

The embedfile package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2007/10/29 v2.1

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	9
2.6	Switches	10
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	22
4	Installation	23
4.1	Download	23
4.2	Bundle installation	23
4.3	Package installation	24
4.4	Refresh file name databases	24
4.5	Some details for the interested	24
5	References	25

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

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 L^AT_EX and plain-T_EX. See [subsection 1.5.1](#) that explains the use with plain-T_EX by an example. In L^AT_EX 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 [subsubsection 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 % Load packages
3 \input miniltx
4 % \def\Gin@driver{pdftex.def}
5 % \input graphicx.sty
6 \input embedfile.sty
7 \resetatcatcode
8
9 % default setting
10 \embedfilesetup{
11   mimetype=text/plain
12 }
13
14 % Embed files
15 \embedfile[
16   filespec=example.tex,
17   desc={Source code (plain-TeX) of this example}]
```

```

18 ]{embedfile-example-plain.tex}
19
20 \embedfile[
21   desc={Source of package 'embedfile'}
22 ]{embedfile.dtx}
23
24 \embedfile[
25   mimetype=application/pdf,
26   desc={Documentation of package 'embedfile'}
27 ]{embedfile.pdf}
28
29 % Some text
30 This example document contains three embedded files.
31
32 % End of document
33 \embedfilefinish % don't forget
34 \bye
35 </exampleplain>

```

1.5.2 Collection example

```

36 <*examplecollection>
37 \NeedsTeXFormat{LaTeX2e}
38 \documentclass{article}
39 \usepackage[bookmarks=false]{hyperref}
40 % provides \pdfstringdef that is then used by 'title' and
41 % other keys.
42 \usepackage{embedfile}[2007/10/29]
43 \embedfilesetup{
44   view=details,
45   initialfile=embedfile.pdf
46 }
47 \embedfilefield{file}{
48   type=file,
49   title={File name}
50 }
51 \embedfilefield{description}{
52   type=desc,
53   title={Description}
54 }
55 \embedfilefield{date}{
56   type=moddate,
57   title={Date}
58 }
59 \embedfilefield{size}{
60   type=size,
61   title={Size}
62 }
63 \embedfilefield{type}{
64   type=text,
65   title={Type},
66   visible=false
67 }
68 \embedfilesort{
69   type,
70   date=descending
71 }
72 \begin{document}
73 An example for embedded files as collection.
74 You need Acrobat Reader 8 or higher.
75
76 \embedfile[
77   desc={Source file of package 'embedfile'},

```

```

78 description.prefix={Package: },
79 type.value={DTX}
80 ]{embedfile.dtx}
81
82 \embedfile[
83 desc={Documentation of package 'embedfile'},
84 description.prefix={Package: },
85 type.value={PDF}
86 ]{embedfile.pdf}
87
88 \embedfile[
89 desc={The source for this example},
90 description.prefix={Example: },
91 type.value={TEX}
92 ]{\jobname.tex}
93
94 \end{document}
95 %
96 </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/](https://ctan.org/pkg/macros-latex-contrib-oberdiek). It also serves as small example for the use of the package with L^AT_EX.

```

97 <{*dtxattach}
98 \NeedsTeXFormat{LaTeX2e}
99 \ProvidesPackage{dtx-attach}
100 [2007/10/29 v2.1 Embed \string\jobname.dtx (HO)]%
101 \RequirePackage{embedfile}[2007/10/29]
102 \embedfile[%
103   stringmethod=escape,%
104   mimetype=plain/text,%
105   desc={LaTeX docstrip source archive for package '\jobname'}%
106 ]{\jobname.dtx}
107 </dtxattach>

```

2 Implementation

```
108 <{*package}>
```

2.1 Reload check and package identification

Reload check, especially if the package is not used with L^AT_EX.

```

109 \begingroup
110 \catcode44 12 % ,
111 \catcode45 12 % -
112 \catcode46 12 % .
113 \catcode58 12 % :
114 \catcode64 11 % @
115 \expandafter\let\expandafter\x\csname ver@embedfile.sty\endcsname
116 \ifcase 0%
117   \ifx\x\relax % plain
118   \else
119     \ifx\x\empty % LaTeX
120     \else
121       1%
122     \fi
123   \fi
124 \else
125   \expandafter\ifx\csname PackageInfo\endcsname\relax
126     \def\x#1#2{%

```

```

127     \immediate\write-1{Package #1 Info: #2.}%
128 }%
129 \else
130     \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
131 \fi
132 \x{embedfile}{The package is already loaded}%
133 \endgroup
134 \expandafter\endinput
135 \fi
136 \endgroup
Package identification:
137 \begingroup
138 \catcode40 12 % (
139 \catcode41 12 % )
140 \catcode44 12 % ,
141 \catcode45 12 % -
142 \catcode46 12 % .
143 \catcode47 12 % /
144 \catcode58 12 % :
145 \catcode64 11 % @
146 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
147     \def\x#1#2#3[#4]{\endgroup
148         \immediate\write-1{Package: #3 #4}%
149         \xdef#1{#4}%
150     }%
151 \else
152     \def\x#1#2[#3]{\endgroup
153         #2[{#3}]%
154         \ifx#1\relax
155             \xdef#1{#3}%
156         \fi
157     }%
158 \fi
159 \expandafter\x\csname ver@embedfile.sty\endcsname
160 \ProvidesPackage{embedfile}%
161 [2007/10/29 v2.1 embed files into PDF (H0)]

```

2.2 Catcodes

```

162 \expandafter\edef\csname EmFi@AtEnd\endcsname{%
163     \catcode64 \the\catcode64\relax
164 }
165 \catcode64 11 % @
166 \def\TMP@EnsureCode#1#2{%
167     \edef\EmFi@AtEnd{%
168         \EmFi@AtEnd
169         \catcode#1 \the\catcode#1\relax
170     }%
171     \catcode#1 #2\relax
172 }
173 \TMP@EnsureCode{39}{12}% '
174 \TMP@EnsureCode{40}{12}% (
175 \TMP@EnsureCode{41}{12}% )
176 \TMP@EnsureCode{44}{12}% ,
177 \TMP@EnsureCode{46}{12}% .
178 \TMP@EnsureCode{47}{12}% /
179 \TMP@EnsureCode{58}{12}% :
180 \TMP@EnsureCode{60}{12}% <
181 \TMP@EnsureCode{61}{12}% =
182 \TMP@EnsureCode{62}{12}% >
183 \TMP@EnsureCode{91}{12}% [
184 \TMP@EnsureCode{93}{12}% ]

```



```
185 \TMP@EnsureCode{96}-{12}% ‘
```

2.3 Tools

```
\EmFi@RequirePackage
```

```
186 \begingroup\expandafter\expandafter\expandafter\endgroup
187 \expandafter\ifx\csname RequirePackage\endcsname\relax
188   \def\EmFi@RequirePackage#1[#2]{%
189     \input #1.sty\relax
190   }%
191 \else
192   \let\EmFi@RequirePackage\RequirePackage
193 \fi
```

```
\EmFi@Error
```

```
194 \EmFi@RequirePackage{infwarerr}[2007/09/09]%
195 \def\EmFi@Error{%
196   \@PackageError{embedfile}%
197 }
```

2.4 Check for recent pdfTeX in PDF mode

Load package ifpdf and check mode.

```
198 \EmFi@RequirePackage{ifpdf}[2007/09/09]
199 \ifpdf
200 \else
201   \EmFi@Error{%
202     Missing pdfTeX in PDF mode%
203   }{%
204     Currently other drivers are not supported. %
205     Package loading is aborted.%
206   }%
207   \EmFi@AtEnd
208   \expandafter\endinput
209 \fi
```

Check version.

```
210 \begingroup\expandafter\expandafter\expandafter\endgroup
211 \expandafter\ifx\csname pdffilesize\endcsname\relax
212   \EmFi@Error{%
213     Unsupported pdfTeX version%
214   }{%
215     At least version 1.30 is necessary. Package loading is aborted.%
216   }%
217   \EmFi@AtEnd
218   \expandafter\endinput
219 \fi
```

2.5 Strings

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

```
220 \EmFi@RequirePackage{pdfescape}[2007/10/27]
221 \def\EmFi@temp#1{%
222   \expandafter\EdefSanitize\csname EmFi@S@#1\endcsname{#1}%
223 }
```

```
\EmFi@details
```

```
224 \EmFi@temp{details}%
```

```
\EmFi@tile
```

```
225 \EmFi@temp{tile}%
```

```

\EmFi@hidden
226 \EmFi@temp{hidden}%

\EmFi@S@text
227 \EmFi@temp{text}

\EmFi@S@date
228 \EmFi@temp{date}

\EmFi@S@number
229 \EmFi@temp{number}

\EmFi@S@file
230 \EmFi@temp{file}

\EmFi@S@desc
231 \EmFi@temp{desc}

\EmFi@S@moddate
232 \EmFi@temp{moddate}

\EmFi@S@creationdate
233 \EmFi@temp{creationdate}

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

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

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

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

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

```

2.6 Switches

```

\ifEmFi@collection
239 \newif\ifEmFi@collection

\ifEmFi@initialfile
240 \newif\ifEmFi@initialfile

\ifEmFi@sort
241 \newif\ifEmFi@sort

\ifEmFi@visible
242 \newif\ifEmFi@visible

\ifEmFi@edit
243 \newif\ifEmFi@edit

\ifEmFi@item
244 \newif\ifEmFi@item

```

```

\ifEmFi@finished

245 \newif\ifEmFi@finished

\ifEmFi@id

246 \newif\ifEmFi@id

```

2.7 Key value definitions

```

247 \expandafter\ifx\csname define@key\endcsname\relax
248   \chardef\EmFi@plain=\z@
249   \def\EmFi@temp#1{%
250     \begingroup\expandafter\expandafter\expandafter\endgroup
251     \expandafter\ifx\csname#1\endcsname\relax
252       \chardef\EmFi@plain=\@ne
253     \fi
254   }%
255   \EmFi@temp{NeedsTeXFormat}%
256   \EmFi@temp{ProvidesPackage}%
257   \EmFi@temp{DeclareOption}%
258   \EmFi@temp{ExecuteOptions}%
259   \EmFi@temp{ProcessOptions}%
260   \ifnum\EmFi@plain=\@ne
261     \def\EmFi@temp#1{%
262       \expandafter\let\csname EmFi@Org#1\expandafter\endcsname
263         \csname#1\endcsname
264       \expandafter\def\csname#1\endcsname
265     }%
266     \EmFi@temp{NeedsTeXFormat}#1{}%
267     \EmFi@temp{ProvidesPackage}#1[#2]{}% hash-ok
268     \EmFi@temp{DeclareOption}#1{}%
269     \EmFi@temp{ExecuteOptions}#1{}%
270     \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.

271   \def\KV@errx#1{%
272     \@PackageError{keyval}{#1}\@ehc
273   }%

```

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

```

\@ifnextchar

274   \expandafter\ifx\csname @ifnextchar\endcsname\relax
275     \def\@ifnextchar#1#2#3{%
276       \let\reserved@d=#1%
277       \def\reserved@a{#2}%
278       \def\reserved@b{#3}%
279       \futurelet\@let@token\@ifnch
280     }%
281     \def\@ifnch{%
282       \ifx\@let@token\@sptoken
283         \let\reserved@c\@xifnch
284       \else
285         \ifx\@let@token\reserved@d
286           \let\reserved@c\reserved@a
287         \else
288           \let\reserved@c\reserved@b
289         \fi
290       \fi
291       \reserved@c
292     }%
293     \begingroup

```

```

294      \def\:{\global\let\@sptoken= }%
295      \: % this makes \@sptoken a space token
296      \def\:{\@xifnch}%
297      \expandafter\gdef\:{%
298      \futurelet\@let@token\@ifnch
299      }%
300      \endgroup
301      \fi

\@namedef

302      \expandafter\ifx\csname @namedef\endcsname\relax
303      \def\@namedef#1{%
304      \expandafter\def\csname#1\endcsname
305      }%
306      \fi

307      \fi
308      \EmFi@RequirePackage{keyval}[1999/03/16]%
309      \ifnum\EmFi@plain=\@ne
310      \def\EmFi@temp#1{%
311      \expandafter\let\csname#1\expandafter\endcsname
312      \csname EmFi@Org#1\endcsname
313      }%
314      \EmFi@temp{NeedsTeXFormat}%
315      \EmFi@temp{ProvidesPackage}%
316      \EmFi@temp{DeclareOption}%
317      \EmFi@temp{ExecuteOptions}%
318      \EmFi@temp{ProcessOptions}%
319      \fi
320      \fi

\EmFi@GlobalKey

321      \def\EmFi@GlobalKey#1#2{%
322      \global\expandafter\let\csname KV@#1@#2\expandafter\endcsname
323      \csname KV@#1@#2\endcsname
324      }

\EmFi@GlobalDefaultKey

325      \def\EmFi@GlobalDefaultKey#1#2{%
326      \EmFi@GlobalKey{#1}{#2}%
327      \global\expandafter\let
328      \csname KV@#1@#2@default\expandafter\endcsname
329      \csname KV@#1@#2@default\endcsname
330      }

\EmFi@DefineKey

331      \def\EmFi@DefineKey#1#2{%
332      \define@key{EmFi}{#1}{%
333      \expandafter\def\csname EmFi@#1\endcsname{##1}%
334      }%
335      \expandafter\def\csname EmFi@#1\endcsname{#2}%
336      }

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

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

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

```

Description (optional).

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

Method for converting text to PDF strings.

```
341 \EmFi@DefineKey{stringmethod}{%  
342   \ifx\pdfstringdef\@undefined  
343     escape%  
344   \else  
345     \ifx\pdfstringdef\relax  
346       escape%  
347     \else  
348       psd%  
349     \fi  
350   \fi  
351 }
```

Option id as key for object numbers.

```
352 \define@key{EmFi}{id}{%  
353   \def\EmFi@id{#1}%  
354   \EmFi@idtrue  
355 }
```

\EmFi@defobj

```
356 \def\EmFi@defobj#1{%  
357   \ifEmFi@id  
358     \expandafter\xdef\csname EmFi@#1\EmFi@id\endcsname{%  
359       \the\pdflastobj\space 0 R%  
360     }%  
361   \fi  
362 }
```

\embedfileifobjectexists

```
363 \def\embedfileifobjectexists#1#2{%  
364   \expandafter\ifx\csname EmFi@#2@#1\endcsname\relax  
365     \expandafter\@secondoftwo  
366   \else  
367     \expandafter\@firstoftwo  
368   \fi  
369 }
```

\@firstoftwo

```
370 \expandafter\ifx\csname @firstoftwo\endcsname\relax  
371   \long\def\@firstoftwo#1#2{#1}%  
372 \fi
```

\@secondoftwo

```
373 \expandafter\ifx\csname @secondoftwo\endcsname\relax  
374   \long\def\@secondoftwo#1#2{#2}%  
375 \fi
```

\embedfilegetobject

```
376 \def\embedfilegetobject#1#2{%  
377   \embedfileifobjectexists{#1}{#2}{%  
378     \csname EmFi@#2@#1\endcsname  
379   }{%  
380     0 0 R%  
381   }%  
382 }
```

Initial view of the collection.

```
383 \define@key{EmFi}{view}[]{%  
384   \EdefSanitize\EmFi@temp{#1}%
```

```

385 \def\EmFi@next{%
386   \global\EmFi@collectiontrue
387 }%
388 \ifx\EmFi@temp\empty
389   \let\EmFi@view\EmFi@S@details
390 \else\ifx\EmFi@temp\EmFi@S@details
391   \let\EmFi@view\EmFi@S@details
392 \else\ifx\EmFi@temp\EmFi@S@tile
393   \let\EmFi@view\EmFi@S@tile
394 \else\ifx\EmFi@temp\EmFi@S@hidden
395   \let\EmFi@view\EmFi@S@hidden
396 \else
397   \let\EmFi@next\relax
398   \EmFi@Error{%
399     Unknown value '\EmFi@temp' for key 'view'.\MessageBreak
400     Supported values: 'details', 'tile', 'hidden'.%
401   }\@ehc
402 \fi\fi\fi\fi
403 \EmFi@next
404 }
405 \EmFi@DefineKey{initialfile}{}

\embedfilessetup

406 \def\embedfilessetup{%
407   \ifEmFi@finished
408     \def\EmFi@next##1{%
409       \EmFi@Error{%
410         \string\embedfilefield\space after \string\embedfilefinish
411       }{%
412         The list of embedded files is already written.%
413       }%
414     \else
415       \def\EmFi@next{%
416         \setkeys{EmFi}%
417       }%
418     \fi
419   \EmFi@next
420 }

\EmFi@schema

421 \def\EmFi@schema{}

\EmFi@order

422 \gdef\EmFi@order{0}

\EmFi@@order

423 \let\EmFi@@order\relax

\EmFi@fieldlist

424 \def\EmFi@fieldlist{}

\EmFi@sortcase

425 \def\EmFi@sortcase{0}%

\embedfilefield

426 \def\embedfilefield#1#2{%
427   \ifEmFi@finished
428     \EmFi@Error{%
429       \string\embedfilefield\space after \string\embedfilefinish
430     }{%

```

```

431     The list of embedded files is already written.%
432 }%
433 \else
434   \global\EmFi@collectiontrue
435   \edef\Sanitize\EmFi@key{#1}%
436   \expandafter\ifx\csname KV@EmFi@\EmFi@key.prefix\endcsname\relax
437     \begin{group}
438       \count@=\EmFi@order
439       \advance\count@ 1 %
440       \xdef\EmFi@order{\the\count@}%
441       \let\EmFi@title\EmFi@key
442       \let\EmFi@type\EmFi@S@text
443       \EmFi@visibletrue
444       \EmFi@editfalse
445       \setkeys{EmFiFi}{#2}%
446       \EmFi@convert\EmFi@title\EmFi@title
447       \xdef\EmFi@schema{%
448         \EmFi@schema
449         /\pdfescapename{\EmFi@key}<<%
450         /Subtype/%
451         \ifx\EmFi@type\EmFi@S@date D%
452         \else\ifx\EmFi@type\EmFi@S@number N%
453         \else\ifx\EmFi@type\EmFi@S@file F%
454         \else\ifx\EmFi@type\EmFi@S@desc Desc%
455         \else\ifx\EmFi@type\EmFi@S@moddate ModDate%
456         \else\ifx\EmFi@type\EmFi@S@creationdate CreationDate%
457         \else\ifx\EmFi@type\EmFi@S@size Size%
458         \else S%
459         \fi\fi\fi\fi\fi\fi\fi
460         /N(\EmFi@title)%
461         \EmFi@@order{\EmFi@order}%
462         \ifEmFi@visible
463         \else
464           /V false%
465         \fi
466         \ifEmFi@edit
467           /E true%
468         \fi
469         >>%
470       }%
471       \let\do\relax
472       \xdef\EmFi@fieldlist{%
473         \EmFi@fieldlist
474         \do{\EmFi@key}%
475       }%
476       \ifx\EmFi@type\EmFi@S@text
477         \define@key{EmFi}{\EmFi@key.value}{%
478           \EmFi@itemtrue
479           \def\EmFi@temp{##1}%
480           \EmFi@convert\EmFi@temp\EmFi@temp
481           \expandafter\def\csname EmFi@V@#1%
482           \expandafter\endcsname\expandafter{%
483             \expandafter(\EmFi@temp)%
484           }%
485         }%
486         \EmFi@GlobalKey{EmFi}{\EmFi@key.value}%
487       \else\ifx\EmFi@type\EmFi@S@date
488         \define@key{EmFi}{\EmFi@key.value}{%
489           \EmFi@itemtrue
490           \def\EmFi@temp{##1}%
491           \EmFi@convert\EmFi@temp\EmFi@temp
492           \expandafter\def\csname EmFi@V@#1%

```

```

493         \expandafter\endcsname\expandafter{%
494         \expandafter(\EmFi@temp)%
495     }%
496 }%
497 \EmFi@GlobalKey{EmFi}{\EmFi@key.value}%
498 \else\ifx\EmFi@type\EmFi@S@number
499     \define@key{EmFi}{\EmFi@key.value}{%
500     \EmFi@itemtrue
501     \expandafter\edef\csname EmFi@V@#1\endcsname{ ##1}%
502     }%
503     \EmFi@GlobalKey{EmFi}{\EmFi@key.value}%
504 \fi\fi\fi
505 \define@key{EmFi}{\EmFi@key.prefix}{%
506     \EmFi@itemtrue
507     \expandafter\def\csname EmFi@P@#1\endcsname{##1}%
508 }%
509 \EmFi@GlobalKey{EmFi}{\EmFi@key.prefix}%
510 \define@key{EmFiSo}{\EmFi@key}[ascending]{%
511     \edef\csname\EmFi@temp{##1}%
512     \ifx\EmFi@temp\EmFi@S@ascending
513         \def\EmFi@temp{true}%
514     \else\ifx\EmFi@temp\EmFi@S@descending
515         \def\EmFi@temp{false}%
516     \else
517         \def\EmFi@temp{}%
518     \EmFi@Error{%
519         Unknown sort order '\EmFi@temp'.\MessageBreak
520         Supported values: '\EmFi@S@ascending', %
521         '\EmFi@S@descending
522     }\@ehc
523 \fi\fi
524 \ifx\EmFi@temp\empty
525 \else
526     \xdef\EmFi@sortkeys{%
527         \EmFi@sortkeys
528         /\pdfescapename{#1}%
529     }%
530     \ifx\EmFi@sortorders\empty
531         \global\let\EmFi@sortorders\EmFi@temp
532         \gdef\EmFi@sortcase{1}%
533     \else
534         \xdef\EmFi@sortorders{%
535             \EmFi@sortorders
536             \space
537             \EmFi@temp
538         }%
539         \xdef\EmFi@sortcase{2}%
540     \fi
541 \fi
542 }%
543 \EmFi@GlobalDefaultKey{EmFiSo}\EmFi@key
544 \endgroup
545 \else
546     \EmFi@Error{%
547         Field '\EmFi@key' is already defined%
548     }\@ehc
549 \fi
550 \fi
551 }

552 \define@key{EmFiFi}{type}{%
553     \edef\csname\EmFi@temp{##1}%
554     \ifx\EmFi@temp\EmFi@S@text

```



```

555 \let\EmFi@type\EmFi@temp
556 \else\ifx\EmFi@temp\EmFi@S@date
557 \let\EmFi@type\EmFi@temp
558 \else\ifx\EmFi@temp\EmFi@S@number
559 \let\EmFi@type\EmFi@temp
560 \else\ifx\EmFi@temp\EmFi@S@file
561 \let\EmFi@type\EmFi@temp
562 \else\ifx\EmFi@temp\EmFi@S@desc
563 \let\EmFi@type\EmFi@temp
564 \else\ifx\EmFi@temp\EmFi@S@moddate
565 \let\EmFi@type\EmFi@temp
566 \else\ifx\EmFi@temp\EmFi@S@creationdate
567 \let\EmFi@type\EmFi@temp
568 \else\ifx\EmFi@temp\EmFi@S@size
569 \let\EmFi@type\EmFi@temp
570 \else
571 \EmFi@Error{%
572     Unknown type '\EmFi@temp'.\MessageBreak
573     Supported types: 'text', 'date', 'number', 'file',\MessageBreak
574     'desc', 'moddate', 'creationdate', 'size'%
575 }%
576 \fi\fi\fi\fi\fi\fi\fi
577 }

578 \define@key{EmFiFi}{title}{%
579 \def\EmFi@title{#1}%
580 }

```

\EmFi@setboolean

```

581 \def\EmFi@setboolean#1#2{%
582 \EdefSanitize\EmFi@temp{#2}%
583 \ifx\EmFi@temp\EmFi@S@true
584 \csname EmFi@#1true\endcsname
585 \else
586 \ifx\EmFi@temp\EmFi@S@false
587 \csname EmFi@#1false\endcsname
588 \else
589 \EmFi@Error{%
590     Unknown value '\EmFi@temp' for key '#1'.\MessageBreak
591     Supported values: 'true', 'false'%
592 }\@ehc
593 \fi
594 \fi
595 }

596 \define@key{EmFiFi}{visible}[true]{%
597 \EmFi@setboolean{visible}{#1}%
598 }

599 \define@key{EmFiFi}{edit}[true]{%
600 \EmFi@setboolean{edit}{#1}%
601 }

```

\EmFi@sortkeys

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

\EmFi@sortorders

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

\embedfilesort

```

604 \def\embedfilesort{%
605 \setkeys{EmFiSo}%
606 }

```

2.8 Embed the file

```
\embedfile

607 \def\embedfile{%
608   \@ifnextchar[\EmFi@embedfile{\EmFi@embedfile[]}%
609 }

\EmFi@embedfile

610 \def\EmFi@embedfile[#1]#2{%
611   \ifEmFi@finished
612     \EmFi@Error{%
613       \string\embedfile\space after \string\embedfilefinish
614     }{%
615       The list of embedded files is already written.%
616     }%
617   \else
618     \begingroup
619     \def\EmFi@file{#2}%
620     \ifx\EmFi@file\EmFi@initialfile
621       \global\EmFi@initialfiletrue
622     \fi
623     \setkeys{EmFi}{#1}%
624     \expandafter\ifx\expandafter\\pdffilesize{\EmFi@file}\\%
625     \EmFi@Error{%
626       File '\EmFi@file' not found%
627     }{%
628       The unknown file is not embedded.%
629     }%
630   \else
631     \EmFi@convert\EmFi@filespec\EmFi@@filespec
632     \ifx\EmFi@desc\empty
633       \let\EmFi@@desc\empty
634     \else
635       \EmFi@convert\EmFi@desc\EmFi@@desc
636     \fi
637     \ifEmFi@item
638       \let\do\EmFi@do
639       \immediate\pdfobj{%
640         <<%
641           \EmFi@fieldlist
642         >>%
643       }%
644       \edef\EmFi@ci{\the\pdflastobj}%
645     \fi
646     \immediate\pdfobj stream attr{%
647       /Type/EmbeddedFile%
648       \ifx\EmFi@mimetype\empty
649       \else
650         /Subtype/\pdfescapename{\EmFi@mimetype}%
651       \fi
652       /Params<<%
653         /ModDate(\pdffilemoddate{\EmFi@file})%
654         /Size \pdffilesize{\EmFi@file}%
655         /Checksum<\pdfmdfivesum file{\EmFi@file}>%
656       >>%
657     }file{\EmFi@file}\relax
658     \EmFi@defobj{EmbeddedFile}%
659     \immediate\pdfobj{%
660       <<%
661         /Type/Filespec%
662         \ifx\EmFi@filesystem\empty
663         \else
```

```

664         /FS/\pdfescapename{\EmFi@filesystem}%
665         \fi
666         /F(\EmFi@@filespec)%
667         \ifx\EmFi@@desc\empty
668         \else
669         /Desc(\EmFi@@desc)%
670         \fi
671         /EF<<%
672         /F \the\pdflastobj\space 0 R%
673         >>%
674         \ifEmFi@item
675         /CI \EmFi@ci\space 0 R%
676         \fi
677         >>%
678     }%
679     \EmFi@defobj{Filespec}%
680     \EmFi@add{%
681         \EmFi@@filespec
682     }{\the\pdflastobj\space 0 R}%
683     \fi
684 \endgroup
685 \fi
686 }

```

\EmFi@do

```

687 \def\EmFi@do#1{%
688   \expandafter\ifx\csname EmFi@P@#1\endcsname\relax
689   \expandafter\ifx\csname EmFi@V@#1\endcsname\relax
690   \else
691     /\pdfescapename{#1}\csname EmFi@V@#1\endcsname
692     \fi
693   \else
694     /\pdfescapename{#1}<<%
695     \expandafter\ifx\csname EmFi@V@#1\endcsname\relax
696     \else
697       /D\csname EmFi@V@#1\endcsname
698       \fi
699       /P(\csname EmFi@P@#1\endcsname)%
700       >>%
701     \fi
702 }

```

\EmFi@convert

```

703 \def\EmFi@convert#1#2{%
704   \ifnum\pdfstrcmp{\EmFi@stringmethod}{psd}=0 %
705     \pdfstringdef\EmFi@temp{#1}%
706     \let#2\EmFi@temp
707   \else
708     \edef#2{\pdfescapestring{#1}}%
709   \fi
710 }

```

```

711 \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, ...).

```

712 \def\EmFi@add#1#2{%
713   \begingroup
714   \edef\key{\pdfescapehex{#1}}%
715   \ifx\EmFi@list\empty

```

```

716     \xdef\EmFi@list{\noexpand\do{\key}{#2}}%
717 \else
718     \def\do##1##2{%
719         \ifnum\pdfstrcmp{##1}{\key}>0 %
720             \edef\x{%
721                 \toks@{%
722                     \the\toks@%
723                     \noexpand\do{\key}{#2}%
724                     \noexpand\do{##1}{##2}%
725                 }%
726             }%
727             \x
728             \def\do####1####2{%
729                 \toks@\expandafter{\the\toks@\do{####1}{####2}}%
730             }%
731             \def\stop{%
732                 \xdef\EmFi@list{\the\toks@}%
733             }%
734         \else
735             \toks@\expandafter{\the\toks@\do{##1}{##2}}%
736         \fi
737     }%
738     \def\stop{%
739         \xdef\EmFi@list{\the\toks@\noexpand\do{\key}{#2}}%
740     }%
741     \toks@{}%
742     \EmFi@list\stop
743 \fi
744 \endgroup
745 }

```

\embedfilefinish

```

746 \def\embedfilefinish{%
747     \ifEmFi@finished
748         \EmFi@Error{%
749             Too many invocations of \string\embedfilefinish
750         }{%
751             The list of embedded files is already written.%
752         }%
753     \else
754         \ifx\EmFi@list\empty
755             \else
756                 \global\EmFi@finishedtrue
757                 \begingroup
758                 \def\do##1##2{%
759                     <##1>##2%
760                 }%
761                 \immediate\pdfobj{%
762                     <<%
763                         /Names[\EmFi@list]%
764                     >>%
765                 }%
766                 \pdfnames{%
767                     /EmbeddedFiles \the\pdflastobj\space 0 R%
768                 }%
769                 \endgroup
770                 Write collection objects.
771                 \ifEmFi@initialfile
772                     \EmFi@collectiontrue
773                 \fi

```

```

773 \ifEmFi@collection
774 \ifEmFi@initialfile
775 \else
776 \ifx\EmFi@initialfile\empty
777 \EmFi@convert\EmFi@initialfile\EmFi@initialfile
778 \else
779 \@PackageWarningNoLine{embedfile}{%
780 Missing initial file '\EmFi@initialfile'\MessageBreak
781 among the embedded files%
782 }%
783 \EmFi@initialfilefalse
784 \fi
785 \fi
786 \ifcase\EmFi@sortcase
787 \def\EmFi@temp{}%
788 \or
789 \def\EmFi@temp{%
790 /S\EmFi@sortkeys
791 /A \EmFi@sortorders
792 }%
793 \else
794 \def\EmFi@temp{%
795 /S[\EmFi@sortkeys]%
796 /A[\EmFi@sortorders]%
797 }%
798 \fi
799 \def\EmFi@@order##1{%
800 \ifnum\EmFi@order>1 %
801 /O ##1%
802 \fi
803 }%
804 \immediate\pdfobj{%
805 <<%
806 \ifx\EmFi@schema\empty
807 \else
808 /Schema<<\EmFi@schema>>%
809 \fi
810 \ifEmFi@initialfile
811 /D(\EmFi@initialfile)%
812 \fi
813 \ifx\EmFi@view\EmFi@S@tile
814 /View/T%
815 \else\ifx\EmFi@view\EmFi@S@hidden
816 /View/H%
817 \fi\fi
818 \ifx\EmFi@temp\empty
819 \EmFi@temp
820 \else
821 /Sort<<\EmFi@temp>>%
822 \fi
823 >>%
824 }%
825 \pdfcatalog{%
826 /Collection \the\pdflastobj\space0 R%
827 }%
828 \fi
829 \fi
830 \fi
831 }

832 \begingroup\expandafter\expandafter\expandafter\endgroup
833 \expandafter\ifx\csname AtEndDocument\endcsname\relax
834 \else

```

```

835 \AtEndDocument{\embedfilefinish}%
836 \fi
837 \EmFi@AtEnd
838 \end{package}

```

3 Test

3.1 Catcode checks for loading

```

839 \test1
840 \catcode'\@=11 %
841 \def\RestoreCatcodes{
842 \count@=0 %
843 \loop
844 \edef\RestoreCatcodes{%
845 \RestoreCatcodes
846 \catcode\the\count@=\the\catcode\count@\relax
847 }%
848 \ifnum\count@<255 %
849 \advance\count@\@ne
850 \repeat
851
852 \def\RangeCatcodeInvalid#1#2{%
853 \count@=#1\relax
854 \loop
855 \catcode\count@=15 %
856 \ifnum\count@<#2\relax
857 \advance\count@\@ne
858 \repeat
859 }
860 \def\Test{%
861 \RangeCatcodeInvalid{0}{47}%
862 \RangeCatcodeInvalid{58}{64}%
863 \RangeCatcodeInvalid{91}{96}%
864 \RangeCatcodeInvalid{123}{255}%
865 \catcode'\@=12 %
866 \catcode'\=0 %
867 \catcode'\{=1 %
868 \catcode'\}=2 %
869 \catcode'\#=6 %
870 \catcode'\[=12 %
871 \catcode'\]=12 %
872 \catcode'\%=14 %
873 \catcode'\ =10 %
874 \catcode13=5 %
875 \input embedfile.sty\relax
876 \RestoreCatcodes
877 }
878 \Test
879 \csname @@end\endcsname
880 \end
881 \test1

```

3.2 Simple test

```

882 \test2
883 \input embedfile.sty\relax
884 \embedfile[%
885 stringmethod=escape,%
886 mimetype=plain/text,%
887 desc={LaTeX docstrip source archive for package 'embedfile'},%

```

```

888 id={embedfile.dtx}%
889 ]{embedfile.dtx}
890 \nopagenumbers
891 Test (plain-TeX): {\tt embedfile.dtx} should be embedded.%
892
893 \def\Test#1{%
894   \par
895   \embedfileifobjectexists{embedfile.dtx}{#1}{%
896     Object #1 (embedfile.dtx): %
897     \embedfilegetobject{embedfile.dtx}{#1}%
898   }{%
899     \errmessage{Missing object #1 (embedfile.dtx)}%
900   }%
901 }
902 \Test{EmbeddedFile}
903 \Test{Filespec}
904 \embedfilefinish
905 \bye
906 </test2>

907 <test3>
908 \NeedsTeXFormat{LaTeX2e}
909 \let\SavedJobname\jobname
910 \def\jobname{embedfile}
911 \RequirePackage{dtx-attach}[2007/10/29]
912 \let\jobname\SavedJobname
913 \documentclass{minimal}
914 \begin{document}
915   Test (\LaTeX): \texttt{embedfile.dtx} should be embedded.%
916 \end{document}
917 </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:macros/latex/contrib/oberdiek/oberdiek-tds.zip](#)

TDS refers to the standard “A Directory Structure for T_EX Files” ([CTAN: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):

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

¹<http://ftp.ctan.org/tex-archive/>

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 `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 \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/.
- [2] Heiko Oberdiek: *The attachfile2 package*; 2006/08/16 v2.2; 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`.

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	%	872
\#	869	\: 294, 295, 296, 297

\@	840, 865	\embedfilefinish	3, 33, 410, 429, 613, <u>746</u> , 835, 904
\@PackageError	196, 272	\embedfilegetobject	5, <u>376</u> , 897
\@PackageWarningNoLine	779	\embedfileifobjectexists	5, <u>363</u> , 377, 895
\@ehc	272, 401, 522, 548, 592	\embedfilessetup	3, 3, 10, 43, <u>406</u>
\@firstoftwo	367, <u>370</u>	\embedfilesort	4, 68, <u>604</u>
\@ifnch	279, 281, 298	\EmFi@desc	633, 635, 667, 669
\@ifnextchar	274, 608	\EmFi@filespec	631, 666, 681
\@let@token	279, 282, 285, 298	\EmFi@order	423, 461, 799
\@namedef	<u>302</u>	\EmFi@add	680, <u>712</u>
\@ne	252, 260, 309, 849, 857	\EmFi@AtEnd	167, 168, 207, 217, 837
\@secondoftwo	365, <u>373</u>	\EmFi@ci	644, 675
\@sptoken	282, 294, 295	\EmFi@collectiontrue	386, 434, 771
\@undefined	342	\EmFi@convert	446, 480, 491, 631, 635, <u>703</u> , 777
\@xifnch	283, 296	\EmFi@DefineKey	331, 337, 338, 339, 340, 341, 405
\[870	\EmFi@defobj	356, 658, 679
\\	624, 866	\EmFi@desc	632, 635
\{	867	\EmFi@details	224
\}	868	\EmFi@do	638, <u>687</u>
\]	871	\EmFi@editfalse	444
		\EmFi@embedfile	608, <u>610</u>
_	873	\EmFi@Error	194, 201, 212, 398, 409, 428, 518, 546, 571, 589, 612, 625, 748
A			
\advance	439, 849, 857	\EmFi@fieldlist	424, 472, 473, 641
\AtEndDocument	835	\EmFi@file	338, 619, 620, 624, 626, 653, 654, 655, 657
B			
\begin	72, 914	\EmFi@filespec	631
\bye	34, 905	\EmFi@filesystem	662, 664
C			
\catcode	110, 111, 112, 113, 114, 138, 139, 140, 141, 142, 143, 144, 145, 163, 165, 169, 171, 840, 846, 855, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874	\EmFi@finishedtrue	756
\chardef	248, 252	\EmFi@GlobalDefaultKey	325, 543
\count@	438, 439, 440, 842, 846, 848, 849, 853, 855, 856, 857	\EmFi@GlobalKey	321, 326, 486, 497, 503, 509
\csname	115, 125, 146, 159, 162, 187, 211, 222, 247, 251, 262, 263, 264, 274, 302, 304, 311, 312, 322, 323, 328, 329, 333, 335, 358, 364, 370, 373, 378, 436, 481, 492, 501, 507, 584, 587, 688, 689, 691, 695, 697, 699, 833, 879	\EmFi@hidden	226
D			
\define@key	332, 352, 383, 477, 488, 499, 505, 510, 552, 578, 596, 599	\EmFi@id	353, 358
\do	471, 474, 638, 716, 718, 723, 724, 728, 729, 735, 739, 758	\EmFi@idtrue	354
\documentclass	38, 913	\EmFi@initialfile	620, 776, 777, 780, 811
E			
\EdefSanitize	222, 384, 435, 501, 511, 553, 582	\EmFi@initialfilefalse	783
\embedfile	2, 15, 20, 24, 76, 82, 88, 102, <u>607</u> , 613, 884	\EmFi@initialfiletrue	621
\embedfilefield	4, 47, 51, 55, 59, 63, 410, <u>426</u>	\EmFi@itemtrue	478, 489, 500, 506
		\EmFi@key	435, 436, 441, 449, 474, 477, 486, 488, 497, 499, 503, 505, 509, 510, 543, 547
		\EmFi@list	711, 715, 716, 732, 739, 742, 754, 763
		\EmFi@mimetype	648, 650
		\EmFi@next	385, 397, 403, 408, 415, 419
		\EmFi@order	422, 438, 440, 461, 800
		\EmFi@plain	248, 252, 260, 309
		\EmFi@RequirePackage	186, 194, 198, 220, 308
		\EmFi@S@ascending	235, 512, 520
		\EmFi@S@creationdate	233, 456, 566
		\EmFi@S@date	228, 451, 487, 556
		\EmFi@S@desc	231, 454, 562
		\EmFi@S@descending	236, 514, 521
		\EmFi@S@details	389, 390, 391
		\EmFi@S@false	238, 586

R		\texttt 915
\RangeCatcodeInvalid		\the 163, 169,
. 852, 861, 862, 863, 864		359, 440, 644, 672, 682, 722,
\repeat 850, 858		729, 732, 735, 739, 767, 826, 846
\RequirePackage 101, 192, 911		\TMP@EnsureCode 166,
\reserved@a 277, 286		173, 174, 175, 176, 177, 178,
\reserved@b 278, 288		179, 180, 181, 182, 183, 184, 185
\reserved@c 283, 286, 288, 291		\toks@ 721, 722, 729, 732, 735, 739, 741
\reserved@d 276, 285		\tt 891
\resetatcatcode 7		
\RestoreCatcodes . . 841, 844, 845, 876		U
S		\usepackage 39, 42
\SavedJobname 909, 912		W
\setkeys 416, 445, 605, 623		\write 127, 148
\space 359, 410, 429,		
536, 613, 672, 675, 682, 767, 826		X
\stop 731, 738, 742		\x 115, 117, 119, 126,
		130, 132, 147, 152, 159, 720, 727
T		Z
\Test 860, 878, 893, 902, 903		\z@ 248
\TeX 891		