25 v2.3 embed files into PDF (HO)]

```

2.2 Catcodes

```

171 \begingroup
172 \catcode123 1 % {
173 \catcode125 2 % }

```



```

174 \def\x{\endgroup
175 \expandafter\edef\csname EmFi@AtEnd\endcsname{%
176 \catcode35 \the\catcode35\relax
177 \catcode64 \the\catcode64\relax
178 \catcode123 \the\catcode123\relax
179 \catcode125 \the\catcode125\relax
180 }%
181 }%
182 \x
183 \catcode35 6 % #
184 \catcode64 11 % @
185 \catcode123 1 % {
186 \catcode125 2 % }
187 \def\TMP@EnsureCode#1#2{%
188 \edef\EmFi@AtEnd{%
189 \EmFi@AtEnd
190 \catcode#1 \the\catcode#1\relax
191 }%
192 \catcode#1 #2\relax
193 }
194 \TMP@EnsureCode{39}{12}% '
195 \TMP@EnsureCode{40}{12}% (
196 \TMP@EnsureCode{41}{12}% )
197 \TMP@EnsureCode{44}{12}% ,
198 \TMP@EnsureCode{46}{12}% .
199 \TMP@EnsureCode{47}{12}% /
200 \TMP@EnsureCode{58}{12}% :
201 \TMP@EnsureCode{60}{12}% <
202 \TMP@EnsureCode{61}{12}% =
203 \TMP@EnsureCode{62}{12}% >
204 \TMP@EnsureCode{91}{12}% [
205 \TMP@EnsureCode{93}{12}% ]
206 \TMP@EnsureCode{96}{12}% `

```

2.3 Tools

\EmFi@RequirePackage

```

207 \begingroup\expandafter\expandafter\expandafter\endgroup
208 \expandafter\ifx\csname RequirePackage\endcsname\relax
209 \def\EmFi@RequirePackage#1[#2]{%
210 \input #1.sty\relax
211 }%
212 \else
213 \let\EmFi@RequirePackage\RequirePackage
214 \fi

```

\EmFi@Error

```

215 \EmFi@RequirePackage{infwarerr}[2007/09/09]%
216 \def\EmFi@Error{%
217 \@PackageError{embedfile}%
218 }

```

2.4 Check for recent pdfTeX in PDF mode

Load package ifpdf and check mode.

```

219 \EmFi@RequirePackage{ifpdf}[2007/09/09]
220 \ifpdf
221 \else
222 \EmFi@Error{%
223 Missing pdfTeX in PDF mode%
224 }{%
225 Currently other drivers are not supported. %

```

```

226     Package loading is aborted.%
227 }%
228 \EmFi@AtEnd
229 \expandafter\endinput
230 \fi

231 \EmFi@RequirePackage{pdftexcmds}[2007/11/11]

Check version.

232 \begingroup\expandafter\expandafter\expandafter\endgroup
233 \expandafter\ifx\csname pdf@filesize\endcsname\relax
234   \EmFi@Error{%
235     Unsupported pdfTeX version%
236   }%
237   At least version 1.30 is necessary. Package loading is aborted.%
238 }%
239 \EmFi@AtEnd
240 \expandafter\endinput
241 \fi

```

2.5 Strings

Minimal version of package pdfescape is 2007/08/27 v1.5 because of \EdefSanitize.

```

242 \EmFi@RequirePackage{pdfescape}[2007/11/11]

243 \def\EmFi@temp#1{%
244   \expandafter\EdefSanitize\csname EmFi@S@#1\endcsname{#1}%
245 }

\EmFi@details

246 \EmFi@temp{details}%

\EmFi@tile

247 \EmFi@temp{tile}%

\EmFi@hidden

248 \EmFi@temp{hidden}%

\EmFi@S@text

249 \EmFi@temp{text}

\EmFi@S@date

250 \EmFi@temp{date}

\EmFi@S@number

251 \EmFi@temp{number}

\EmFi@S@file

252 \EmFi@temp{file}

\EmFi@S@desc

253 \EmFi@temp{desc}

\EmFi@S@moddate

254 \EmFi@temp{moddate}

\EmFi@S@creationdate

255 \EmFi@temp{creationdate}

\EmFi@S@size

256 \EmFi@temp{size}

```

```

\EmFi@S@ascending
257 \EmFi@temp{ascending}

\EmFi@S@descending
258 \EmFi@temp{descending}

\EmFi@S@true
259 \EmFi@temp{true}

\EmFi@S@false
260 \EmFi@temp{false}

```

2.6 Switches

```

\ifEmFi@collection
261 \newif\ifEmFi@collection

\ifEmFi@initialfile
262 \newif\ifEmFi@initialfile

\ifEmFi@sort
263 \newif\ifEmFi@sort

\ifEmFi@visible
264 \newif\ifEmFi@visible

\ifEmFi@edit
265 \newif\ifEmFi@edit

\ifEmFi@item
266 \newif\ifEmFi@item

\ifEmFi@finished
267 \newif\ifEmFi@finished

\ifEmFi@id
268 \newif\ifEmFi@id

```

2.7 Key value definitions

```

269 \expandafter\ifx\csname define@key\endcsname\relax
270 \chardef\EmFi@plain=\z@
271 \def\EmFi@temp#1{%
272 \begingroup\expandafter\expandafter\expandafter\endgroup
273 \expandafter\ifx\csname#1\endcsname\relax
274 \chardef\EmFi@plain=\@ne
275 \fi
276 }%
277 \EmFi@temp{NeedsTeXFormat}%
278 \EmFi@temp{ProvidesPackage}%
279 \EmFi@temp{DeclareOption}%
280 \EmFi@temp{ExecuteOptions}%
281 \EmFi@temp{ProcessOptions}%
282 \ifnum\EmFi@plain=\@ne
283 \def\EmFi@temp#1{%
284 \expandafter\let\csname EmFi@Org#1\expandafter\endcsname
285 \csname#1\endcsname
286 \expandafter\def\csname#1\endcsname
287 }%

```

```

288 \EmFi@temp{NeedsTeXFormat}#1{}%
289 \EmFi@temp{ProvidesPackage}#1[#2]{}% hash-ok
290 \EmFi@temp{DeclareOption}#1{}%
291 \EmFi@temp{ExecuteOptions}#1{}%
292 \EmFi@temp{ProcessOptions}{}%

\KV@errx LATEX's option processing is not available with plain-TEX. Thus we define the de-
fault error command \KV@errx here, also using package infwarerr's \@PackageError.

293 \def\KV@errx#1{%
294 \PackageError{keyval}{#1}\@ehc
295 }%

```

Other macros from L^AT_EX's kernel that are used by package keyval.

```

\@ifnextchar

296 \expandafter\ifx\csname @ifnextchar\endcsname\relax
297 \def\@ifnextchar#1#2#3{%
298 \let\reserved@d=#1%
299 \def\reserved@a{#2}%
300 \def\reserved@b{#3}%
301 \futurelet\@let@token\@ifnch
302 }%
303 \def\@ifnch{%
304 \ifx\@let@token\@sptoken
305 \let\reserved@c\@xifnch
306 \else
307 \ifx\@let@token\reserved@d
308 \let\reserved@c\reserved@a
309 \else
310 \let\reserved@c\reserved@b
311 \fi
312 \fi
313 \reserved@c
314 }%
315 \begingroup
316 \def\:{\global\let\@sptoken= }%
317 \: % this makes \@sptoken a space token
318 \def\:{\@xifnch}%
319 \expandafter\gdef\:{%
320 \futurelet\@let@token\@ifnch
321 }%
322 \endgroup
323 \fi

\@namedef

324 \expandafter\ifx\csname @namedef\endcsname\relax
325 \def\@namedef#1{%
326 \expandafter\def\csname#1\endcsname
327 }%
328 \fi

329 \fi
330 \EmFi@RequirePackage{keyval}[1999/03/16]%
331 \ifnum\EmFi@plain=\@ne
332 \def\EmFi@temp#1{%
333 \expandafter\let\csname#1\expandafter\endcsname
334 \csname EmFi@Org#1\endcsname
335 }%
336 \EmFi@temp{NeedsTeXFormat}%
337 \EmFi@temp{ProvidesPackage}%
338 \EmFi@temp{DeclareOption}%
339 \EmFi@temp{ExecuteOptions}%
340 \EmFi@temp{ProcessOptions}%

```

```

341 \fi
342 \fi

\EmFi@GlobalKey
343 \def\EmFi@GlobalKey#1#2{%
344 \global\expandafter\let\csname KV@#1@#2\expandafter\endcsname
345 \csname KV@#1@#2\endcsname
346 }

\EmFi@GlobalDefaultKey
347 \def\EmFi@GlobalDefaultKey#1#2{%
348 \EmFi@GlobalKey{#1}{#2}%
349 \global\expandafter\let
350 \csname KV@#1@#2@default\expandafter\endcsname
351 \csname KV@#1@#2@default\endcsname
352 }

\EmFi@DefineKey
353 \def\EmFi@DefineKey#1#2{%
354 \define@key{EmFi}{#1}{%
355 \expandafter\def\csname EmFi@#1\endcsname{##1}%
356 }%
357 \expandafter\def\csname EmFi@#1\endcsname{#2}%
358 }

Subtype of the embedded file (optional).
359 \EmFi@DefineKey{mimetype}{}

File specification string.
360 \EmFi@DefineKey{filespec}{\EmFi@file}

File system (optional).
361 \EmFi@DefineKey{filesystem}{}

Description (optional).
362 \EmFi@DefineKey{desc}{}

Method for converting text to PDF strings.
363 \EmFi@DefineKey{stringmethod}{%
364 \ifx\pdfstringdef\@undefined
365 escape%
366 \else
367 \ifx\pdfstringdef\relax
368 escape%
369 \else
370 psd%
371 \fi
372 \fi
373 }

Option id as key for object numbers.
374 \define@key{EmFi}{id}{%
375 \def\EmFi@id{#1}%
376 \EmFi@idtrue
377 }

\EmFi@defobj
378 \def\EmFi@defobj#1{%
379 \ifEmFi@id
380 \expandafter\xdef\csname EmFi@#1@\EmFi@id\endcsname{%
381 \the\pdflastobj\space 0 R%
382 }%
383 \fi
384 }

```

\embedfileifobjectexists

```
385 \def\embedfileifobjectexists#1#2{%
386   \expandafter\ifx\csname EmFi@#2@#1\endcsname\relax
387     \expandafter\@secondoftwo
388   \else
389     \expandafter\@firstoftwo
390   \fi
391 }
```

\@firstoftwo

```
392 \expandafter\ifx\csname @firstoftwo\endcsname\relax
393   \long\def\@firstoftwo#1#2{#1}%
394 \fi
```

\@secondoftwo

```
395 \expandafter\ifx\csname @secondoftwo\endcsname\relax
396   \long\def\@secondoftwo#1#2{#2}%
397 \fi
```

\embedfilegetobject

```
398 \def\embedfilegetobject#1#2{%
399   \embedfileifobjectexists{#1}{#2}{%
400     \csname EmFi@#2@#1\endcsname
401   }{%
402     O O R%
403   }%
404 }
```

Initial view of the collection.

```
405 \define@key{EmFi}{view}[]{%
406   \EdefSanitize\EmFi@temp{#1}%
407   \def\EmFi@next{%
408     \global\EmFi@collectiontrue
409   }%
410   \ifx\EmFi@temp\empty
411     \let\EmFi@view\EmFi@S@details
412   \else\ifx\EmFi@temp\EmFi@S@details
413     \let\EmFi@view\EmFi@S@details
414   \else\ifx\EmFi@temp\EmFi@S@tile
415     \let\EmFi@view\EmFi@S@tile
416   \else\ifx\EmFi@temp\EmFi@S@hidden
417     \let\EmFi@view\EmFi@S@hidden
418   \else
419     \let\EmFi@next\relax
420     \EmFi@Error{%
421       Unknown value '\EmFi@temp' for key 'view'.\MessageBreak
422       Supported values: 'details', 'tile', 'hidden'.%
423     }\@ehc
424   \fi\fi\fi\fi
425   \EmFi@next
426 }
427 \EmFi@DefineKey{initialfile}{}
```

\embedfilessetup

```
428 \def\embedfilessetup{%
429   \ifEmFi@finished
430     \def\EmFi@next##1{%
431       \EmFi@Error{%
432         \string\embedfilefield\space after \string\embedfilefinish
433       }{%
434         The list of embedded files is already written.%

```

```

435     }%
436 \else
437     \def\EmFi@next{%
438         \setkeys{EmFi}%
439     }%
440 \fi
441 \EmFi@next
442 }

\EmFi@schema
443 \def\EmFi@schema{}

\EmFi@order
444 \gdef\EmFi@order{0}

\EmFi@@order
445 \let\EmFi@@order\relax

\EmFi@fieldlist
446 \def\EmFi@fieldlist{}

\EmFi@sortcase
447 \def\EmFi@sortcase{0}%

\embedfilefield
448 \def\embedfilefield#1#2{%
449     \ifEmFi@finished
450         \EmFi@Error{%
451             \string\embedfilefield\space after \string\embedfilefinish
452         }{%
453             The list of embedded files is already written.%
454         }%
455     \else
456         \global\EmFi@collectiontrue
457         \EdefSanitize\EmFi@key{#1}%
458         \expandafter\ifx\csname KV@EmFi@\EmFi@key.prefix\endcsname\relax
459         \begingroup
460             \count@=\EmFi@order
461             \advance\count@ 1 %
462             \xdef\EmFi@order{\the\count@}%
463             \let\EmFi@title\EmFi@key
464             \let\EmFi@type\EmFi@S@text
465             \EmFi@visibletrue
466             \EmFi@editfalse
467             \setkeys{EmFiFi}{#2}%
468             \EmFi@convert\EmFi@title\EmFi@title
469             \xdef\EmFi@schema{%
470                 \EmFi@schema
471                 /\pdf@escapename{\EmFi@key}<<%
472                 /Subtype/%
473                 \ifx\EmFi@type\EmFi@S@date D%
474                 \else\ifx\EmFi@type\EmFi@S@number N%
475                 \else\ifx\EmFi@type\EmFi@S@file F%
476                 \else\ifx\EmFi@type\EmFi@S@desc Desc%
477                 \else\ifx\EmFi@type\EmFi@S@moddate ModDate%
478                 \else\ifx\EmFi@type\EmFi@S@creationdate CreationDate%
479                 \else\ifx\EmFi@type\EmFi@S@size Size%
480                 \else S%
481                 \fi\fi\fi\fi\fi\fi\fi
482                 /N(\EmFi@title)%
483                 \EmFi@@order{\EmFi@order}%

```

```

484         \ifEmFi@visible
485     \else
486         /V false%
487     \fi
488     \ifEmFi@edit
489         /E true%
490     \fi
491     >>%
492 }%
493 \let\do\relax
494 \xdef\EmFi@fieldlist{%
495     \EmFi@fieldlist
496     \do{\EmFi@key}%
497 }%
498 \ifx\EmFi@type\EmFi@S@text
499     \define@key{EmFi}{\EmFi@key.value}{%
500         \EmFi@itemtrue
501         \def\EmFi@temp{##1}%
502         \EmFi@convert\EmFi@temp\EmFi@temp
503         \expandafter\def\csname EmFi@V@#1%
504         \expandafter\endcsname\expandafter{%
505             \expandafter(\EmFi@temp)%
506         }%
507     }%
508     \EmFi@GlobalKey{EmFi}{\EmFi@key.value}%
509 \else\ifx\EmFi@type\EmFi@S@date
510     \define@key{EmFi}{\EmFi@key.value}{%
511         \EmFi@itemtrue
512         \def\EmFi@temp{##1}%
513         \EmFi@convert\EmFi@temp\EmFi@temp
514         \expandafter\def\csname EmFi@V@#1%
515         \expandafter\endcsname\expandafter{%
516             \expandafter(\EmFi@temp)%
517         }%
518     }%
519     \EmFi@GlobalKey{EmFi}{\EmFi@key.value}%
520 \else\ifx\EmFi@type\EmFi@S@number
521     \define@key{EmFi}{\EmFi@key.value}{%
522         \EmFi@itemtrue
523         \expandafter\edef\csname EmFi@V@#1\endcsname{ ##1}%
524     }%
525     \EmFi@GlobalKey{EmFi}{\EmFi@key.value}%
526 \fi\fi\fi
527 \define@key{EmFi}{\EmFi@key.prefix}{%
528     \EmFi@itemtrue
529     \expandafter\def\csname EmFi@P@#1\endcsname{##1}%
530 }%
531 \EmFi@GlobalKey{EmFi}{\EmFi@key.prefix}%
532 \define@key{EmFiSo}{\EmFi@key}[ascending]{%
533     \edef\EmFi@temp{##1}%
534     \ifx\EmFi@temp\EmFi@S@ascending
535         \def\EmFi@temp{true}%
536     \else\ifx\EmFi@temp\EmFi@S@descending
537         \def\EmFi@temp{false}%
538     \else
539         \def\EmFi@temp{}%
540     \EmFi@Error{%
541         Unknown sort order '\EmFi@temp'.\MessageBreak
542         Supported values: '\EmFi@S@ascending', %
543         '\EmFi@S@descending
544     }\@ehc
545 \fi\fi

```



```

546         \ifx\EmFi@temp\empty
547     \else
548         \xdef\EmFi@sortkeys{%
549             \EmFi@sortkeys
550             /\pdf@escapename{#1}%
551         }%
552     \ifx\EmFi@sortorders\empty
553         \global\let\EmFi@sortorders\EmFi@temp
554     \gdef\EmFi@sortcase{1}%
555     \else
556         \xdef\EmFi@sortorders{%
557             \EmFi@sortorders
558             \space
559             \EmFi@temp
560         }%
561     \xdef\EmFi@sortcase{2}%
562 \fi
563 \fi
564 }%
565 \EmFi@GlobalDefaultKey{EmFiSo}\EmFi@key
566 \endgroup
567 \else
568     \EmFi@Error{%
569         Field '\EmFi@key' is already defined%
570     }\@ehc
571 \fi
572 \fi
573 }

574 \define@key{EmFiFi}{type}{%
575     \EdefSanitize\EmFi@temp{#1}%
576     \ifx\EmFi@temp\EmFi@S@text
577         \let\EmFi@type\EmFi@temp
578     \else\ifx\EmFi@temp\EmFi@S@date
579         \let\EmFi@type\EmFi@temp
580     \else\ifx\EmFi@temp\EmFi@S@number
581         \let\EmFi@type\EmFi@temp
582     \else\ifx\EmFi@temp\EmFi@S@file
583         \let\EmFi@type\EmFi@temp
584     \else\ifx\EmFi@temp\EmFi@S@desc
585         \let\EmFi@type\EmFi@temp
586     \else\ifx\EmFi@temp\EmFi@S@moddate
587         \let\EmFi@type\EmFi@temp
588     \else\ifx\EmFi@temp\EmFi@S@creationdate
589         \let\EmFi@type\EmFi@temp
590     \else\ifx\EmFi@temp\EmFi@S@size
591         \let\EmFi@type\EmFi@temp
592     \else
593         \EmFi@Error{%
594             Unknown type '\EmFi@temp'.\MessageBreak
595             Supported types: 'text', 'date', 'number', 'file',\MessageBreak
596             'desc', 'moddate', 'creationdate', 'size'%
597         }%
598     \fi\fi\fi\fi\fi\fi\fi\fi
599 }

600 \define@key{EmFiFi}{title}{%
601     \def\EmFi@title{#1}%
602 }

\EmFi@setboolean

603 \def\EmFi@setboolean#1#2{%
604     \EdefSanitize\EmFi@temp{#2}%

```

```

605 \ifx\EmFi@temp\EmFi@S@true
606 \csname EmFi@#1true\endcsname
607 \else
608 \ifx\EmFi@temp\EmFi@S@false
609 \csname EmFi@#1false\endcsname
610 \else
611 \EmFi@Error{%
612     Unknown value '\EmFi@temp' for key '#1'.\MessageBreak
613     Supported values: 'true', 'false'%
614 } \@ehc
615 \fi
616 \fi
617 }

618 \define@key{EmFiFi}{visible}[true]{%
619 \EmFi@setboolean{visible}{#1}%
620 }

621 \define@key{EmFiFi}{edit}[true]{%
622 \EmFi@setboolean{edit}{#1}%
623 }

```

\EmFi@sortkeys

```
624 \def\EmFi@sortkeys{}
```

\EmFi@sortorders

```
625 \def\EmFi@sortorders{}
```

\embedfilesort

```

626 \def\embedfilesort{%
627 \setkeys{EmFiSo}%
628 }

```

2.8 Embed the file

\embedfile

```

629 \def\embedfile{%
630 \@ifnextchar[\EmFi@embedfile{\EmFi@embedfile[]}%
631 }

```

\EmFi@embedfile

```

632 \def\EmFi@embedfile[#1]#2{%
633 \ifEmFi@finished
634 \EmFi@Error{%
635 \string\embedfile\space after \string\embedfilefinish
636 }{%
637 The list of embedded files is already written.%
638 }%
639 \else
640 \begingroup
641 \def\EmFi@file{#2}%
642 \ifx\EmFi@file\EmFi@initialfile
643 \global\EmFi@initialfiletrue
644 \fi
645 \setkeys{EmFi}{#1}%
646 \expandafter\expandafter\expandafter
647 \ifx\expandafter\expandafter\expandafter
648 \\\pdf@filesize{\EmFi@file}\\%
649 \EmFi@Error{%
650 File '\EmFi@file' not found%
651 }{%
652 The unknown file is not embedded.%

```

```

653     }%
654 \else
655     \EmFi@convert\EmFi@filespec\EmFi@@filespec
656     \ifx\EmFi@desc\empty
657         \let\EmFi@@desc\empty
658     \else
659         \EmFi@convert\EmFi@desc\EmFi@@desc
660     \fi
661     \ifEmFi@item
662         \let\do\EmFi@do
663         \immediate\pdfobj{%
664             <<%
665                 \EmFi@fieldlist
666             >>%
667         }%
668         \edef\EmFi@ci{\the\pdflastobj}%
669     \fi
670     \immediate\pdfobj stream attr{%
671         /Type/EmbeddedFile%
672         \ifx\EmFi@mimetype\empty
673             \else
674                 /Subtype/\pdf@escapename{\EmFi@mimetype}%
675             \fi
676         /Params<<%
677             /ModDate(\pdf@filemoddate{\EmFi@file})%
678             /Size \pdf@filesize{\EmFi@file}%
679             /Checksum<\pdf@filemdfivesum{\EmFi@file}>%
680         >>%
681     }file{\EmFi@file}\relax
682     \EmFi@defobj{EmbeddedFile}%
683     \immediate\pdfobj{%
684         <<%
685             /Type/Filespec%
686             \ifx\EmFi@filesystem\empty
687                 \else
688                 /FS/\pdf@escapename{\EmFi@filesystem}%
689             \fi
690             /F(\EmFi@@filespec)%
691             \ifx\EmFi@@desc\empty
692                 \else
693                 /Desc(\EmFi@@desc)%
694             \fi
695             /EF<<%
696                 /F \the\pdflastobj\space 0 R%
697             >>%
698             \ifEmFi@item
699                 /CI \EmFi@ci\space 0 R%
700             \fi
701         >>%
702     }%
703     \EmFi@defobj{Filespec}%
704     \EmFi@add{%
705         \EmFi@@filespec
706     }{\the\pdflastobj\space 0 R}%
707 \fi
708 \endgroup
709 \fi
710 }

\EmFi@do
711 \def\EmFi@do#1{%
712     \expandafter\ifx\csname EmFi@P@#1\endcsname\relax
713         \expandafter\ifx\csname EmFi@V@#1\endcsname\relax

```

```

714     \else
715         /\pdf@escapename{#1}\csname EmFi@V@#1\endcsname
716     \fi
717 \else
718     /\pdf@escapename{#1}<<%
719     \expandafter\ifx\csname EmFi@V@#1\endcsname\relax
720     \else
721         /D\csname EmFi@V@#1\endcsname
722     \fi
723     /P(\csname EmFi@P@#1\endcsname)%
724     >>%
725 \fi
726 }

```

\EmFi@convert

```

727 \def\EmFi@convert#1#2{%
728     \ifnum\pdf@strcmp{\EmFi@stringmethod}{psd}=0 %
729         \pdfstringdef\EmFi@temp{#1}%
730         \let#2\EmFi@temp
731     \else
732         \edef#2{\pdf@escapestring{#1}}%
733     \fi
734 }

```

```

735 \global\let\EmFi@list\empty

```

\EmFi@add Sorting is done by the insertion sort algorithm. Probably the sorting could be done more reliable. However, the PDF specification is not too clear to me regarding precise sorting rules (how to deal with different encodings, escaped characters, ...).

```

736 \def\EmFi@add#1#2{%
737     \begingroup
738     \edef\key{\pdf@escapehex{#1}}%
739     \ifx\EmFi@list\empty
740         \xdef\EmFi@list{\noexpand\do{\key}{#2}}%
741     \else
742         \def\do###1##2{%
743             \ifnum\pdf@strcmp{##1}{\key}>0 %
744                 \edef\x{%
745                     \toks@{%
746                         \the\toks@%
747                         \noexpand\do{\key}{#2}%
748                         \noexpand\do{##1}{##2}%
749                     }%
750                 }%
751                 \x
752             \def\do####1####2{%
753                 \toks@\expandafter{\the\toks@\do{####1}{####2}}%
754             }%
755             \def\stop{%
756                 \xdef\EmFi@list{\the\toks@}%
757             }%
758         \else
759             \toks@\expandafter{\the\toks@\do{##1}{##2}}%
760         \fi
761     }%
762     \def\stop{%
763         \xdef\EmFi@list{\the\toks@\noexpand\do{\key}{#2}}%
764     }%
765     \toks@{}%
766     \EmFi@list\stop
767 \fi

```

```

768 \endgroup
769 }

\embedfilefinish

770 \def\embedfilefinish{%
771   \ifEmFi@finished
772     \EmFi@Error{%
773       Too many invocations of \string\embedfilefinish
774     }{%
775       The list of embedded files is already written.%
776     }%
777   \else
778     \ifx\EmFi@list\empty
779     \else
Write /EmbeddedFiles entry.
780       \global\EmFi@finishedtrue
781       \begingroup
782         \def\do##1##2{%
783           <##1>##2%
784         }%
785         \immediate\pdfobj{%
786           <<%
787             /Names[\EmFi@list]%
788           >>%
789         }%
790         \pdfnames{%
791           /EmbeddedFiles \the\pdflastobj\space 0 R%
792         }%
793       \endgroup
Write collection objects.
794       \ifEmFi@initialfile
795         \EmFi@collectiontrue
796       \fi
797       \ifEmFi@collection
798         \ifEmFi@initialfile
799         \else
800           \ifx\EmFi@initialfile\empty
801             \EmFi@convert\EmFi@initialfile\EmFi@initialfile
802           \else
803             \@PackageWarningNoLine{embedfile}{%
804               Missing initial file '\EmFi@initialfile'\MessageBreak
805               among the embedded files%
806             }%
807             \EmFi@initialfilefalse
808           \fi
809         \fi
810         \ifcase\EmFi@sortcase
811           \def\EmFi@temp{%
812         \or
813           \def\EmFi@temp{%
814             /S\EmFi@sortkeys
815             /A \EmFi@sortorders
816           }%
817         \else
818           \def\EmFi@temp{%
819             /S[\EmFi@sortkeys]%
820             /A[\EmFi@sortorders]%
821           }%
822         \fi
823         \def\EmFi@@order##1{%
824           \ifnum\EmFi@order>1 %

```

```

825         /O ##1%
826     \fi
827 }%
828 \immediate\pdfobj{
829     <<%
830         \ifx\EmFi@schema\empty
831         \else
832             /Schema<<\EmFi@schema>>%
833         \fi
834         \ifEmFi@initialfile
835             /D(\EmFi@initialfile)%
836         \fi
837         \ifx\EmFi@view\EmFi@S@tile
838             /View/T%
839         \else\ifx\EmFi@view\EmFi@S@hidden
840             /View/H%
841         \fi\fi
842         \ifx\EmFi@temp\empty
843             \EmFi@temp
844         \else
845             /Sort<<\EmFi@temp>>%
846         \fi
847     >>%
848 }%
849 \pdfcatalog{
850     /Collection \the\pdflastobj\space0 R%
851 }%
852 \fi
853 \fi
854 \fi
855 }

856 \begingroup\expandafter\expandafter\expandafter\endgroup
857 \expandafter\ifx\csname AtEndDocument\endcsname\relax
858 \else
859     \AtEndDocument{\embedfilefinish}%
860 \fi

861 \EmFi@AtEnd
862 </package>

```

3 Test

3.1 Catcode checks for loading

```

863 <test1>

864 \catcode'\{=1 %
865 \catcode'\}=2 %
866 \catcode'\#=6 %
867 \catcode'\@=11 %
868 \expandafter\ifx\csname count@\endcsname\relax
869     \countdef\count@=255 %
870 \fi
871 \expandafter\ifx\csname @gobble\endcsname\relax
872     \long\def\@gobble#1{}%
873 \fi
874 \expandafter\ifx\csname @firstofone\endcsname\relax
875     \long\def\@firstofone#1{#1}%
876 \fi
877 \expandafter\ifx\csname loop\endcsname\relax
878     \expandafter\@firstofone
879 \else

```

```

880 \expandafter\@gobble
881 \fi
882 {%
883 \def\loop#1\repeat{%
884 \def\body{#1}%
885 \iterate
886 }%
887 \def\iterate{%
888 \body
889 \let\next\iterate
890 \else
891 \let\next\relax
892 \fi
893 \next
894 }%
895 \let\repeat=\fi
896 }%
897 \def\RestoreCatcodes{}
898 \count@=0 %
899 \loop
900 \edef\RestoreCatcodes{%
901 \RestoreCatcodes
902 \catcode\the\count@=\the\catcode\count@\relax
903 }%
904 \ifnum\count@<255 %
905 \advance\count@ 1 %
906 \repeat
907
908 \def\RangeCatcodeInvalid#1#2{%
909 \count@=#1\relax
910 \loop
911 \catcode\count@=15 %
912 \ifnum\count@<#2\relax
913 \advance\count@ 1 %
914 \repeat
915 }
916 \expandafter\ifx\csname LoadCommand\endcsname\relax
917 \def\LoadCommand{\input embedfile.sty\relax}%
918 \fi
919 \def\Test{%
920 \RangeCatcodeInvalid{0}{47}%
921 \RangeCatcodeInvalid{58}{64}%
922 \RangeCatcodeInvalid{91}{96}%
923 \RangeCatcodeInvalid{123}{255}%
924 \catcode'\@=12 %
925 \catcode'\=0 %
926 \catcode'\{=1 %
927 \catcode'\}=2 %
928 \catcode'\#=6 %
929 \catcode'\[=12 %
930 \catcode'\]=12 %
931 \catcode'\%=14 %
932 \catcode'\ =10 %
933 \catcode13=5 %
934 \LoadCommand
935 \RestoreCatcodes
936 }
937 \Test
938 \csname @@end\endcsname
939 \end
940 </test1>

```

3.2 Simple test

```
941 <*test2>
942 \input embedfile.sty\relax
943 \embedfile[%
944   stringmethod=escape,%
945   mimetype=plain/text,%
946   desc={LaTeX docstrip source archive for package 'embedfile'},%
947   id={embedfile.dtx}%
948 ]{embedfile.dtx}
949 \nopagenumbers
950 Test (plain-TeX): {\tt embedfile.dtx} should be embedded.%
951
952 \def\Test#1{%
953   \par
954   \embedfileifobjectexists{embedfile.dtx}{#1}{%
955     Object #1 (embedfile.dtx): %
956     \embedfilegetobject{embedfile.dtx}{#1}%
957   }{%
958     \errmessage{Missing object #1 (embedfile.dtx)}%
959   }%
960 }
961 \Test{EmbeddedFile}
962 \Test{Filespec}
963 \embedfilefinish
964 \bye
965 </test2>

966 <*test3>
967 \NeedsTeXFormat{LaTeX2e}
968 \let\SavedJobname\jobname
969 \def\jobname{embedfile}
970 \RequirePackage{dtx-attach}[2007/11/25]
971 \let\jobname\SavedJobname
972 \documentclass{minimal}
973 \begin{document}
974   Test (\LaTeX): \texttt{embedfile.dtx} should be embedded.%
975 \end{document}
976 </test3>
```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

[CTAN:macros/latex/contrib/oberdiek/embedfile.dtx](http://ctan.org/macros/latex/contrib/oberdiek/embedfile.dtx) The source file.

[CTAN:macros/latex/contrib/oberdiek/embedfile.pdf](http://ctan.org/macros/latex/contrib/oberdiek/embedfile.pdf) Documentation.

Bundle. All the packages of the bundle ‘oberdiek’ are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

[CTAN:install/macros/latex/contrib/oberdiek.tds.zip](http://ctan.org/install/macros/latex/contrib/oberdiek.tds.zip)

TDS refers to the standard “A Directory Structure for \TeX Files” ([CTAN:tds/tds.pdf](http://ctan.org/tds/tds.pdf)). Directories with `texmf` in their name are usually organized this way.

4.2 Bundle installation

Unpacking. Unpack the `oberdiek.tds.zip` in the TDS tree (also known as `texmf` tree) of your choice. Example (linux):

¹[ftp://ftp.ctan.org/tex-archive/](http://ftp.ctan.org/tex-archive/)


```
unzip oberdiek.tds.zip -d ~/texmf
```

Script installation. Check the directory `TDS:scripts/oberdiek/` for scripts that need further installation steps. Package `attachfile2` comes with the Perl script `pdfatfi.pl` that should be installed in such a way that it can be called as `pdfatfi`. Example (linux):

```
chmod +x scripts/oberdiek/pdfatfi.pl
cp scripts/oberdiek/pdfatfi.pl /usr/local/bin/
```

4.3 Package installation

Unpacking. The `.dtx` file is a self-extracting `docstrip` archive. The files are extracted by running the `.dtx` through plain-`TEX`:

```
tex embedfile.dtx
```

TDS. Now the different files must be moved into the different directories in your installation TDS tree (also known as `texmf` tree):

<code>embedfile.sty</code>	→ <code>tex/latex/oberdiek/embedfile.sty</code>
<code>dtx-attach.sty</code>	→ <code>tex/latex/oberdiek/dtx-attach.sty</code>
<code>embedfile.pdf</code>	→ <code>doc/latex/oberdiek/embedfile.pdf</code>
<code>embedfile-example-plain.tex</code>	→ <code>doc/latex/oberdiek/embedfile-example-plain.tex</code>
<code>embedfile-example-collection.tex</code>	→ <code>doc/latex/oberdiek/embedfile-example-collection.tex</code>
<code>test/embedfile-test1.tex</code>	→ <code>doc/latex/oberdiek/test/embedfile-test1.tex</code>
<code>test/embedfile-test2.tex</code>	→ <code>doc/latex/oberdiek/test/embedfile-test2.tex</code>
<code>test/embedfile-test3.tex</code>	→ <code>doc/latex/oberdiek/test/embedfile-test3.tex</code>
<code>embedfile.dtx</code>	→ <code>source/latex/oberdiek/embedfile.dtx</code>

If you have a `docstrip.cfg` that configures and enables `docstrip`'s TDS installing feature, then some files can already be in the right place, see the documentation of `docstrip`.

4.4 Refresh file name databases

If your `TEX` distribution (te`TEX`, mi`TEX`, ...) relies on file name databases, you must refresh these. For example, te`TEX` users run `texhash` or `mktextlsr`.

4.5 Some details for the interested

Attached source. The PDF documentation on CTAN also includes the `.dtx` source file. It can be extracted by AcrobatReader 6 or higher. Another option is `pdftk`, e.g. unpack the file into the current directory:

```
pdftk embedfile.pdf unpack_files output .
```

Unpacking with L^AT_EX. The `.dtx` chooses its action depending on the format:

plain-`TEX`: Run `docstrip` and extract the files.

L^AT_EX: Generate the documentation.

If you insist on using L^AT_EX for `docstrip` (really, `docstrip` does not need L^AT_EX), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{embedfile.dtx}
```

Do not forget to quote the argument according to the demands of your shell.

Generating the documentation. You can use both the `.dtx` or the `.drv` to generate the documentation. The process can be configured by the configuration file `ltxdoc.cfg`. For instance, put this line into this file, if you want to have A4 as paper format:

```
\PassOptionsToClass{a4paper}{article}
```

An example follows how to generate the documentation with pdfL^AT_EX:

```
pdflatex embedfile.dtx
makeindex -s gind.ist embedfile.idx
pdflatex embedfile.dtx
makeindex -s gind.ist embedfile.idx
pdflatex embedfile.dtx
```

5 References

- [1] Scott Pakin: *The attachfile package*; 2005/02/20 v1.2; [CTAN:macros/latex/contrib/attachfile/](http://ctan.org/ctan/ctan-macros/latex/contrib/attachfile/).
- [2] Heiko Oberdiek: *The attachfile2 package*; 2006/08/16 v2.2; [CTAN:macros/latex/contrib/oberdiek/attachfile2.pdf](http://ctan.org/ctan/ctan-macros/latex/contrib/oberdiek/attachfile2.pdf).
- [3] Adobe Systems Incorporated: *PDF Reference, Sixth Edition, Version 1.7*, Oktober 2006; http://www.adobe.com/devnet/pdf/pdf_reference.html.
- [4] Network Working Group: RFC 2046, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*, November 1996; <http://www.rfc-editor.org/>.
- [5] IANA (Internet Assigned Numbers Authority): *MIME Media Types*, May 2006; <http://www.iana.org/assignments/media-types/>.

6 History

[2006/08/16 v1.0]

- First public version.

[2007/04/11 v1.1]

- Line ends sanitized.

[2007/09/09 v1.2]

- Fixes for plain-TeX, wrapper for package `keyval` added.
- Catcode section rewritten.

[2007/10/28 v2.0]

- Collection support added (PDF 1.7).

[2007/10/29 v2.1]

- Export of object references by adding new option `id` and new macros `\embedfileifobjectexists` and `\embedfilegetobject`.

[2007/11/11 v2.2]

- Use of package `pdftexcmds` for L^AT_EX support.

- Fix in use of `\pdf@filesize`, bug introduced in previous version.

7 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>	866, 928	<code>\csname</code>	118, 131, 155, 168, 175, 208, 233, 244, 269, 273, 284, 285, 286, 296, 324, 326, 333, 334, 344, 345, 350, 351, 355, 357, 380, 386, 392, 395, 400, 458, 503, 514, 523, 529, 606, 609, 712, 713, 715, 719, 721, 723, 857, 868, 871, 874, 877, 916, 938
<code>\%</code>	931		
<code>\:</code>	316, 317, 318, 319		
<code>\@</code>	867, 924		
<code>\@PackageError</code>	217, 294		
<code>\@PackageWarningNoLine</code>	803		
<code>\@ehc</code>	294, 423, 544, 570, 614		
<code>\@firstofone</code>	875, 878		
<code>\@firstoftwo</code>	389, <u>392</u>		
<code>\@gobble</code>	872, 880		
<code>\@ifnch</code>	301, 303, 320		
<code>\@ifnextchar</code>	<u>296</u> , 630		
<code>\@let@token</code>	301, 304, 307, 320		
<code>\@namedef</code>	<u>324</u>		
<code>\@ne</code>	274, 282, 331		
<code>\@secondoftwo</code>	387, <u>395</u>		
<code>\@sptoken</code>	304, 316, 317		
<code>\@undefined</code>	364		
<code>\@xifnch</code>	305, 318		
<code>\[</code>	929		
<code>\]</code>	648, 925		
<code>\{</code>	864, 926		
<code>\}</code>	865, 927		
<code>\]</code>	930		
<code>_</code>	932		
A		D	
<code>\advance</code>	461, 905, 913	<code>\define@key</code>	354, 374, 405, 499, 510, 521, 527, 532, 574, 600, 618, 621
<code>\AtEndDocument</code>	859	<code>\do</code>	493, 496, 662, 740, 742, 747, 748, 752, 753, 759, 763, 782
B		<code>\documentclass</code>	41, 972
<code>\begin</code>	75, 973		
<code>\body</code>	884, 888		
<code>\bye</code>	35, 964		
C		E	
<code>\catcode</code>	113, 114, 115, 116, 117, 128, 129, 130, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 172, 173, 176, 177, 178, 179, 183, 184, 185, 186, 190, 192, 864, 865, 866, 867, 902, 911, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933	<code>\EdefSanitize</code>	244, 406, 457, 523, 533, 575, 604
<code>\chardef</code>	270, 274	<code>\embedfile</code>	3, 16, 21, 25, 79, 85, 91, 105, <u>629</u> , 635, 943
<code>\count@</code>	460, 461, 462, 869, 898, 902, 904, 905, 909, 911, 912, 913	<code>\embedfilefield</code>	4, 50, 54, 58, 62, 66, 432, <u>448</u>
<code>\countdef</code>	869	<code>\embedfilefinish</code>	3, 34, 432, 451, 635, <u>770</u> , 859, 963
		<code>\embedfilegetobject</code>	5, <u>398</u> , 956
		<code>\embedfileifobjectexists</code>	5, <u>385</u> , 399, 954
		<code>\embedfilessetup</code>	3, 4, 11, 46, <u>428</u>
		<code>\embedfilesort</code>	5, 71, <u>626</u>
		<code>\EmFi@@desc</code>	657, 659, 691, 693
		<code>\EmFi@@filespec</code>	655, 690, 705
		<code>\EmFi@@order</code>	<u>445</u> , 483, 823
		<code>\EmFi@cadd</code>	704, <u>736</u>
		<code>\EmFi@AtEnd</code>	188, 189, 228, 239, 861
		<code>\EmFi@ci</code>	668, 699
		<code>\EmFi@collectiontrue</code>	408, 456, 795
		<code>\EmFi@convert</code>	468, 502, 513, 655, 659, <u>727</u> , 801
		<code>\EmFi@DefineKey</code>	353, 359, 360, 361, 362, 363, 427
		<code>\EmFi@defobj</code>	<u>378</u> , 682, 703
		<code>\EmFi@desc</code>	656, 659
		<code>\EmFi@details</code>	<u>246</u>
		<code>\EmFi@do</code>	662, <u>711</u>
		<code>\EmFi@editfalse</code>	466
		<code>\EmFi@embedfile</code>	630, <u>632</u>
		<code>\EmFi@Error</code>	215, 222, 234, 420, 431, 450, 540, 568, 593, 611, 634, 649, 772
		<code>\EmFi@fieldlist</code>	<u>446</u> , 494, 495, 665

<code>\EmFi@file</code>	360, 641, 642, 648, 650, 677, 678, 679, 681	594, 604, 605, 608, 612, 729, 730, 811, 813, 818, 842, 843, 845	
<code>\EmFi@filespec</code>	655	<code>\EmFi@tile</code>	247
<code>\EmFi@filesystem</code>	686, 688	<code>\EmFi@title</code>	463, 468, 482, 601
<code>\EmFi@finishedtrue</code>	780	<code>\EmFi@type</code>	
<code>\EmFi@GlobalDefaultKey</code>	347, 565		464, 473, 474, 475, 476, 477,
<code>\EmFi@GlobalKey</code>			478, 479, 498, 509, 520, 577,
	343, 348, 508, 519, 525, 531		579, 581, 583, 585, 587, 589, 591
<code>\EmFi@hidden</code>	248	<code>\EmFi@view</code>	411, 413, 415, 417, 837, 839
<code>\EmFi@id</code>	375, 380	<code>\EmFi@visibletrue</code>	465
<code>\EmFi@idtrue</code>	376	<code>\empty</code>	122, 410,
<code>\EmFi@initialfile</code>			546, 552, 656, 657, 672, 686,
	642, 800, 801, 804, 835		691, 735, 739, 778, 800, 830, 842
<code>\EmFi@initialfilefalse</code>	807	<code>\end</code>	97, 939, 975
<code>\EmFi@initialfiletrue</code>	643	<code>\endcsname</code>	
<code>\EmFi@itemtrue</code>	500, 511, 522, 528		118, 131, 155, 168, 175, 208,
<code>\EmFi@key</code>	457, 458, 463, 471, 496, 499, 508, 510, 519, 521, 525, 527, 531, 532, 565, 569		233, 244, 269, 273, 284, 285, 286, 296, 324, 326, 333, 334, 344, 345, 350, 351, 355, 357,
<code>\EmFi@list</code>	735, 739, 740, 756, 763, 766, 778, 787		380, 386, 392, 395, 400, 458, 504, 515, 523, 529, 606, 609, 712, 713, 715, 719, 721, 723,
<code>\EmFi@mimetype</code>	672, 674		857, 868, 871, 874, 877, 916, 938
<code>\EmFi@next</code>	407, 419, 425, 430, 437, 441	<code>\endinput</code>	140, 229, 240
<code>\EmFi@order</code> ...	444, 460, 462, 483, 824	<code>\errmessage</code>	958
<code>\EmFi@plain</code>	270, 274, 282, 331		
<code>\EmFi@RequirePackage</code>		F	
	207, 215, 219, 231, 242, 330	<code>\futurelet</code>	301, 320
<code>\EmFi@S@ascending</code>	257, 534, 542	G	
<code>\EmFi@S@creationdate</code> ..	255, 478, 588	<code>\gdef</code>	319, 444, 554
<code>\EmFi@S@date</code>	250, 473, 509, 578	<code>\Gin@driver</code>	5
<code>\EmFi@S@desc</code>	253, 476, 584	I	
<code>\EmFi@S@descending</code>	258, 536, 543	<code>\ifcase</code>	119, 810
<code>\EmFi@S@details</code>	411, 412, 413	<code>\ifEmFi@collection</code>	261, 797
<code>\EmFi@S@false</code>	260, 608	<code>\ifEmFi@edit</code>	265, 488
<code>\EmFi@S@file</code>	252, 475, 582	<code>\ifEmFi@finished</code>	267, 429, 449, 633, 771
<code>\EmFi@S@hidden</code>	416, 417, 839	<code>\ifEmFi@id</code>	268, 379
<code>\EmFi@S@moddate</code>	254, 477, 586	<code>\ifEmFi@initialfile</code>	262, 794, 798, 834
<code>\EmFi@S@number</code>	251, 474, 520, 580	<code>\ifEmFi@item</code>	266, 661, 698
<code>\EmFi@S@size</code>	256, 479, 590	<code>\ifEmFi@sort</code>	263
<code>\EmFi@S@text</code>	249, 464, 498, 576	<code>\ifEmFi@visible</code>	264, 484
<code>\EmFi@S@tile</code>	414, 415, 837	<code>\ifnum</code>	282, 331, 728, 743, 824, 904, 912
<code>\EmFi@S@true</code>	259, 605	<code>\ifpdf</code>	220
<code>\EmFi@schema</code> ..	443, 469, 470, 830, 832	<code>\ifx</code>	120, 122,
<code>\EmFi@setboolean</code>	603, 619, 622		131, 155, 163, 208, 233, 269,
<code>\EmFi@sortcase</code>	447, 554, 561, 810		273, 296, 304, 307, 324, 364,
<code>\EmFi@sortkeys</code>	548, 549, 624, 814, 819		367, 386, 392, 395, 410, 412,
<code>\EmFi@sortorders</code>			414, 416, 458, 473, 474, 475,
	552, 553, 556, 557, 625, 815, 820		476, 477, 478, 479, 498, 509,
<code>\EmFi@stringmethod</code>	728		520, 534, 536, 546, 552, 576,
<code>\EmFi@temp</code>	243, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 271, 277, 278, 279, 280, 281, 283, 288, 289, 290, 291, 292, 332, 336, 337, 338, 339, 340, 406, 410, 412, 414, 416, 421, 501, 502, 505, 512, 513, 516, 533, 534, 535, 536, 537, 539, 541, 546, 553, 559, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591,		578, 580, 582, 584, 586, 588, 590, 605, 608, 642, 647, 656, 672, 686, 691, 712, 713, 719, 739, 778, 800, 830, 837, 839, 842, 857, 868, 871, 874, 877, 916
		<code>\immediate</code>	
			133, 157, 663, 670, 683, 785, 828
		<code>\input</code>	4, 6, 7, 210, 917, 942
		<code>\iterate</code>	885, 887, 889
		J	
		<code>\jobname</code>	95, 103, 108, 109, 968, 969, 971

K	
\key	738, 740, 743, 747, 763
\KV@errx	293
L	
\LaTeX	974
\LoadCommand	917, 934
\loop	883, 899, 910
M	
\MessageBreak	
.....	421, 541, 594, 595, 612, 804
N	
\NeedsTeXFormat	40, 101, 967
\newif	261,
.....	262, 263, 264, 265, 266, 267, 268
\next	889, 891, 893
\nopagenumbers	949
P	
\PackageInfo	136
\par	953
\pdf@escapehex	738
\pdf@escapename	
.....	471, 550, 674, 688, 715, 718
\pdf@escapestring	732
\pdf@filemdfivesum	679
\pdf@filemoddate	677
\pdf@filesize	648, 678
\pdf@strcmp	728, 743
\pdfcatalog	849
\pdflastobj	381, 668, 696, 706, 791, 850
\pdfnames	790
\pdfobj	663, 670, 683, 785, 828
\pdfstringdef	43, 364, 367, 729
\ProvidesPackage	102, 169
R	
\RangeCatcodeInvalid	
.....	908, 920, 921, 922, 923
S	
\repeat	883, 895, 906, 914
\RequirePackage	104, 213, 970
\reserved@a	299, 308
\reserved@b	300, 310
\reserved@c	305, 308, 310, 313
\reserved@d	298, 307
\resetatcatcode	8
\RestoreCatcodes ..	897, 900, 901, 935
T	
\SavedJobname	968, 971
\setkeys	438, 467, 627, 645
\space	381, 432, 451,
.....	558, 635, 696, 699, 706, 791, 850
\stop	755, 762, 766
T	
\Test	919, 937, 952, 961, 962
\TeX	950
\texttt	974
\the	176, 177, 178, 179, 190,
.....	381, 462, 668, 696, 706, 746,
.....	753, 756, 759, 763, 791, 850, 902
\TMP@EnsureCode	187,
.....	194, 195, 196, 197, 198, 199,
.....	200, 201, 202, 203, 204, 205, 206
\toks@	745, 746, 753, 756, 759, 763, 765
\tt	950
U	
\usepackage	42, 45
W	
\write	133, 157
X	
\x	118, 120, 122, 132, 136, 138,
.....	156, 161, 168, 174, 182, 744, 751
Z	
\z@	270