

# The embedfile package

Heiko Oberdiek  
<heiko.oberdiek at gmail.com>

2009/09/25 v2.4

## Abstract

This package embeds files to a PDF document. Currently the only supported driver is pdfTeX  $\geq 1.30$  in PDF mode.

## Contents

<b>1</b>	<b>Documentation</b>	<b>2</b>
1.1	Introduction	2
1.1.1	Future development	2
1.2	User interface	2
1.3	Collection support (PDF 1.7)	4
1.4	Export of object references	5
1.4.1	Example	5
1.5	Examples	6
1.5.1	plain-TeX	6
1.5.2	Collection example	6
1.6	Package dtx-attach	7
<b>2</b>	<b>Implementation</b>	<b>8</b>
2.1	Reload check and package identification	8
2.2	Catcodes	9
2.3	Tools	9
2.4	Check for recent pdfTeX in PDF mode	10
2.5	Strings	10
2.6	Switches	11
2.7	Key value definitions	12
2.8	Embed the file	18
<b>3</b>	<b>Test</b>	<b>23</b>
3.1	Catcode checks for loading	23
3.2	Simple test	24
<b>4</b>	<b>Installation</b>	<b>25</b>
4.1	Download	25
4.2	Bundle installation	25
4.3	Package installation	25
4.4	Refresh file name databases	26
4.5	Some details for the interested	26
<b>5</b>	<b>References</b>	<b>26</b>

<b>6 History</b>	<b>27</b>
[2006/08/16 v1.0]	27
[2007/04/11 v1.1]	27
[2007/09/09 v1.2]	27
[2007/10/28 v2.0]	27
[2007/10/29 v2.1]	27
[2007/11/11 v2.2]	27
[2007/11/25 v2.3]	27
[2009/09/25 v2.4]	27
<b>7 Index</b>	<b>28</b>

## 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 [subsection 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 (8-bit, path separators, ...), look into the PDF specification ([3]). There are rules how these file names must be written/encoded. Avoid 8-bit characters and other special characters, the behaviour is currently undefined. Use option `ucfilespec` for more funny file names. The string method, see below, is `escape` since version 2.4.

This name is also used as entry in a name tree (see PDF specification: `/EmbeddedFiles`). Therefore the value for `filespec` must be unique among all embedded files. Also key `initialfiles` refers to this name, if the file name and the value of `filespec` are different.

**ucfilespec** Since PDF 1.7 the file name may be provided in Unicode. The conversion of the option value into a PDF string is controlled by option `stringmethod`.

**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 `ucfilespec` and `desc` into a PDF string (before version 2.4: `filespec` and `desc`). 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 `LATEX` this is automatically done. The package uses `\AtEndDocument`. Then the list of all files should be known. However, `plain-TEX` 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. There must be an `\embedfile` command somewhere whose value for key `filespec` is used here. The `\embedfile` command can drop option `filespec` if the file name is not different.

`\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}[2009/09/25]
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/](https://ctan.org/ctan/packages/macros/latex/contrib/oberdiek/). It also serves as small example for the use of the package with L<sup>A</sup>T<sub>E</sub>X.

```

100 (*dtxattach)
101 \NeedsTeXFormat{LaTeX2e}
102 \ProvidesPackage{dtx-attach}
103 [2009/09/25 v2.4 Embed \string\jobname.dtx (HO)]%
104 \RequirePackage{embedfile}[2009/09/25]
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 L<sup>A</sup>T<sub>E</sub>X.

```

112 \begingroup
113 \catcode44 12 % ,
114 \catcode45 12 % -
115 \catcode46 12 % .
116 \catcode58 12 % :
117 \catcode64 11 % @
118 \catcode123 1 % {
119 \catcode125 2 % }
120 \expandafter\let\expandafter\x\csname ver@embedfile.sty\endcsname
121 \ifx\x\relax % plain-TeX, first loading
122 \else
123 \def\empty{}%
124 \ifx\x\empty % LaTeX, first loading,
125 % variable is initialized, but \ProvidesPackage not yet seen
126 \else
127 \catcode35 6 % #
128 \expandafter\ifx\csname PackageInfo\endcsname\relax
129 \def\x#1#2{%
130 \immediate\write-1{Package #1 Info: #2.}%
131 }%
132 \else
133 \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
134 \fi
135 \x{embedfile}{The package is already loaded}%
136 \aftergroup\endinput
137 \fi
138 \fi
139 \endgroup

```

Package identification:

```

140 \begingroup
141 \catcode35 6 % #
142 \catcode40 12 % (
143 \catcode41 12 % )
144 \catcode44 12 % ,
145 \catcode45 12 % -
146 \catcode46 12 % .
147 \catcode47 12 % /
148 \catcode58 12 % :
149 \catcode64 11 % @
150 \catcode91 12 % [
151 \catcode93 12 % ]
152 \catcode123 1 % {
153 \catcode125 2 % }
154 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
155 \def\x#1#2#3[#4]{\endgroup
156 \immediate\write-1{Package: #3 #4}%
157 \xdef#1{#4}%
158 }%
159 \else
160 \def\x#1#2[#3]{\endgroup
161 #2[#{#3}]%

```



```

162     \ifx#1\@undefined
163     \xdef#1{#3}%
164     \fi
165     \ifx#1\relax
166     \xdef#1{#3}%
167     \fi
168   }%
169   \fi
170 \expandafter\x\csname ver@embedfile.sty\endcsname
171 \ProvidesPackage{embedfile}%
172 [2009/09/25 v2.4 embed files into PDF (HO)]

```

## 2.2 Catcodes

```

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

```

## 2.3 Tools

\EmFi@RequirePackage

```

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

```

`\EmFi@Error`

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

## 2.4 Check for recent pdfTeX in PDF mode

Load package ifpdf and check mode.

```
221 \EmFi@RequirePackage{ifpdf}[2007/09/09]
222 \ifpdf
223 \else
224   \EmFi@Error{%
225     Missing pdfTeX in PDF mode%
226   }{%
227     Currently other drivers are not supported. %
228     Package loading is aborted.%
229   }%
230   \EmFi@AtEnd
231   \expandafter\endinput
232 \fi
```

```
233 \EmFi@RequirePackage{pdftexcmds}[2007/11/11]
234 \EmFi@RequirePackage{ltxcmds}[2009/08/05]
```

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@sort
    265 \newif\ifEmFi@sort

\ifEmFi@visible
    266 \newif\ifEmFi@visible

\ifEmFi@edit
    267 \newif\ifEmFi@edit

\ifEmFi@item
    268 \newif\ifEmFi@item

\ifEmFi@finished
    269 \newif\ifEmFi@finished

\ifEmFi@id
    270 \newif\ifEmFi@id

```

## 2.7 Key value definitions

```

271 \expandafter\ifx\csname define@key\endcsname\relax
272   \chardef\EmFi@plain=\z@
273   \def\EmFi@temp#1{%
274     \begingroup\expandafter\expandafter\expandafter\endgroup
275     \expandafter\ifx\csname#1\endcsname\relax
276       \chardef\EmFi@plain=\@ne
277       \fi
278   }%
279   \EmFi@temp{NeedsTeXFormat}%
280   \EmFi@temp{ProvidesPackage}%
281   \EmFi@temp{DeclareOption}%
282   \EmFi@temp{ExecuteOptions}%
283   \EmFi@temp{ProcessOptions}%
284   \ifnum\EmFi@plain=\@ne
285     \def\EmFi@temp#1{%
286       \expandafter\let\csname EmFi@Org#1\expandafter\endcsname
287         \csname#1\endcsname
288       \expandafter\def\csname#1\endcsname
289     }%
290     \EmFi@temp{NeedsTeXFormat}#1{}%
291     \EmFi@temp{ProvidesPackage}#1[#2]{}% hash-ok
292     \EmFi@temp{DeclareOption}#1{}%
293     \EmFi@temp{ExecuteOptions}#1{}%
294     \EmFi@temp{ProcessOptions}{}%

```

`\KV@errx` L<sup>A</sup>T<sub>E</sub>X's option processing is not available with plain-T<sub>E</sub>X. Thus we define the default error command `\KV@errx` here, also using package `infwarerr`'s `\@PackageError`.

```

295   \def\KV@errx#1{%
296     \@PackageError{keyval}{#1}\@ehc
297   }%

```

Other macros from L<sup>A</sup>T<sub>E</sub>X's kernel that are used by package `keyval`.

```

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

```

```

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

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

```

\EmFi@GlobalKey

```

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

```

\EmFi@GlobalDefaultKey

```

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

```

\EmFi@DefineKey

```

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

```

Subtype of the embedded file (optional).

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

File specification string.

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

File specification string in Unicode.

```
363 \EmFi@DefineKey{ucfilespec}{}
```

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\ltx@secondoftwo
391 \else
392 \expandafter\ltx@firstoftwo
393 \fi
394 }

```

\embedfilegetobject

```

395 \def\embedfilegetobject#1#2{%
396 \embedfileifobjectexists{#1}{#2}{%
397 \csname EmFi@#2@#1\endcsname
398 }{%
399 0 0 R%
400 }%
401 }

```

Initial view of the collection.

```

402 \define@key{EmFi}{view}[]{%
403 \EdefSanitize\EmFi@temp{#1}%
404 \def\EmFi@next{%
405 \global\EmFi@collectiontrue
406 }%
407 \ifx\EmFi@temp\ltx@empty
408 \let\EmFi@view\EmFi@S@details
409 \else\ifx\EmFi@temp\EmFi@S@details
410 \let\EmFi@view\EmFi@S@details
411 \else\ifx\EmFi@temp\EmFi@S@tile
412 \let\EmFi@view\EmFi@S@tile
413 \else\ifx\EmFi@temp\EmFi@S@hidden
414 \let\EmFi@view\EmFi@S@hidden
415 \else
416 \let\EmFi@next\relax
417 \EmFi@Error{%
418 Unknown value '\EmFi@temp' for key 'view'.\MessageBreak
419 Supported values: 'details', 'tile', 'hidden'.%
420 }\@ehc

```

```

421 \fi\fi\fi\fi
422 \EmFi@next
423 }
424 \EmFi@DefineKey{initialfile}{}

\embedfilesetup
425 \def\embedfilesetup{%
426   \ifEmFi@finished
427     \def\EmFi@next##1{%
428       \EmFi@Error{%
429         \string\embedfilefield\space after \string\embedfilefinish
430       }{%
431         The list of embedded files is already written.%
432       }%
433     \else
434       \def\EmFi@next{%
435         \setkeys{EmFi}%
436       }%
437     \fi
438   \EmFi@next
439 }

\EmFi@schema
440 \def\EmFi@schema{}

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

\EmFi@@order
442 \let\EmFi@@order\relax

\EmFi@fieldlist
443 \def\EmFi@fieldlist{}

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

\embedfilefield
445 \def\embedfilefield#1#2{%
446   \ifEmFi@finished
447     \EmFi@Error{%
448       \string\embedfilefield\space after \string\embedfilefinish
449     }{%
450       The list of embedded files is already written.%
451     }%
452   \else
453     \global\EmFi@collectiontrue
454     \EdefSanitize\EmFi@key{#1}%
455     \expandafter\ifx\csname KV@EmFi@\EmFi@key.prefix\endcsname\relax
456     \begingroup
457       \count@=\EmFi@order
458       \advance\count@ 1 %
459       \xdef\EmFi@order{\the\count@}%
460       \let\EmFi@title\EmFi@key
461       \let\EmFi@type\EmFi@S@text
462       \EmFi@visibletrue
463       \EmFi@editfalse
464       \setkeys{EmFiFi}{#2}%
465       \EmFi@convert\EmFi@title\EmFi@title
466       \xdef\EmFi@schema{%

```

```

467 \EmFi@schema
468 /\pdf@escapename{\EmFi@key}<<%
469 /Subtype/%
470 \ifx\EmFi@type\EmFi@S@date D%
471 \else\ifx\EmFi@type\EmFi@S@number N%
472 \else\ifx\EmFi@type\EmFi@S@file F%
473 \else\ifx\EmFi@type\EmFi@S@desc Desc%
474 \else\ifx\EmFi@type\EmFi@S@moddate ModDate%
475 \else\ifx\EmFi@type\EmFi@S@creationdate CreationDate%
476 \else\ifx\EmFi@type\EmFi@S@size Size%
477 \else S%
478 \fi\fi\fi\fi\fi\fi\fi
479 /N(\EmFi@title)%
480 \EmFi@@order{\EmFi@order}%
481 \ifEmFi@visible
482 \else
483 /V false%
484 \fi
485 \ifEmFi@edit
486 /E true%
487 \fi
488 >>%
489 }%
490 \let\do\relax
491 \xdef\EmFi@fieldlist{%
492 \EmFi@fieldlist
493 \do{\EmFi@key}%
494 }%
495 \ifx\EmFi@type\EmFi@S@text
496 \define@key{EmFi}{\EmFi@key.value}{%
497 \EmFi@itemtrue
498 \def\EmFi@temp{##1}%
499 \EmFi@convert\EmFi@temp\EmFi@temp
500 \expandafter\def\csname EmFi@V@#1%
501 \expandafter\endcsname\expandafter{%
502 \expandafter(\EmFi@temp)%
503 }%
504 }%
505 \EmFi@GlobalKey{EmFi}{\EmFi@key.value}%
506 \else\ifx\EmFi@type\EmFi@S@date
507 \define@key{EmFi}{\EmFi@key.value}{%
508 \EmFi@itemtrue
509 \def\EmFi@temp{##1}%
510 \EmFi@convert\EmFi@temp\EmFi@temp
511 \expandafter\def\csname EmFi@V@#1%
512 \expandafter\endcsname\expandafter{%
513 \expandafter(\EmFi@temp)%
514 }%
515 }%
516 \EmFi@GlobalKey{EmFi}{\EmFi@key.value}%
517 \else\ifx\EmFi@type\EmFi@S@number
518 \define@key{EmFi}{\EmFi@key.value}{%
519 \EmFi@itemtrue
520 \expandafter\edefSanitize\csname EmFi@V@#1\endcsname{ ##1}%
521 }%
522 \EmFi@GlobalKey{EmFi}{\EmFi@key.value}%
523 \fi\fi\fi
524 \define@key{EmFi}{\EmFi@key.prefix}{%
525 \EmFi@itemtrue
526 \expandafter\def\csname EmFi@P@#1\endcsname{##1}%
527 }%
528 \EmFi@GlobalKey{EmFi}{\EmFi@key.prefix}%

```



```

529     \define@key{EmFiSo}{\EmFi@key}[ascending]{%
530     \EdefSanitize\EmFi@temp{##1}%
531     \ifx\EmFi@temp\EmFi@S@ascending
532     \def\EmFi@temp{true}%
533     \else\ifx\EmFi@temp\EmFi@S@descending
534     \def\EmFi@temp{false}%
535     \else
536     \def\EmFi@temp{}%
537     \EmFi@Error{%
538     Unknown sort order '\EmFi@temp'.\MessageBreak
539     Supported values: '\EmFi@S@ascending', %
540     '\EmFi@S@descending
541     }\@ehc
542     \fi\fi
543     \ifx\EmFi@temp\ltx@empty
544     \else
545     \xdef\EmFi@sortkeys{%
546     \EmFi@sortkeys
547     /\pdf@escapename{#1}%
548     }%
549     \ifx\EmFi@sortorders\ltx@empty
550     \global\let\EmFi@sortorders\EmFi@temp
551     \gdef\EmFi@sortcase{1}%
552     \else
553     \xdef\EmFi@sortorders{%
554     \EmFi@sortorders
555     \space
556     \EmFi@temp
557     }%
558     \xdef\EmFi@sortcase{2}%
559     \fi
560     \fi
561     }%
562     \EmFi@GlobalDefaultKey{EmFiSo}\EmFi@key
563     \endgroup
564     \else
565     \EmFi@Error{%
566     Field '\EmFi@key' is already defined%
567     }\@ehc
568     \fi
569     \fi
570 }

571 \define@key{EmFiFi}{type}{%
572 \EdefSanitize\EmFi@temp{#1}%
573 \ifx\EmFi@temp\EmFi@S@text
574 \let\EmFi@type\EmFi@temp
575 \else\ifx\EmFi@temp\EmFi@S@date
576 \let\EmFi@type\EmFi@temp
577 \else\ifx\EmFi@temp\EmFi@S@number
578 \let\EmFi@type\EmFi@temp
579 \else\ifx\EmFi@temp\EmFi@S@file
580 \let\EmFi@type\EmFi@temp
581 \else\ifx\EmFi@temp\EmFi@S@desc
582 \let\EmFi@type\EmFi@temp
583 \else\ifx\EmFi@temp\EmFi@S@moddate
584 \let\EmFi@type\EmFi@temp
585 \else\ifx\EmFi@temp\EmFi@S@creationdate
586 \let\EmFi@type\EmFi@temp
587 \else\ifx\EmFi@temp\EmFi@S@size
588 \let\EmFi@type\EmFi@temp
589 \else
590 \EmFi@Error{%

```

```

591     Unknown type '\EmFi@temp'.\MessageBreak
592     Supported types: 'text', 'date', 'number', 'file',\MessageBreak
593     'desc', 'moddate', 'creationdate', 'size'%
594   }%
595 \fi\fi\fi\fi\fi\fi\fi
596 }

597 \define@key{EmFiFi}{title}{%
598   \def\EmFi@title{#1}%
599 }

```

\EmFi@setboolean

```

600 \def\EmFi@setboolean#1#2{%
601   \EdefSanitize\EmFi@temp{#2}%
602   \ifx\EmFi@temp\EmFi@S@true
603     \csname EmFi@#1true\endcsname
604   \else
605     \ifx\EmFi@temp\EmFi@S@false
606       \csname EmFi@#1false\endcsname
607     \else
608       \EmFi@Error{%
609         Unknown value '\EmFi@temp' for key '#1'.\MessageBreak
610         Supported values: 'true', 'false'%
611       } \@ehc
612     \fi
613   \fi
614 }

615 \define@key{EmFiFi}{visible}[true]{%
616   \EmFi@setboolean{visible}{#1}%
617 }

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

```

\EmFi@sortkeys

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

\EmFi@sortorders

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

\embedfilesort

```

623 \def\embedfilesort{%
624   \setkeys{EmFiSo}%
625 }

```

## 2.8 Embed the file

\embedfile

```

626 \def\embedfile{%
627   \@ifnextchar[\EmFi@embedfile{\EmFi@embedfile[]}%
628 }

```

\EmFi@embedfile

```

629 \def\EmFi@embedfile[#1]#2{%
630   \ifEmFi@finished
631     \EmFi@Error{%
632       \string\embedfile\space after \string\embedfilefinish
633     }{%
634       The list of embedded files is already written.%
635     }%

```

```

636 \else
637   \beginingroup
638     \def\EmFi@file{#2}%
639     \setkeys{EmFi}{#1}%
640     \expandafter\expandafter\expandafter
641     \ifx\expandafter\expandafter\expandafter
642       \\\pdf@filesize{\EmFi@file}\\%
643       \EmFi@Error{%
644         File '\EmFi@file' not found%
645       }{%
646         The unknown file is not embedded.%
647       }%
648     \else
649       \edef\EmFi@@filespec{%
650         \pdf@escapestring{\EmFi@filespec}%
651       }%
652       \ifx\EmFi@ucfilespec\ltx@empty
653         \let\EmFi@@ucfilespec\ltx@empty
654       \else
655         \EmFi@convert\EmFi@ucfilespec\EmFi@@ucfilespec
656       \fi
657       \ifx\EmFi@desc\ltx@empty
658         \let\EmFi@@desc\ltx@empty
659       \else
660         \EmFi@convert\EmFi@desc\EmFi@@desc
661       \fi
662       \ifEmFi@item
663         \let\do\EmFi@do
664         \immediate\pdfobj{%
665           <<%
666             \EmFi@fieldlist
667           >>%
668         }%
669         \edef\EmFi@ci{\the\pdf@lastobj}%
670       \fi
671       \immediate\pdfobj stream attr{%
672         /Type/EmbeddedFile%
673         \ifx\EmFi@mimetype\ltx@empty
674         \else
675           /Subtype/\pdf@escapename{\EmFi@mimetype}%
676         \fi
677         /Params<<%
678           /ModDate(\pdf@filemoddate{\EmFi@file})%
679           /Size \pdf@filesize{\EmFi@file}%
680           /Checksum<\pdf@filemdfivesum{\EmFi@file}>%
681         >>%
682       }file{\EmFi@file}\relax
683       \EmFi@defobj{EmbeddedFile}%
684       \immediate\pdfobj{%
685         <<%
686           /Type/Filespec%
687           \ifx\EmFi@filesystem\ltx@empty
688           \else
689             /FS/\pdf@escapename{\EmFi@filesystem}%
690           \fi
691           /F(\EmFi@@filespec)%
692           \ifx\EmFi@@ucfilespec\ltx@empty
693           \else
694             /UF(\EmFi@@ucfilespec)%
695           \fi
696           \ifx\EmFi@@desc\ltx@empty
697           \else

```

```

698         /Desc(\EmFi@@desc)%
699         \fi
700         /EF<<%
701         /F \the\pdflastobj\space 0 R%
702         >>%
703         \ifEmFi@item
704         /CI \EmFi@ci\space 0 R%
705         \fi
706         >>%
707     }%
708     \EmFi@defobj{Filespec}%
709     \EmFi@add{%
710         \EmFi@@filespec
711     }{\the\pdflastobj\space 0 R}%
712     \fi
713 \endgroup
714 \fi
715 }

```

\EmFi@do

```

716 \def\EmFi@do#1{%
717     \expandafter\ifx\csname EmFi@P@#1\endcsname\relax
718     \expandafter\ifx\csname EmFi@V@#1\endcsname\relax
719     \else
720         /\pdf@escapename{#1}\csname EmFi@V@#1\endcsname
721     \fi
722 \else
723     /\pdf@escapename{#1}<<%
724     \expandafter\ifx\csname EmFi@V@#1\endcsname\relax
725     \else
726         /D\csname EmFi@V@#1\endcsname
727     \fi
728     /P(\csname EmFi@P@#1\endcsname)%
729     >>%
730 \fi
731 }

```

\EmFi@convert

```

732 \def\EmFi@convert#1#2{%
733     \ifnum\pdf@strcmp{\EmFi@stringmethod}{psd}=0 %
734     \pdfstringdef\EmFi@temp{#1}%
735     \let#2\EmFi@temp
736 \else
737     \edef#2{\pdf@escapestring{#1}}%
738 \fi
739 }

```

```

740 \global\let\EmFi@list\ltx@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, ...).

```

741 \def\EmFi@add#1#2{%
742     \begingroup
743     \ifx\EmFi@list\ltx@empty
744         \xdef\EmFi@list{\noexpand\do{#1}{#2}}%
745     \else
746         \def\do##1##2{%
747             \ifnum\pdf@strcmp{##1}{#1}>0 %
748                 \edef\x{%
749                     \toks@{%

```

```

750         \the\toks@%
751         \noexpand\do{#1}{#2}%
752         \noexpand\do{##1}{##2}%
753     }%
754 }%
755 \x
756 \def\do###1###2{%
757     \toks@\expandafter{\the\toks@\do{###1}{###2}}%
758 }%
759 \def\stop{%
760     \xdef\EmFi@list{\the\toks@}%
761 }%
762 \else
763     \toks@\expandafter{\the\toks@\do{##1}{##2}}%
764 \fi
765 }%
766 \def\stop{%
767     \xdef\EmFi@list{\the\toks@\noexpand\do{#1}{#2}}%
768 }%
769 \toks@{}%
770 \EmFi@list\stop
771 \fi
772 \endgroup
773 }

```

\embedfilefinish

```

774 \def\embedfilefinish{%
775     \ifEmFi@finished
776         \EmFi@Error{%
777             Too many invocations of \string\embedfilefinish
778         }%
779         The list of embedded files is already written.%
780     }%
781 \else
782     \ifx\EmFi@list\ltx@empty
783     \else
784         Write /EmbeddedFiles entry.
785         \global\EmFi@finishedtrue
786         \begingroup
787         \def\do##1##2{%
788             (##1)##2%
789         }%
790         \immediate\pdfobj{%
791             <<%
792             /Names[\EmFi@list]%
793             >>%
794         }%
795         \pdfnames{%
796             /EmbeddedFiles \the\pdflastobj\space 0 R%
797         }%
798     \endgroup
799     Write collection objects.
800     \ifx\EmFi@initialfile\ltx@empty
801     \else
802         \EmFi@collectiontrue
803     \fi
804     \ifEmFi@collection
805         \ifx\EmFi@initialfile\ltx@empty
806             \let\EmFi@initialfile\ltx@empty
807         \else
808             \edef\EmFi@@initialfile{%

```

```

807         \pdf@escapestring{\EmFi@initialfile}%
808     }%
809     \fi

Look for initial file among the embedded files.

810     \begingroup
811     \let\f=N%
812     \def\do##1##2{%
813         \def\x{##1}%
814         \ifx\x\EmFi@initialfile
815             \let\f=Y%
816             \let\do\ltx@gobbletwo
817         \fi
818     }%
819     \EmFi@list
820 \expandafter\endgroup
821 \ifx\f Y%
822 \else
823     \@PackageWarningNoLine{embedfile}{%
824         Missing initial file '\EmFi@initialfile'\MessageBreak
825         among the embedded files%
826     }%
827     \let\EmFi@initialfile\ltx@empty
828     \let\EmFi@initialfile\ltx@empty
829 \fi
830 \ifcase\EmFi@sortcase
831     \def\EmFi@temp{%
832 \or
833     \def\EmFi@temp{%
834         /S\EmFi@sortkeys
835         /A \EmFi@sortorders
836     }%
837 \else
838     \def\EmFi@temp{%
839         /S[\EmFi@sortkeys]%
840         /A[\EmFi@sortorders]%
841     }%
842 \fi
843 \def\EmFi@order##1{%
844     \ifnum\EmFi@order>1 %
845         /O ##1%
846     \fi
847 }%
848 \immediate\pdfobj{%
849     <<%
850         \ifx\EmFi@schema\ltx@empty
851         \else
852             /Schema<<\EmFi@schema>>%
853         \fi
854         \ifx\EmFi@initialfile\ltx@empty
855         \else
856             /D(\EmFi@initialfile)%
857         \fi
858         \ifx\EmFi@view\EmFi@S@tile
859             /View/T%
860         \else\ifx\EmFi@view\EmFi@S@hidden
861             /View/H%
862         \fi\fi
863         \ifx\EmFi@temp\ltx@empty
864             \EmFi@temp
865         \else
866             /Sort<<\EmFi@temp>>%
867         \fi

```

```

868         >>%
869     }%
870     \pdfcatalog{%
871         /Collection \the\pdflastobj\space0 R%
872     }%
873     \fi
874 \fi
875 \fi
876 }

877 \begingroup\expandafter\expandafter\expandafter\endgroup
878 \expandafter\ifx\csname AtEndDocument\endcsname\relax
879 \else
880 \AtEndDocument{\embedfilefinish}%
881 \fi

882 \EmFi@AtEnd
883 </package>

```

## 3 Test

### 3.1 Catcode checks for loading

```

884 (*test1)

885 \catcode'\{=1 %
886 \catcode'\}=2 %
887 \catcode'\#=6 %
888 \catcode'\@=11 %
889 \expandafter\ifx\csname count@\endcsname\relax
890 \countdef\count@=255 %
891 \fi
892 \expandafter\ifx\csname @gobble\endcsname\relax
893 \long\def\@gobble#1{}%
894 \fi
895 \expandafter\ifx\csname @firstofone\endcsname\relax
896 \long\def\@firstofone#1{#1}%
897 \fi
898 \expandafter\ifx\csname loop\endcsname\relax
899 \expandafter\@firstofone
900 \else
901 \expandafter\@gobble
902 \fi
903 {%
904 \def\loop#1\repeat{%
905 \def\body{#1}%
906 \iterate
907 }%
908 \def\iterate{%
909 \body
910 \let\next\iterate
911 \else
912 \let\next\relax
913 \fi
914 \next
915 }%
916 \let\repeat=\fi
917 }%
918 \def\RestoreCatcodes{}
919 \count@=0 %
920 \loop
921 \edef\RestoreCatcodes{%
922 \RestoreCatcodes

```

```

923   \catcode\the\count@=\the\catcode\count@\relax
924 }%
925 \ifnum\count@<255 %
926   \advance\count@ 1 %
927 \repeat
928
929 \def\RangeCatcodeInvalid#1#2{%
930   \count@=#1\relax
931   \loop
932     \catcode\count@=15 %
933   \ifnum\count@<#2\relax
934     \advance\count@ 1 %
935   \repeat
936 }
937 \expandafter\ifx\csname LoadCommand\endcsname\relax
938 \def\LoadCommand{\input embedfile.sty\relax}%
939 \fi
940 \def\Test{%
941   \RangeCatcodeInvalid{0}{47}%
942   \RangeCatcodeInvalid{58}{64}%
943   \RangeCatcodeInvalid{91}{96}%
944   \RangeCatcodeInvalid{123}{255}%
945   \catcode'\@=12 %
946   \catcode'\=0 %
947   \catcode'\{=1 %
948   \catcode'\}=2 %
949   \catcode'\#=6 %
950   \catcode'\ [=12 %
951   \catcode'\]=12 %
952   \catcode'\%=14 %
953   \catcode'\ =10 %
954   \catcode13=5 %
955   \LoadCommand
956   \RestoreCatcodes
957 }
958 \Test
959 \csname @@end\endcsname
960 \end
961 </test1>

```

### 3.2 Simple test

```

962 (*test2)
963 \input embedfile.sty\relax
964 \embedfile[%
965   stringmethod=escape,%
966   mimetype=plain/text,%
967   desc={LaTeX docstrip source archive for package 'embedfile'},%
968   id={embedfile.dtx}%
969 ]{embedfile.dtx}
970 \nopagenumbers
971 Test (plain-TeX): {\tt embedfile.dtx} should be embedded.%
972
973 \def\Test#1{%
974   \par
975   \embedfileifobjectexists{embedfile.dtx}{#1}{%
976     Object #1 (embedfile.dtx): %
977     \embedfilegetobject{embedfile.dtx}{#1}%
978   }{%
979     \errmessage{Missing object #1 (embedfile.dtx)}%
980   }%
981 }
982 \Test{EmbeddedFile}

```



```

983 \Test{Filespec}
984 \embedfilefinish
985 \bye
986 </test2>

987 /*test3>
988 \NeedsTeXFormat{LaTeX2e}
989 \let\SavedJobname\jobname
990 \def\jobname{embedfile}
991 \RequirePackage{dtx-attach}[2009/09/25]
992 \let\jobname\SavedJobname
993 \documentclass{minimal}
994 \begin{document}
995 Test (\LaTeX): \texttt{embedfile.dtx} should be embedded.%
996 \end{document}
997 </test3>

```

## 4 Installation

### 4.1 Download

**Package.** This package is available on CTAN<sup>1</sup>:

[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.

### 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
```

---

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

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

```

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

```

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

#### 4.4 Refresh file name databases

If your  $\TeX$  distribution (te $\TeX$ , mi $\TeX$ , ...) relies on file name databases, you must refresh these. For example, te $\TeX$  users run `texhash` or `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 pdf $\LaTeX$ :

```

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](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<sup>A</sup>T<sub>E</sub>X support.

### [2007/11/25 v2.3]

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

### [2009/09/25 v2.4]

- Bug fix: If `hyperref` is used with option `unicode`, the Unicode encoded file name causes trouble. Therefore `\pdfstringdef` is now never used for option `filespec`, always method `escape` is applied (Peter Cibulka).
- Bug fix for `initialfile`.
- Bug fix for file names in `/EmbeddedFiles`.
- New option `ucfilespec` for file name support in Unicode (since PDF 1.7).

## 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	
\#	887, 949
\%	952
\:	318, 319, 320, 321
\@	888, 945
\@PackageError	219, 296
\@PackageWarningNoLine	823
\@ehc	296, 420, 541, 567, 611
\@firstofone	896, 899
\@gobble	893, 901
\@ifnch	303, 305, 322
\@ifnextchar	298, 627
\@let@token	303, 306, 309, 322
\@namedef	326
\@ne	276, 284, 333
\@sptoken	306, 318, 319
\@undefined	162, 367
\@xifnch	307, 320
\[	950
\\	642, 946
\{	885, 947
\}	886, 948
\]	951
\_	953
A	
\advance	458, 926, 934
\aftergroup	136
\AtEndDocument	880
B	
\begin	75, 994
\body	905, 909
\bye	35, 985
C	
\catcode	113, 114, 115, 116, 117, 118, 119, 127, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 174, 175, 178, 179, 180, 181, 185, 186, 187, 188, 192, 194, 885, 886, 887, 888, 923, 932, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954
\chardef	272, 276
\count@	457, 458, 459, 890, 919, 923, 925, 926, 930, 932, 933, 934
\countdef	890
\csname	120, 128, 154, 170, 177, 210, 236, 247, 271, 275, 286, 287, 288, 298, 326, 328, 335, 336, 346, 347, 352, 353, 357, 359, 383, 389, 397, 455, 500, 511, 520, 526, 603, 606,
	717, 718, 720, 724, 726, 728, 878, 889, 892, 895, 898, 937, 959
D	
\define@key	356, 377, 402, 496, 507, 518, 524, 529, 571, 597, 615, 618
\do	490, 493, 663, 744, 746, 751, 752, 756, 757, 763, 767, 786, 812, 816
\documentclass	41, 993
E	
\EdefSanitize	247, 403, 454, 520, 530, 572, 601
\embedfile	3, 16, 21, 25, 79, 85, 91, 105, <u>626</u> , 632, 964
\embedfilefield	4, 50, 54, 58, 62, 66, 429, <u>445</u>
\embedfilefinish	3, 34, 429, 448, 632, <u>774</u> , 880, 984
\embedfilegetobject	5, <u>395</u> , 977
\embedfileifobjectexists	5, <u>388</u> , 396, 975
\embedfilessetup	3, 4, 11, 46, <u>425</u>
\embedfilesort	5, 71, <u>623</u>
\EmFi@desc	658, 660, 696, 698
\EmFi@filespec	649, 691, 710
\EmFi@initialfile	804, 806, 814, 828, 854, 856
\EmFi@order	442, 480, 843
\EmFi@ucfilespec	653, 655, 692, 694
\EmFi@add	709, <u>741</u>
\EmFi@AtEnd	190, 191, 230, 242, 882
\EmFi@ci	669, 704
\EmFi@collectiontrue	405, 453, 800
\EmFi@convert	465, 499, 510, 655, 660, <u>732</u>
\EmFi@DefineKey	355, 361, 362, 363, 364, 365, 366, 424
\EmFi@defobj	381, 683, 708
\EmFi@desc	657, 660
\EmFi@details	<u>249</u>
\EmFi@do	663, <u>716</u>
\EmFi@editfalse	463
\EmFi@embedfile	627, <u>629</u>
\EmFi@Error	217, 224, 237, 417, 428, 447, 537, 565, 590, 608, 631, 643, 776
\EmFi@fieldlist	443, 491, 492, 666
\EmFi@file	362, 638, 642, 644, 678, 679, 680, 682
\EmFi@filespec	650
\EmFi@filesystem	687, 689
\EmFi@finishedtrue	784
\EmFi@GlobalDefaultKey	349, 562
\EmFi@GlobalKey	345, 350, 505, 516, 522, 528



