# The epstopdf package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2008/05/06 v1.7

**Abstract**

This packages adds support of handling eps images to package graphics or graphicx with option pdftex. If an eps image is detected, epstopdf is automatically called to convert it to pdf format.

## Contents

# 1 Documentation

## 1.1 Introduction

LaTeX provides its graphics bundle to include graphics files. Both packages graphics or graphicx may be used. the latter one loads the first and adds options in key value style for `\includegraphics`.

Usually the drivers do not support all kind of graphics files. Other image types must be converted, before they become usuable. In case of driver dvips, the graphics rule may contain a conversion rule. Then all that package graphics must know is the bounding box, the command is passed to dvips that calls it and embeds the converted image.

However, pdfTeX has its driver for PDF output already build in. It's graphics inclusion commands (`\pdfximage`) does not allow the execution of external commands. Therefore commands in the last argument of `\DeclareGraphicsRule` were of no use. But external programs can be calles within pdfTeX. This feature is called "shell escape" or "write 18" and must usually enabled explicitely because of security reasons. Now, this package epstopdf hooks into package graphics' code to catch that argument with the external command and executes it to convert the graphics file to a supported format and passes the control of graphics inclusion back to package graphics.

## 1.2 Requirements

- The feature `\write18` must be enabled. This allows the running of external programs during TeX's compile run. Keep in mind that this is a security risk. The feature is an addition to `\TeX`. MikTeX, teTeX, TeX Live support it. In Web2C based TeX distributions (teTeX, TeX Live) it can be enabled in the configuration file `texmf.cnf`:

  ```
  shell_escape = 1
  ```

  Because of the security risk, it is better to do it on the command line only:

  `--shell-escape` (teTeX, TeX Live)
  `--enable-write18` (MiKTeX)

  Example:

  ```
  pdflatex -shell-escape test.tex
  ```

- The program epstopdf for the conversion from EPS to PDF. However, other programs can be used and configured by `\DeclareGraphicsRule`. Example:

  ```
  \DeclareGraphicsRule{.eps}{pdf}{.pdf}{%
    `ps2pdf -dEPSCrop #1 \OutputFile
  }
  ```

## 1.3 Usage

The package is loaded after graphic{s,x}, e.g.:

```
\usepackage[pdftex]{graphicx}
\usepackage{epstopdf}
```

Now images with file name extension `.eps` are detected and supported using `\includegraphics`.

If the graphics file name is explicitly specified with extension `.eps` the new rule for EPS files is called and the conversion performed. If option update is in force then the conversion step is dropped if the target file already exists and is not older then the EPS file.

The situation is more complicate if the graphics file is given without file name extension. Then the graphics package must search for a supported image file. The possible extensions are stored in the graphics extension list, that can be set by `\DeclareGraphicsExtensions`. The algorithm:

function search( ⟨*filebase*⟩ ) foreach ⟨*ext*⟩ in ⟨*graphics extensions*⟩
  foreach ⟨*dir*⟩ in ⟨*current directory*⟩, ⟨`\graphicspath`⟩
    ⟨*file*⟩ := ⟨*dir*⟩ + ⟨*filebase*⟩ + ⟨*ext*⟩
    if exist ⟨*file*⟩ return found return not found

Package epstopdf puts `.eps` at the end of the graphics extension list. This is the behaviour of option append that is enabled by default. That means, the conversion is called last unless a supported file type cannot be found earlier. This avoids unnecessary conversion steps that slow down the LaTeX run. If you want to use option update and your pdfTeX supports it, then an outdated PDF file also would be found earlier. Therefore extension `.eps` should be put in front of the list. This is achieved by option prepend. Then the EPS file is found before the PDF file. Then option update have the control and can compare file dates.

Note: Usually the conversion program needs the exact location of the image file. Usually the current directory works. Also if the image file is found using `\graphicspath`, the location is known. However, if the image is somewhere in a directory of environment variable `TEXINPUTS`, then the package does not know the exact location and the conversion program will not find the image file unless it implements a search using `TEXINPUTS` (program kpsewhich may be of help in this task).

## 1.4 Options

Options can be given as package options or later using:

`\epstopdfsetup` {⟨*key value list*⟩}

**update:** The conversion program is only called, if the target file does not exist or is older than the source image file.

**append:** Puts the extension `.eps` at the end of the graphics extension list (default).

**prepend:** Puts the extension `.eps` at the begin of the graphics extension list.

**outdir:** The converted file may put in an other output directory. The value of outdir must include the directory separator. Example for the current directory:

```
\epstopdfsetup{outdir=./}
```

For other directories ensure, that they can be found. See `\graphicspath` or `TEXINPUTS`.

**verbose:** It prints some information about the image in the `.log` file.

## 1.5 Configuration

### 1.5.1 Configuration file

A configuration file `epstopdf.cfg` is loaded at the end of the package if it exists. It can be used for changing the default option setting. My favourite setting is:

```
\epstopdfsetup{update,prepend,verbose}
```

### 1.5.2 Conversion program

You can use `\DeclareGraphicsRule` the same way as the route via dvips to specify the conversion command line, examples below.

Additionally you can use the following macros:

`\OutputFile`: : output file name (with known path and extension)

`\SourceFile`: : source file name (with known path and extension), usually the same as `#1`.

**Conversion from EPS to PDF.** Other programs than epstopdf can be used to convert from EPS to PDF. Example that uses Ghostscript:

```
\DeclareGraphicsRule{.eps}{pdf}{.pdf}{%
  `ps2pdf -dEPSCrop #1 \noexpand\OutputFile
}
```

`\DeclareGraphicsRule` expands the argument, therefore `\noexpand` is necessary. Also `\OutputFile` respects the setting of option outdir.

## 1.6 Other image formats

The support that package epstopdf implements is not limited to EPS files. Other image conversions can be declared. The following example shows it for GIF images under Unix with ImageMagick's convert:

```
\DeclareGraphicsRule{.gif}{png}{.png}{%
  `convert #1 \noexpand\OutputFile
}
```

The file extension `.gif` can be added to the extension list that package graphics searches if the file extension is not given in `\includegraphics`. The list can be set by `\GraphicsExtensions`.

```
\AppendGraphicsExtensions{.gif}
```
or
```
\PrependGraphicsExtensions{.gif}
```

# 2 Implementation

1 ⟨∗package⟩

## 2.1 Preparations

### 2.1.1 Relead check and identification

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

```
2 \begingroup
3   \catcode44 12 % ,
4   \catcode45 12 % -
5   \catcode46 12 % .
6   \catcode58 12 % :
7   \catcode64 11 % @
```

```
  \expandafter\let\expandafter\x\csname ver@epstopdf.sty\endcsname
  \ifcase 0%
    \ifx\x\relax % plain
    \else
      \ifx\x\empty % LaTeX
      \else
        1%
      \fi
    \fi
  \else
    \catcode35 6 % #
    \catcode123 1 % {
    \catcode125 2 % }
    \expandafter\ifx\csname PackageInfo\endcsname\relax
      \def\x#1#2{%
        \immediate\write-1{Package #1 Info: #2.}%
      }%
    \else
      \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
    \fi
    \x{epstopdf}{The package is already loaded}%
    \endgroup
    \expandafter\endinput
  \fi
\endgroup
```

Package identification:

```
\begingroup
  \catcode35 6 % #
  \catcode40 12 % (
  \catcode41 12 % )
  \catcode44 12 % ,
  \catcode45 12 % -
  \catcode46 12 % .
  \catcode47 12 % /
  \catcode58 12 % :
  \catcode64 11 % @
  \catcode123 1 % {
  \catcode125 2 % }
  \expandafter\ifx\csname ProvidesPackage\endcsname\relax
    \def\x#1#2#3[#4]{\endgroup
      \immediate\write-1{Package: #3 #4}%
      \xdef#1{#4}%
    }%
  \else
    \def\x#1#2[#3]{\endgroup
      #2[{#3}]%
      \ifx#1\relax
        \xdef#1{#3}%
      \fi
    }%
  \fi
\expandafter\x\csname ver@epstopdf.sty\endcsname
\ProvidesPackage{epstopdf}%
  [2008/05/06 v1.7 Conversion with epstopdf on the fly (HO)]
```

### 2.1.2 Catcodes

```
\begingroup
  \catcode123 1 % {
  \catcode125 2 % }
  \def\x{\endgroup
    \expandafter\edef\csname ETE@AtEnd\endcsname{%
```

```
      \catcode35 \the\catcode35\relax
      \catcode64 \the\catcode64\relax
      \catcode123 \the\catcode123\relax
      \catcode125 \the\catcode125\relax
    }%
  }%
\x
\catcode35 6 % #
\catcode64 11 % @
\catcode123 1 % {
\catcode125 2 % }
\def\TMP@EnsureCode#1#2{%
  \edef\ETE@AtEnd{%
    \ETE@AtEnd
    \catcode#1 \the\catcode#1\relax
  }%
  \catcode#1 #2\relax
}
\TMP@EnsureCode{33}{12}% !
\TMP@EnsureCode{39}{12}% '
\TMP@EnsureCode{42}{12}% *
\TMP@EnsureCode{44}{12}% ,
\TMP@EnsureCode{45}{12}% -
\TMP@EnsureCode{46}{12}% .
\TMP@EnsureCode{47}{12}% /
\TMP@EnsureCode{58}{12}% :
\TMP@EnsureCode{60}{12}% <
\TMP@EnsureCode{61}{12}% =
\TMP@EnsureCode{62}{12}% >
\TMP@EnsureCode{96}{12}% `
```

### 2.1.3 Load packages

```
\RequirePackage{infwarerr}[2007/09/09]
\RequirePackage{grfext}\relax
\RequirePackage{kvoptions}[2007/10/02]
\RequirePackage{pdftexcmds}[2007/11/11]
```

## 2.2 Checks

Check, whether package graphics is loaded (also graphicx loads graphics). Because miniltx.tex does not know `\@ifpackageloaded` we test for `\Gin@setfile` instead.

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname Gin@setfile\endcsname\relax
  \@PackageWarningNoLine{epstopdf}{%
    No graphics package \string`graphic{s,x}\string' loaded%
  }%
  \newcommand*{\epstopdfsetup}[1]{}%
  \ETE@AtEnd
  \expandafter\endinput
\fi
```

Check, whether pdftex.def is loaded. `\ver@pdftex.def` is not available with miniltx.tex, thus we test for `\Gin@driver`.

```
\begingroup
  \def\x{pdftex.def}%
  \ifx\Gin@driver\x
  \else
    \@PackageWarningNoLine{epstopdf}{%
      Graphics driver file \string`pdftex.def\string' not loaded%
    }%
    \endgroup
    \newcommand*{\epstopdfsetup}[1]{}%
    \ETE@AtEnd
```

```
    \expandafter\endinput
  \fi
\endgroup
```

Check, whether the shell escape feature is enabled.

```
\begingroup
  \expandafter\ifx\csname pdf@shellescape\endcsname\relax
  \else
    \ifnum\pdf@shellescape>0 %
    \else
      \@PackageWarningNoLine{epstopdf}{%
        Shell escape feature is not enabled%
      }%
    \fi
  \fi
\endgroup
```

## 2.3 Package loading

## 2.4 Options

```
\SetupKeyvalOptions{family=ETE,prefix=ETE@}
\DeclareBoolOption{update}
\DeclareBoolOption{verbose}
\newif\ifETE@prepend
\DeclareVoidOption{prepend}{\ETE@prependtrue}
\DeclareVoidOption{append}{\ETE@prependfalse}
\DeclareStringOption{outdir}
\ProcessKeyvalOptions*
\newcommand*{\epstopdfsetup}{\setkeys{ETE}}
```

## 2.5 Make and verbose

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname pdf@filemoddate\endcsname\relax
  \def\ETE@Make#1#2{%
    \ifETE@update
      \ETE@WarnModDate
    \fi
    \@firstofone
  }%
  \def\ETE@WarnModDate{%
    \@PackageWarningNoLine{epstopdf}{%
      \string\pdffilemoddate\space is not available,\MessageBreak
      option 'update' will be ignored%
    }%
    \global\let\ETE@WarnModDate\relax
  }%
  \def\ETE@FileInfo#1#2{#1 file: <#2>}%
\else
  \def\ETE@Make#1#2{%
    \ifETE@update
      \ifnum\pdf@strcmp{\pdf@filemoddate{#1}}{\pdf@filemoddate{#2}}>0 %
        \expandafter\expandafter\expandafter\@firstofone
      \else
        \@PackageInfoNoLine{epstopdf}{%
          Output file is already uptodate%
        }%
        \expandafter\expandafter\expandafter\@gobble
      \fi
    \else
      \expandafter\@firstofone
    \fi
```

```
  }%
  \def\ETE@FileInfo#1#2{%
    #1 file: <#2>%
    \expandafter\expandafter\expandafter
    \ETE@Date\pdf@filemoddate{#2}\@nil
    \expandafter\expandafter\expandafter
    \ETE@Size\pdf@filesize{#2}\@nil
  }%
  \def\ETE@Date#1\@nil{%
    \ifx\\#1\\%
    \else
      \ETE@@Date#1\@nil
    \fi
  }%
  \def\ETE@@Date#1:#2#3#4#5#6#7#8#9{%
    \MessageBreak
    \@spaces\space\space\space date: #2#3#4#5-#6#7-#8#9 %
    \ETE@@Time
  }%
  \def\ETE@@Time#1#2#3#4#5#6#7\@nil{%
    #1#2:#3#4:#5#6%
  }%
  \def\ETE@Size#1\@nil{%
    \ifx\\#1\\%
    \else
      \MessageBreak
      \@spaces\space\space\space size: #1 bytes%
    \fi
  }%
\fi
```

## 2.6 Adding conversion support

Patch `\Gin@setfile` to execute `#3`, if it contains a command.

```
\expandafter\ifx\csname ETE@OrgGin@setfile\endcsname\relax
  \let\ETE@OrgGin@setfile\Gin@setfile
\else
  \@PackageError{epstopdf}{%
    Command \string\ETE@OrgGin@setfile\space
    already defined.\MessageBreak
  }{%
    Probably some package has included the code of this package%
    \MessageBreak
    instead of using \string\RequirePackage{epstopdf}.%
    \MessageBreak
    \@ehc
  }%
\fi
\def\Gin@setfile#1#2#3{%
  \if‘\@car #3\relax\@nil
    \begingroup
      \def\GraphicsType{#1}%
      \def\GraphicsRead{#2}%
      \ifx\Gin@ext\relax
        \def\SourceFile{\Gin@base\Gin@eext}%
      \else
        \def\SourceFile{\Gin@base\Gin@ext}%
      \fi
      \let\OutputDirectory\ETE@outdir
      \ifx\OutputDirectory\@empty
        \def\OutputFile{\Gin@base#2}%
      \else
        \begingroup
```

```
          \filename@parse{\Gin@base#2}%
          \edef\x{\endgroup
            \def\noexpand\OutputFile{%
              \OutputDirectory\filename@base#2%
            }%
          }%
        \x
      \fi
      \edef\CommandLine{\@cdr#3\@empty\@nil}%
      \ifETE@verbose
        \@PackageInfo{epstopdf}{%
          \ETE@FileInfo{Source}\SourceFile\MessageBreak
          \ETE@FileInfo{Output}\OutputFile\MessageBreak
          Command: <\CommandLine>\MessageBreak
          \string\includegraphics
        }%
      \fi
      \ETE@Make\SourceFile\OutputFile{%
        \pdf@system{\CommandLine}%
        \ifETE@verbose
          \@PackageInfoNoLine{epstopdf}{%
            \ETE@FileInfo{Result}\OutputFile
          }%
        \fi
      }%
      \edef\x{\endgroup
        \ifx\OutputDirectory\@empty
        \else
          \def\noexpand\Gin@base{%
            \OutputDirectory\noexpand\filename@base
          }%
        \fi
        \noexpand\ETE@OrgGin@setfile{%
          \GraphicsType
        }{%
          \GraphicsRead
        }{%
          \OutputFile
        }%
      }%
    \x
  \else
    \ETE@OrgGin@setfile{#1}{#2}{#3}%
  \fi
}
```

### `\DeclareGraphicsRule` for .eps

```
\expandafter\ifx\csname Gin@rule@.eps\endcsname\relax
\else
  \@PackageInfo{epstopdf}{Overwriting graphics rule for `.eps'}%
\fi
\@namedef{Gin@rule@.eps}#1{{pdf}{.pdf}{`\ETE@epstopdf{#1}}}
\def\ETE@epstopdf#1{%
  epstopdf %
  \ifx\OutputDirectory\@empty
  \else
    --outfile=\OutputFile\space
  \fi
  #1%
}
\ifETE@prepend
  \expandafter\PrependGraphicsExtensions
\else
```

```
  \expandafter\AppendGraphicsExtensions
\fi
{.eps}
\let\ETE@prepend\@undefined
\DeclareVoidOption{prepend}{%
  \PrependGraphicsExtensions{.eps}%
}
\let\ETE@append\@undefined
\DeclareVoidOption{append}{%
  \AppendGraphicsExtensions{.eps}%
}
\InputIfFileExists{epstopdf.cfg}{}{}
\ETE@AtEnd
⟨/package⟩
```

# 3 Test

## 3.1 Preface for standard catcode check

```
⟨*test1⟩
\input miniltx.tex\relax
\def\Gin@driver{pdftex.def}
\input graphicx.sty\relax
\resetatcatcode
⟨/test1⟩
```

## 3.2 Catcode checks for loading

```
⟨*test1⟩
\catcode`\{=1 %
\catcode`\}=2 %
\catcode`\#=6 %
\catcode`\@=11 %
\expandafter\ifx\csname count@\endcsname\relax
  \countdef\count@=255 %
\fi
\expandafter\ifx\csname @gobble\endcsname\relax
  \long\def\@gobble#1{}%
\fi
\expandafter\ifx\csname @firstofone\endcsname\relax
  \long\def\@firstofone#1{#1}%
\fi
\expandafter\ifx\csname loop\endcsname\relax
  \expandafter\@firstofone
\else
  \expandafter\@gobble
\fi
{%
  \def\loop#1\repeat{%
    \def\body{#1}%
    \iterate
  }%
  \def\iterate{%
    \body
      \let\next\iterate
    \else
      \let\next\relax
    \fi
    \next
  }%
  \let\repeat=\fi
```

```
}%
\def\RestoreCatcodes{}
\count@=0 %
\loop
  \edef\RestoreCatcodes{%
    \RestoreCatcodes
    \catcode\the\count@=\the\catcode\count@\relax
  }%
\ifnum\count@<255 %
  \advance\count@ 1 %
\repeat

\def\RangeCatcodeInvalid#1#2{%
  \count@=#1\relax
  \loop
    \catcode\count@=15 %
  \ifnum\count@<#2\relax
    \advance\count@ 1 %
  \repeat
}
\expandafter\ifx\csname LoadCommand\endcsname\relax
  \def\LoadCommand{\input epstopdf.sty\relax}%
\fi
\def\Test{%
  \RangeCatcodeInvalid{0}{47}%
  \RangeCatcodeInvalid{58}{64}%
  \RangeCatcodeInvalid{91}{96}%
  \RangeCatcodeInvalid{123}{255}%
  \catcode`\@=12 %
  \catcode`\\=0 %
  \catcode`\{=1 %
  \catcode`\}=2 %
  \catcode`\#=6 %
  \catcode`\[=12 %
  \catcode`\]=12 %
  \catcode`\%=14 %
  \catcode`\ =10 %
  \catcode13=5 %
  \LoadCommand
  \RestoreCatcodes
}
\Test
\csname @@end\endcsname
\end
```

⟨/test1⟩

# 4 Installation

## 4.1 Download

**Package.** This package is available on CTAN[1]:

CTAN:macros/latex/contrib/oberdiek/epstopdf.dtx The source file.

CTAN:macros/latex/contrib/oberdiek/epstopdf.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

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

*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 epstopdf.dtx
```

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

```
epstopdf.sty            → tex/latex/oberdiek/epstopdf.sty
epstopdf.pdf            → doc/latex/oberdiek/epstopdf.pdf
test/epstopdf-test1.tex → doc/latex/oberdiek/test/epstopdf-test1.tex
epstopdf.dtx            → source/latex/oberdiek/epstopdf.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 (teTEX, mikTEX, ...) relies on file name databases, you must refresh these. For example, teTEX 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 epstopdf.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{epstopdf.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 pdfLaTeX:

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

# 5 History

## [2001/01/06 v1.0]

- First public version, published in the pdfTeX mailing list.

## [2001/02/04 v1.1]

- Minor documentation update.
- CTAN.

## [2006/02/20 v1.2]

- DTX framework.
- Compatibility for `miniltx.tex`.

## [2006/08/26 v1.3]

- Check for `\write18` if available and print a warning if the feature is not enabled.

## [2007/04/26 v1.4]

- Documentation rewritten and extended.

## [2007/10/02 v1.5]

- New option update: If the converted file exists, it will be only converted if it is out of date.
- Updating the extension list is delegated to package grfext. Fine tuning is done by the new options append, prepend.
- New option outdir for changing the output directory.
- New option verbose.
- `\SourceFile` and `\OutputFile` introduced.
- Configuration file support added.

## [2007/11/11 v1.6]

- Use of package pdftexcmds for LuaTeX support.

### [2008/05/06 v1.7]

- Warning messages uses “loaded” instead of “found”.

# 6 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.

### L



### M



### N



### O



### P



### R



### S



### T



### W



### X