

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-T _E X	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	25
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\@undefined
164 \xdef#1{#3}%
165 \fi
166 \ifx#1\relax
167 \xdef#1{#3}%
168 \fi
169 }%
170 \fi
171 \expandafter\x\csname ver@embedfile.sty\endcsname
172 \ProvidesPackage{embedfile}%
173 [2007/11/25 v2.3 embed files into PDF (H0)]

```

2.2 Catcodes

```

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

```

2.3 Tools

\EmFi@RequirePackage

```

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

```

\EmFi@Error

```

218 \EmFi@RequirePackage{infwarerr}[2007/09/09]%
219 \def\EmFi@Error{%
220   \@PackageError{embedfile}%
221 }

```

2.4 Check for recent pdf_T_EX in PDF mode

Load package ifpdf and check mode.

```

222 \EmFi@RequirePackage{ifpdf}[2007/09/09]
223 \ifpdf
224 \else
225   \EmFi@Error{%

```

```

226     Missing pdfTeX in PDF mode%
227 }{%
228     Currently other drivers are not supported. %
229     Package loading is aborted.%
230 }%
231 \EmFi@AtEnd
232 \expandafter\endinput
233 \fi

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

Check version.

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

```

2.5 Strings

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

```

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

246 \def\EmFi@temp#1{%
247   \expandafter\EdefSanitize\csname EmFi@S@#1\endcsname{#1}%
248 }

\EmFi@details

249 \EmFi@temp{details}%

\EmFi@tile

250 \EmFi@temp{tile}%

\EmFi@hidden

251 \EmFi@temp{hidden}%

\EmFi@S@text

252 \EmFi@temp{text}

\EmFi@S@date

253 \EmFi@temp{date}

\EmFi@S@number

254 \EmFi@temp{number}

\EmFi@S@file

255 \EmFi@temp{file}

\EmFi@S@desc

256 \EmFi@temp{desc}

\EmFi@S@moddate

257 \EmFi@temp{moddate}

\EmFi@S@creationdate

258 \EmFi@temp{creationdate}

```

```

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

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

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

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

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

```

2.6 Switches

```

\ifEmFi@collection
264 \newif\ifEmFi@collection

\ifEmFi@initialfile
265 \newif\ifEmFi@initialfile

\ifEmFi@sort
266 \newif\ifEmFi@sort

\ifEmFi@visible
267 \newif\ifEmFi@visible

\ifEmFi@edit
268 \newif\ifEmFi@edit

\ifEmFi@item
269 \newif\ifEmFi@item

\ifEmFi@finished
270 \newif\ifEmFi@finished

\ifEmFi@id
271 \newif\ifEmFi@id

```

2.7 Key value definitions

```

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

```

```

288             \csname#1\endcsname
289     \expandafter\def\csname#1\endcsname
290 }%
291 \EmFi@temp{NeedsTeXFormat}#1{}%
292 \EmFi@temp{ProvidesPackage}#1[#2]{}% hash-ok
293 \EmFi@temp{DeclareOption}#1{}%
294 \EmFi@temp{ExecuteOptions}#1{}%
295 \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.
296     \def\KV@errx#1{%
297         \@PackageError{keyval}{#1}\@ehc
298     }%

```

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

```

\@ifnextchar

299     \expandafter\ifx\csname @ifnextchar\endcsname\relax
300     \def\@ifnextchar#1#2#3{%
301         \let\reserved@d=#1%
302         \def\reserved@a{#2}%
303         \def\reserved@b{#3}%
304         \futurelet\@let@token\@ifnch
305     }%
306     \def\@ifnch{%
307         \ifx\@let@token\@sptoken
308             \let\reserved@c\@xifnch
309         \else
310             \ifx\@let@token\reserved@d
311                 \let\reserved@c\reserved@a
312             \else
313                 \let\reserved@c\reserved@b
314             \fi
315         \fi
316         \reserved@c
317     }%
318     \begingroup
319     \def\:{\global\let\@sptoken=}%
320     \: % this makes \@sptoken a space token
321     \def\:{\@xifnch}%
322     \expandafter\gdef\:{%
323         \futurelet\@let@token\@ifnch
324     }%
325     \endgroup
326     \fi

\@namedef

327     \expandafter\ifx\csname @namedef\endcsname\relax
328     \def\@namedef#1{%
329         \expandafter\def\csname#1\endcsname
330     }%
331     \fi

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

```

```

341 \EmFi@temp{DeclareOption}%
342 \EmFi@temp{ExecuteOptions}%
343 \EmFi@temp{ProcessOptions}%
344 \fi
345 \fi

```

\EmFi@GlobalKey

```

346 \def\EmFi@GlobalKey#1#2{%
347 \global\expandafter\let\csname KV@#1@#2\expandafter\endcsname
348 \csname KV@#1@#2\endcsname
349 }

```

\EmFi@GlobalDefaultKey

```

350 \def\EmFi@GlobalDefaultKey#1#2{%
351 \EmFi@GlobalKey{#1}{#2}%
352 \global\expandafter\let
353 \csname KV@#1@#2@default\expandafter\endcsname
354 \csname KV@#1@#2@default\endcsname
355 }

```

\EmFi@DefineKey

```

356 \def\EmFi@DefineKey#1#2{%
357 \define@key{EmFi}{#1}{%
358 \expandafter\def\csname EmFi@#1\endcsname{##1}%
359 }%
360 \expandafter\def\csname EmFi@#1\endcsname{#2}%
361 }

```

Subtype of the embedded file (optional).

```
362 \EmFi@DefineKey{mimetype}{}

```

File specification string.

```
363 \EmFi@DefineKey{filespec}{\EmFi@file}

```

File system (optional).

```
364 \EmFi@DefineKey{filesystem}{}

```

Description (optional).

```
365 \EmFi@DefineKey{desc}{}

```

Method for converting text to PDF strings.

```

366 \EmFi@DefineKey{stringmethod}{%
367 \ifx\pdfstringdef\@undefined
368 escape%
369 \else
370 \ifx\pdfstringdef\relax
371 escape%
372 \else
373 psd%
374 \fi
375 \fi
376 }

```

Option id as key for object numbers.

```

377 \define@key{EmFi}{id}{%
378 \def\EmFi@id{#1}%
379 \EmFi@idtrue
380 }

```

\EmFi@defobj

```

381 \def\EmFi@defobj#1{%
382 \ifEmFi@id
383 \expandafter\xdef\csname EmFi@#1@\EmFi@id\endcsname{%

```

```

384     \the\pdflastobj\space 0 R%
385   }%
386 \fi
387 }

```

\embedfileifobjectexists

```

388 \def\embedfileifobjectexists#1#2{%
389   \expandafter\ifx\csname EmFi@#2@#1\endcsname\relax
390     \expandafter\@secondoftwo
391   \else
392     \expandafter\@firstoftwo
393   \fi
394 }

```

\@firstoftwo

```

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

```

\@secondoftwo

```

398 \expandafter\ifx\csname @secondoftwo\endcsname\relax
399   \long\def\@secondoftwo#1#2{#2}%
400 \fi

```

\embedfilegetobject

```

401 \def\embedfilegetobject#1#2{%
402   \embedfileifobjectexists{#1}{#2}{%
403     \csname EmFi@#2@#1\endcsname
404   }{%
405     0 0 R%
406   }%
407 }

```

Initial view of the collection.

```

408 \define@key{EmFi}{view}[]{%
409   \EdefSanitize\EmFi@temp{#1}%
410   \def\EmFi@next{%
411     \global\EmFi@collectiontrue
412   }%
413   \ifx\EmFi@temp\empty
414     \let\EmFi@view\EmFi@S@details
415   \else\ifx\EmFi@temp\EmFi@S@details
416     \let\EmFi@view\EmFi@S@details
417   \else\ifx\EmFi@temp\EmFi@S@tile
418     \let\EmFi@view\EmFi@S@tile
419   \else\ifx\EmFi@temp\EmFi@S@hidden
420     \let\EmFi@view\EmFi@S@hidden
421   \else
422     \let\EmFi@next\relax
423     \EmFi@Error{%
424       Unknown value '\EmFi@temp' for key 'view'.\MessageBreak
425       Supported values: 'details', 'tile', 'hidden'.%
426     }\@ehc
427   \fi\fi\fi\fi
428   \EmFi@next
429 }
430 \EmFi@DefineKey{initialfile}{}

```

\embedfilessetup

```

431 \def\embedfilessetup{%
432   \ifEmFi@finished

```

```

433     \def\EmFi@next##1{%
434     \EmFi@Error{%
435         \string\embedfilefield\space after \string\embedfilefinish
436     }{%
437         The list of embedded files is already written.%
438     }%
439     \else
440     \def\EmFi@next{%
441         \setkeys{EmFi}%
442     }%
443     \fi
444     \EmFi@next
445 }

\EmFi@schema
446 \def\EmFi@schema{}

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

\EmFi@@order
448 \let\EmFi@@order\relax

\EmFi@fieldlist
449 \def\EmFi@fieldlist{}

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

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

```

```

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

```

```

544         Unknown sort order '\EmFi@temp'.\MessageBreak
545         Supported values: '\EmFi@S@ascending', %
546         '\EmFi@S@descending
547     }\@ehc
548 \fi\fi
549 \ifx\EmFi@temp\empty
550 \else
551     \xdef\EmFi@sortkeys{%
552         \EmFi@sortkeys
553         /\pdf@escapename{#1}%
554     }%
555     \ifx\EmFi@sortorders\empty
556         \global\let\EmFi@sortorders\EmFi@temp
557         \gdef\EmFi@sortcase{1}%
558     \else
559         \xdef\EmFi@sortorders{%
560             \EmFi@sortorders
561             \space
562             \EmFi@temp
563         }%
564         \xdef\EmFi@sortcase{2}%
565     \fi
566 \fi
567 }%
568 \EmFi@GlobalDefaultKey{EmFiSo}\EmFi@key
569 \endgroup
570 \else
571     \EmFi@Error{%
572         Field '\EmFi@key' is already defined%
573     }\@ehc
574 \fi
575 \fi
576 }

577 \define@key{EmFiFi}{type}{%
578     \EdefSanitize\EmFi@temp{#1}%
579     \ifx\EmFi@temp\EmFi@S@text
580         \let\EmFi@type\EmFi@temp
581     \else\ifx\EmFi@temp\EmFi@S@date
582         \let\EmFi@type\EmFi@temp
583     \else\ifx\EmFi@temp\EmFi@S@number
584         \let\EmFi@type\EmFi@temp
585     \else\ifx\EmFi@temp\EmFi@S@file
586         \let\EmFi@type\EmFi@temp
587     \else\ifx\EmFi@temp\EmFi@S@desc
588         \let\EmFi@type\EmFi@temp
589     \else\ifx\EmFi@temp\EmFi@S@moddate
590         \let\EmFi@type\EmFi@temp
591     \else\ifx\EmFi@temp\EmFi@S@creationdate
592         \let\EmFi@type\EmFi@temp
593     \else\ifx\EmFi@temp\EmFi@S@size
594         \let\EmFi@type\EmFi@temp
595     \else
596         \EmFi@Error{%
597             Unknown type '\EmFi@temp'.\MessageBreak
598             Supported types: 'text', 'date', 'number', 'file',\MessageBreak
599             'desc', 'moddate', 'creationdate', 'size'%
600         }%
601     \fi\fi\fi\fi\fi\fi\fi\fi
602 }

603 \define@key{EmFiFi}{title}{%
604     \def\EmFi@title{#1}%
605 }

```

```

\EmFi@setboolean
606 \def\EmFi@setboolean#1#2{%
607   \EdefSanitize\EmFi@temp{#2}%
608   \ifx\EmFi@temp\EmFi@S@true
609     \csname EmFi@#1true\endcsname
610   \else
611     \ifx\EmFi@temp\EmFi@S@false
612       \csname EmFi@#1false\endcsname
613     \else
614       \EmFi@Error{%
615         Unknown value ‘\EmFi@temp’ for key ‘#1’.\MessageBreak
616         Supported values: ‘true’, ‘false’%
617       }\@ehc
618     \fi
619   \fi
620 }

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

624 \define@key{EmFiFi}{edit}[true]{%
625   \EmFi@setboolean{edit}{#1}%
626 }

\EmFi@sortkeys
627 \def\EmFi@sortkeys{}

\EmFi@sortorders
628 \def\EmFi@sortorders{}

\embedfilesort
629 \def\embedfilesort{%
630   \setkeys{EmFiSo}%
631 }

```

2.8 Embed the file

```

\embedfile
632 \def\embedfile{%
633   \@ifnextchar[\EmFi@embedfile{\EmFi@embedfile[]}%
634 }

\EmFi@embedfile
635 \def\EmFi@embedfile[#1]#2{%
636   \ifEmFi@finished
637     \EmFi@Error{%
638       \string\embedfile\space after \string\embedfilefinish
639     }{%
640       The list of embedded files is already written.%
641     }%
642   \else
643     \begingroup
644     \def\EmFi@file{#2}%
645     \ifx\EmFi@file\EmFi@initialfile
646       \global\EmFi@initialfiletrue
647     \fi
648     \setkeys{EmFi}{#1}%
649     \expandafter\expandafter\expandafter
650     \ifx\expandafter\expandafter\expandafter
651       \\pdf@filesize{\EmFi@file}\\%

```

```

652     \EmFi@Error{%
653         File '\EmFi@file' not found%
654     }{%
655         The unknown file is not embedded.%
656     }%
657 \else
658     \EmFi@convert\EmFi@filespec\EmFi@@filespec
659     \ifx\EmFi@desc\empty
660         \let\EmFi@@desc\empty
661     \else
662         \EmFi@convert\EmFi@desc\EmFi@@desc
663     \fi
664     \ifEmFi@item
665         \let\do\EmFi@do
666         \immediate\pdfobj{%
667             <<%
668                 \EmFi@fieldlist
669             >>%
670         }%
671         \edef\EmFi@ci{\the\pdflastobj}%
672     \fi
673     \immediate\pdfobj stream attr{%
674         /Type/EmbeddedFile%
675         \ifx\EmFi@mimetype\empty
676         \else
677             /Subtype/\pdf@escapename{\EmFi@mimetype}%
678         \fi
679         /Params<<%
680             /ModDate(\pdf@filemoddate{\EmFi@file})%
681             /Size \pdf@filesize{\EmFi@file}%
682             /Checksum<\pdf@filemdfivesum{\EmFi@file}>%
683         >>%
684     }file{\EmFi@file}\relax
685     \EmFi@defobj{EmbeddedFile}%
686     \immediate\pdfobj{%
687         <<%
688             /Type/Filespec%
689             \ifx\EmFi@filesystem\empty
690             \else
691                 /FS/\pdf@escapename{\EmFi@filesystem}%
692             \fi
693             /F(\EmFi@@filespec)%
694             \ifx\EmFi@@desc\empty
695             \else
696                 /Desc(\EmFi@@desc)%
697             \fi
698             /EF<<%
699                 /F \the\pdflastobj\space 0 R%
700             >>%
701             \ifEmFi@item
702                 /CI \EmFi@ci\space 0 R%
703             \fi
704             >>%
705         }%
706     \EmFi@defobj{Filespec}%
707     \EmFi@add{%
708         \EmFi@@filespec
709     }{\the\pdflastobj\space 0 R}%
710 \fi
711 \endgroup
712 \fi
713 }

```

\EmFi@do

```
714 \def\EmFi@do#1{%
715   \expandafter\ifx\csname EmFi@P@#1\endcsname\relax
716     \expandafter\ifx\csname EmFi@V@#1\endcsname\relax
717       \else
718         /\pdf@escapename{#1}\csname EmFi@V@#1\endcsname
719       \fi
720     \else
721       /\pdf@escapename{#1}<<%
722       \expandafter\ifx\csname EmFi@V@#1\endcsname\relax
723         \else
724           /D\csname EmFi@V@#1\endcsname
725         \fi
726       /P(\csname EmFi@P@#1\endcsname)%
727     >>%
728   \fi
729 }
```

\EmFi@convert

```
730 \def\EmFi@convert#1#2{%
731   \ifnum\pdf@strcmp{\EmFi@stringmethod}{psd}=0 %
732     \pdfstringdef\EmFi@temp{#1}%
733     \let#2\EmFi@temp
734   \else
735     \edef#2{\pdf@escapestring{#1}}%
736   \fi
737 }
```

```
738 \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, ...).

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

```

767     }%
768     \toks@{ }%
769     \EmFi@list\stop
770     \fi
771 \endgroup
772 }

```

\embedfilefinish

```

773 \def\embedfilefinish{%
774   \ifEmFi@finished
775     \EmFi@Error{%
776       Too many invocations of \string\embedfilefinish
777     }{%
778       The list of embedded files is already written.%
779     }%
780   \else
781     \ifx\EmFi@list\empty
782     \else

```

Write /EmbeddedFiles entry.

```

783     \global\EmFi@finishedtrue
784     \begingroup
785     \def\do##1##2{%
786       <##1>##2%
787     }%
788     \immediate\pdfobj{%
789       <<%
790         /Names[\EmFi@list]%
791       >>%
792     }%
793     \pdfnames{%
794       /EmbeddedFiles \the\pdflastobj\space 0 R%
795     }%
796 \endgroup

```

Write collection objects.

```

797   \ifEmFi@initialfile
798     \EmFi@collectiontrue
799   \fi
800   \ifEmFi@collection
801     \ifEmFi@initialfile
802     \else
803       \ifx\EmFi@initialfile\empty
804         \EmFi@convert\EmFi@initialfile\EmFi@initialfile
805       \else
806         \@PackageWarningNoLine{embedfile}{%
807           Missing initial file '\EmFi@initialfile'\MessageBreak
808           among the embedded files%
809         }%
810         \EmFi@initialfilefalse
811       \fi
812     \fi
813     \ifcase\EmFi@sortcase
814       \def\EmFi@temp{ }%
815     \or
816       \def\EmFi@temp{%
817         /S\EmFi@sortkeys
818         /A \EmFi@sortorders
819       }%
820     \else
821       \def\EmFi@temp{%
822         /S[\EmFi@sortkeys]%
823         /A[\EmFi@sortorders]%

```

```

824     }%
825   \fi
826   \def\EmFi@@order##1{%
827     \ifnum\EmFi@order>1 %
828       /O ##1%
829     \fi
830   }%
831   \immediate\pdfobj{%
832     <<%
833     \ifx\EmFi@schema\empty
834       \else
835         /Schema<<\EmFi@schema>>%
836       \fi
837     \ifEmFi@initialfile
838       /D(\EmFi@initialfile)%
839     \fi
840     \ifx\EmFi@view\EmFi@S@tile
841       /View/T%
842     \else\ifx\EmFi@view\EmFi@S@hidden
843       /View/H%
844     \fi\fi
845     \ifx\EmFi@temp\empty
846       \EmFi@temp
847     \else
848       /Sort<<\EmFi@temp>>%
849     \fi
850     >>%
851   }%
852   \pdfcatalog{%
853     /Collection \the\pdflastobj\space0 R%
854   }%
855 \fi
856 \fi
857 \fi
858 }

859 \begingroup\expandafter\expandafter\expandafter\endgroup
860 \expandafter\ifx\csname AtEndDocument\endcsname\relax
861 \else
862   \AtEndDocument{\embedfilefinish}%
863 \fi

864 \EmFi@AtEnd
865 \end{package}

```

3 Test

3.1 Catcode checks for loading

```

866 \test1\

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

```

```

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

```

```

941 \csname @@end\endcsname
942 \end
943 </test1>

3.2 Simple test

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

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

```

4 Installation

4.1 Download

Package. This package is available on CTAN¹:

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

[CTAN: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](#)

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

¹<http://ftp.ctan.org/tex-archive/>

4.2 Bundle installation

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

```
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 , mik \TeX , ...) relies on file name databases, you must refresh these. For example, te \TeX users run `texhash` or `mktexlsr`.

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 \LaTeX . The `.dtx` chooses its action depending on the format:

plain- \TeX : Run `docstrip` and extract the files.

\LaTeX : Generate the documentation.

If you insist on using \LaTeX for `docstrip` (really, `docstrip` does not need \LaTeX), 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/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/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>	869, 931	<code>\csname</code> <i>118, 131, 155, 171, 178, 211,</i>	
<code>\%</code>	934	<i>236, 247, 272, 276, 287, 288,</i>	
<code>\:</code>	<i>319, 320, 321, 322</i>	<i>289, 299, 327, 329, 336, 337,</i>	
<code>\@</code>	870, 927	<i>347, 348, 353, 354, 358, 360,</i>	
<code>\@PackageError</code>	220, 297	<i>383, 389, 395, 398, 403, 461,</i>	
<code>\@PackageWarningNoLine</code>	806	<i>506, 517, 526, 532, 609, 612,</i>	
<code>\@ehc</code>	<i>297, 426, 547, 573, 617</i>	<i>715, 716, 718, 722, 724, 726,</i>	
<code>\@firstofone</code>	878, 881	<i>860, 871, 874, 877, 880, 919, 941</i>	
<code>\@firstoftwo</code>	<i>392, 395</i>		
<code>\@gobble</code>	875, 883	D	
<code>\@ifnch</code>	<i>304, 306, 323</i>	<code>\define@key</code> <i>357, 377, 408, 502, 513,</i>	
<code>\@ifnextchar</code>	<i>299, 633</i>	<i>524, 530, 535, 577, 603, 621, 624</i>	
<code>\@let@token</code>	<i>304, 307, 310, 323</i>	<code>\do</code>	<i>496, 499, 665, 743, 745,</i>
<code>\@namedef</code>	<i>327</i>	<i>750, 751, 755, 756, 762, 766, 785</i>	
<code>\@ne</code>	<i>277, 285, 334</i>	<code>\documentclass</code>	<i>41, 975</i>
<code>\@secondoftwo</code>	<i>390, 398</i>	E	
<code>\@sptoken</code>	<i>307, 319, 320</i>	<code>\EdefSanitize</code>	
<code>\@undefined</code>	<i>163, 367</i>	<i>247, 409, 460, 526, 536, 578, 607</i>	
<code>\@xifnch</code>	<i>308, 321</i>	<code>\embedfile</code>	<i>3, 16,</i>
<code>\[</code>	932	<i>21, 25, 79, 85, 91, 105, <u>632</u>, 638, 946</i>	
<code>\</code>	651, 928	<code>\embedfilefield</code>	
<code>\{</code>	867, 929	<i>... 4, 50, 54, 58, 62, 66, 435, <u>451</u></i>	
<code>\}</code>	868, 930	<code>\embedfilefinish</code>	<i>3,</i>
<code>\]</code>	933	<i>34, 435, 454, 638, <u>773</u>, 862, 966</i>	
 		<code>\embedfilegetobject</code>	<i>5, <u>401</u>, 959</i>
<code>_</code>	935	<code>\embedfileifobjectexists</code>	
A		<i>... 5, <u>388</u>, 402, 957</i>	
<code>\advance</code>	<i>464, 908, 916</i>	<code>\embedfilessetup</code>	<i>3, 4, 11, 46, <u>431</u></i>
<code>\AtEndDocument</code>	862	<code>\embedfilesort</code>	<i>5, 71, <u>629</u></i>
B		<code>\EmFi@@desc</code>	<i>660, 662, 694, 696</i>
<code>\begin</code>	<i>75, 976</i>	<code>\EmFi@@filespec</code>	<i>658, 693, 708</i>
<code>\body</code>	887, 891	<code>\EmFi@@order</code>	<i><u>448</u>, 486, 826</i>
<code>\bye</code>	<i>35, 967</i>	<code>\EmFi@cadd</code>	<i>707, <u>739</u></i>
C		<code>\EmFi@AtEnd</code> ...	<i>191, 192, 231, 242, 864</i>
<code>\catcode</code>	<i>113, 114, 115, 116,</i>	<code>\EmFi@ci</code>	<i>671, 702</i>
<i>117, 128, 129, 130, 144, 145,</i>		<code>\EmFi@collectiontrue</code> ..	<i>411, 459, 798</i>
<i>146, 147, 148, 149, 150, 151,</i>		<code>\EmFi@convert</code>	
<i>152, 153, 154, 175, 176, 179,</i>		<i>471, 505, 516, 658, 662, <u>730</u>, 804</i>	
<i>180, 181, 182, 186, 187, 188,</i>		<code>\EmFi@DefineKey</code>	
<i>189, 193, 195, 867, 868, 869,</i>		<i>356, 362, 363, 364, 365, 366, 430</i>	
<i>870, 905, 914, 927, 928, 929,</i>		<code>\EmFi@defobj</code>	<i><u>381</u>, 685, 706</i>
<i>930, 931, 932, 933, 934, 935, 936</i>		<code>\EmFi@desc</code>	<i>659, 662</i>
<code>\chardef</code>	<i>273, 277</i>	<code>\EmFi@details</code>	<i><u>249</u></i>
<code>\count@</code> ...	<i>463, 464, 465, 872, 901,</i>	<code>\EmFi@do</code>	<i>665, <u>714</u></i>
<i>905, 907, 908, 912, 914, 915, 916</i>		<code>\EmFi@editfalse</code>	<i>469</i>
<code>\countdef</code>	872	<code>\EmFi@embedfile</code>	<i>633, <u>635</u></i>
		<code>\EmFi@Error</code>	
		<i>218, 225, 237, 423, 434, 453,</i>	
		<i>543, 571, 596, 614, 637, 652, 775</i>	
		<code>\EmFi@fieldlist</code> ...	<i><u>449</u>, 497, 498, 668</i>

<code>\EmFi@file</code>	363, 644, 645, 651, 653, 680, 681, 682, 684	597, 607, 608, 611, 615, 732, 733, 814, 816, 821, 845, 846, 848	
<code>\EmFi@filespec</code>	658	<code>\EmFi@tile</code>	250
<code>\EmFi@filesystem</code>	689, 691	<code>\EmFi@title</code>	466, 471, 485, 604
<code>\EmFi@finishedtrue</code>	783	<code>\EmFi@type</code>	
<code>\EmFi@GlobalDefaultKey</code>	350, 568		467, 476, 477, 478, 479, 480,
<code>\EmFi@GlobalKey</code>			481, 482, 501, 512, 523, 580,
	346, 351, 511, 522, 528, 534		582, 584, 586, 588, 590, 592, 594
<code>\EmFi@hidden</code>	251	<code>\EmFi@view</code>	414, 416, 418, 420, 840, 842
<code>\EmFi@id</code>	378, 383	<code>\EmFi@visibletrue</code>	468
<code>\EmFi@idtrue</code>	379	<code>\empty</code>	122, 413,
<code>\EmFi@initialfile</code>			549, 555, 659, 660, 675, 689,
	645, 803, 804, 807, 838		694, 738, 742, 781, 803, 833, 845
<code>\EmFi@initialfilefalse</code>	810	<code>\end</code>	97, 942, 978
<code>\EmFi@initialfiletrue</code>	646	<code>\endcsname</code>	
<code>\EmFi@itemtrue</code>	503, 514, 525, 531		118, 131, 155, 171, 178, 211,
<code>\EmFi@key</code>	460, 461, 466, 474, 499, 502, 511, 513, 522, 524, 528, 530, 534, 535, 568, 572		236, 247, 272, 276, 287, 288,
			289, 299, 327, 329, 336, 337,
			347, 348, 353, 354, 358, 360,
<code>\EmFi@list</code>	738, 742, 743, 759, 766, 769, 781, 790		383, 389, 395, 398, 403, 461,
			507, 518, 526, 532, 609, 612,
<code>\EmFi@mimetype</code>	675, 677		715, 716, 718, 722, 724, 726,
<code>\EmFi@next</code>	410, 422, 428, 433, 440, 444		860, 871, 874, 877, 880, 919, 941
<code>\EmFi@order</code> ...	447, 463, 465, 486, 827	<code>\endinput</code>	140, 232, 243
<code>\EmFi@plain</code>	273, 277, 285, 334	<code>\errmessage</code>	961
<code>\EmFi@RequirePackage</code>		F	
	210, 218, 222, 234, 245, 333	<code>\futurelet</code>	304, 323
<code>\EmFi@S@ascending</code>	260, 537, 545	G	
<code>\EmFi@S@creationdate</code> ..	258, 481, 591	<code>\gdef</code>	322, 447, 557
<code>\EmFi@S@date</code>	253, 476, 512, 581	<code>\Gin@driver</code>	5
<code>\EmFi@S@desc</code>	256, 479, 587	I	
<code>\EmFi@S@descending</code>	261, 539, 546	<code>\ifcase</code>	119, 813
<code>\EmFi@S@details</code>	414, 415, 416	<code>\ifEmFi@collection</code>	264, 800
<code>\EmFi@S@false</code>	263, 611	<code>\ifEmFi@edit</code>	268, 491
<code>\EmFi@S@file</code>	255, 478, 585	<code>\ifEmFi@finished</code>	270, 432, 452, 636, 774
<code>\EmFi@S@hidden</code>	419, 420, 842	<code>\ifEmFi@id</code>	271, 382
<code>\EmFi@S@moddate</code>	257, 480, 589	<code>\ifEmFi@initialfile</code>	265, 797, 801, 837
<code>\EmFi@S@number</code>	254, 477, 523, 583	<code>\ifEmFi@item</code>	269, 664, 701
<code>\EmFi@S@size</code>	259, 482, 593	<code>\ifEmFi@sort</code>	266
<code>\EmFi@S@text</code>	252, 467, 501, 579	<code>\ifEmFi@visible</code>	267, 487
<code>\EmFi@S@tile</code>	417, 418, 840	<code>\ifnum</code>	285, 334, 731, 746, 827, 907, 915
<code>\EmFi@S@true</code>	262, 608	<code>\ifpdf</code>	223
<code>\EmFi@schema</code> ..	446, 472, 473, 833, 835	<code>\ifx</code>	120, 122, 131,
<code>\EmFi@setboolean</code>	606, 622, 625		155, 163, 166, 211, 236, 272,
<code>\EmFi@sortcase</code>	450, 557, 564, 813		276, 299, 307, 310, 327, 367,
<code>\EmFi@sortkeys</code>	551, 552, 627, 817, 822		370, 389, 395, 398, 413, 415,
<code>\EmFi@sortorders</code>			417, 419, 461, 476, 477, 478,
	555, 556, 559, 560, 628, 818, 823		479, 480, 481, 482, 501, 512,
<code>\EmFi@stringmethod</code>	731		523, 537, 539, 549, 555, 579,
<code>\EmFi@temp</code>	246, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 274, 280, 281, 282, 283, 284, 286, 291, 292, 293, 294, 295, 335, 339, 340, 341, 342, 343, 409, 413, 415, 417, 419, 424, 504, 505, 508, 515, 516, 519, 536, 537, 538, 539, 540, 542, 544, 549, 556, 562, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594,		581, 583, 585, 587, 589, 591, 593, 608, 611, 645, 650, 659, 675, 689, 694, 715, 716, 722, 742, 781, 803, 833, 840, 842, 845, 860, 871, 874, 877, 880, 919
		<code>\immediate</code>	
			133, 157, 666, 673, 686, 788, 831
		<code>\input</code>	4, 6, 7, 213, 920, 945
		<code>\iterate</code>	888, 890, 892
		J	
		<code>\jobname</code>	95, 103, 108, 109, 971, 972, 974

K	
\key	741, 743, 746, 750, 766
\KV@errx	296
L	
\LaTeX	977
\LoadCommand	920, 937
\loop	886, 902, 913
M	
\MessageBreak	424, 544, 597, 598, 615, 807
N	
\NeedsTeXFormat	40, 101, 970
\newif	264, 265, 266, 267, 268, 269, 270, 271
\next	892, 894, 896
\nopagenumbers	952
P	
\PackageInfo	136
\par	956
\pdf@escapehex	741
\pdf@escapename	474, 553, 677, 691, 718, 721
\pdf@escapestring	735
\pdf@filemdfivesum	682
\pdf@filemoddate	680
\pdf@filesize	651, 681
\pdf@strcmp	731, 746
\pdfcatalog	852
\pdflastobj	384, 671, 699, 709, 794, 853
\pdfnames	793
\pdfobj	666, 673, 686, 788, 831
\pdfstringdef	43, 367, 370, 732
\ProvidesPackage	102, 172
R	
\RangeCatcodeInvalid	911, 923, 924, 925, 926
S	
\SavedJobname	971, 974
\setkeys	441, 470, 630, 648
\space	384, 435, 454, 561, 638, 699, 702, 709, 794, 853
\stop	758, 765, 769
T	
\Test	922, 940, 955, 964, 965
\TeX	953
\texttt	977
\the	179, 180, 181, 182, 193, 384, 465, 671, 699, 709, 749, 756, 759, 762, 766, 794, 853, 905
\TMP@EnsureCode	190, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209
\toks@	748, 749, 756, 759, 762, 766, 768
\tt	953
U	
\usepackage	42, 45
W	
\write	133, 157
X	
\x	118, 120, 122, 132, 136, 138, 156, 161, 171, 177, 185, 747, 754
Z	
\z@	273
\repeat	
\RequirePackage	
\reserved@a	
\reserved@b	
\reserved@c	
\reserved@d	
\resetatcatcode	
\RestoreCatcodes ..	