Epspdf
Easy conversion between PostScript en Pdf

Abstract
This article introduces epspdf, a converter between eps, PostScript and pdf which can be run either via a graphical interface or from the command-line.

Keywords
Pdf, PostScript, eps, Ruby, Tk, cropping, page selection, Ghostscript, pdftops

Introduction
When preparing a LaTeX document, it is often convenient to have graphics available both in eps- and in pdf format. Epspdf[1] improves on previous solutions by having both a CLI or command-line interface and a GUI, by converting both ways, using pdftops from the xpdf suite[2], and by various new options, which were made possible by round-tripping between PostScript and pdf.

Sample applications
Case 1: Converting a Powerpoint slide to pdf and eps. A.U. Thor writes a paper in LaTeX and creates his illustrations with PowerPoint. He likes to turn these into pdf graphics, so that he can process his paper with pdflatex.

From PowerPoint, he 'prints' to an eps file (see the appendix). The Windows Print dialog is rather insistent on giving the eps file an extension '.prn'. He loads the graphic in epspdftk, where the .prn file is accurately identified as eps. He checks the 'Compute tight boundingbox' option, selects pdf output format, and clicks 'Convert and save'. Some annoying black boxes flit across his screen, but soon a message 'Conversion completed' appears. He presses the 'View' button and Adobe Reader displays what he hoped to see.

He might as well replace the eps with one with a good boundingbox, so he converts the pdf back to eps. It may save epspdf some work if he first unchecks the boundingbox checkbox.

When viewing the resulting eps with GSview, there is once more a lot of whitespace around the figure. Hunting around in the GSview menus, he finds 'EPS Clip' and 'Show Bounding Box' in the Options menu, and with either option checked, he sees that all is well.

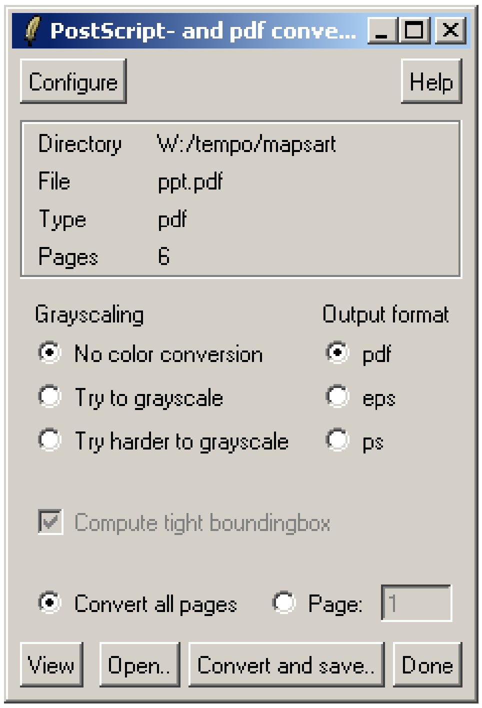
Case 2: Converting multiple slides from a PowerPoint presentation to pdf graphics. A.U. Thor, encouraged by his previous success, adds several new graphics to his PowerPoint file. Since epspdftk can handle multi-page documents, he prints the entire document to a PostScript file, which he loads in epspdftk.

He notices that the box with file info doesn't tell him the number of pages. For general PostScript files, there is no sure-fire quick way to get this information.

He checks 'Convert all pages' and sets Output format to pdf. After conversion, the info box does give him the number of pages.

He writes the first page to a pdf file with a tight boundingbox, reloads the complete pdf, then writes the second page to a pdf, and then wonders whether the command-line might not be more convenient.

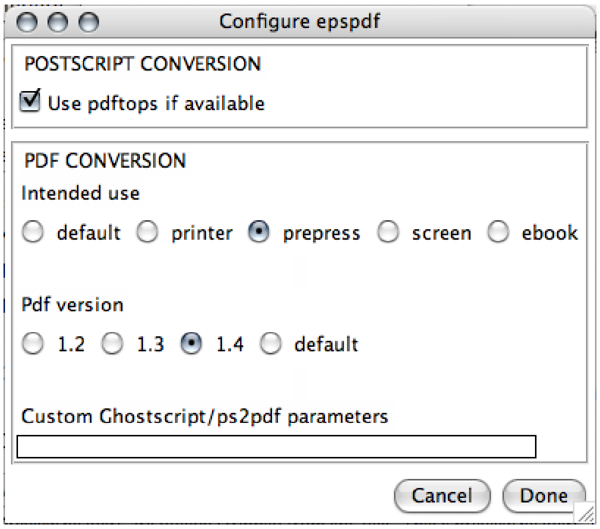
He reads the epspdf webpage and manual, browses



Directory W:/tempo/mapsart
File ppt.pdf
Type pdf
Pages 6

[✓ No color conversion]
[✓ pdf]
[✓ Compute tight boundingbox]
[✓ Convert all pages] Page: 1

Figure 1. Main window of epspdftk (MS Windows)



☑ Use pdftops if available

Intended use: ○ default ○ printer ● prepress ○ screen ○ ebook

Pdf version: ○ 1.2 ○ 1.3 ● 1.4 ○ default

Custom Ghostscript/ps2pdf parameters

Figure 2. Configuration screen (Mac OS X)

the epspdf directory and comes up with a batchfile epspdf.bat containing the following line:

```
"C:\Program Files\epspdf\bin\ruby.exe"
  "C:\Program Files\epspdf\app\epspdf.rb"
  %1 %2 %3 %4 %5 %6 %7 %8 %9
```

(everything on one line) and types

```
epspdf -b -p 3 ppt_slides.pdf figure3.pdf
```

Then he outdoes himself in cleverness and does the remainder with one command (everything on one line):

```
for %f in (4 5 6) do
  epspdf -b -p %f ppt_slides.pdf figure%f.pdf
```

Case 3: Creating cropped typeset samples. Mrs. TeX-HeX writes a paper for the MAPS about her adventures with LaTEX. She wants to display typeset examples with her own fonts and formatting, not with those of the MAPS. So she creates a LaTEX document containing one sample per page, and makes sure, with a preamble command `\pagestyle{empty}`, that each sample is the only content on its page. She compiles the document to a pdf and extracts the samples with a tight boundingbox, in the same way as in the previous example.

Case 4: Embedding fonts into an existing pdf. Ed Itor is collecting papers in pdf format for a conference proceedings. The printshop tells him that one of the submitted pdf files doesn't have all its fonts embedded, which is a no-no, even though the font is just Courier and Ed Itor doesn't quite understand the fuss.

Luckily, when converting PostScript to pdf, Ghostscript can embed standard PostScript fonts (Times etc.) even if they are missing in the PostScript file. Ed Itor goes to the Configure screen and verifies that 'Intended use' is set to to 'prepress'. With this setting, converting to PostScript and back to pdf does the trick.

Warning. It is quite possible that the original document was created with slightly different versions of the missing fonts than the ones included by Ghostscript. Usually this will cause no problems, but one might want to check the result anyway.

Some implementation details

The program is written in Ruby and Ruby/Tk, and uses Ghostscript and pdftops from the xpdf suite for the real work. The installation instructions in the download and on the web page explain how to obtain these programs.

The program consists of several modules. The main ones are a main library epspdf.rb which does double duty as command-line program, and a Ruby/Tk GUI main program. Conversions are organized as a series of single-step conversion methods and an any-to-any conversion method which strings together whatever single-step methods are required to get from A to B.

I have included all conversions and options into the program that easily fit into this scheme.

Configuration. Epspdf saves some conversion options between sessions. Under Windows it uses the registry, under Unix/Linux/Mac OS X a file .epspdfrc in the user's home directory. Besides some precooked options, advanced users can also enter custom options for Ghostscript (GUI and CLI) and for pdftops (CLI only).

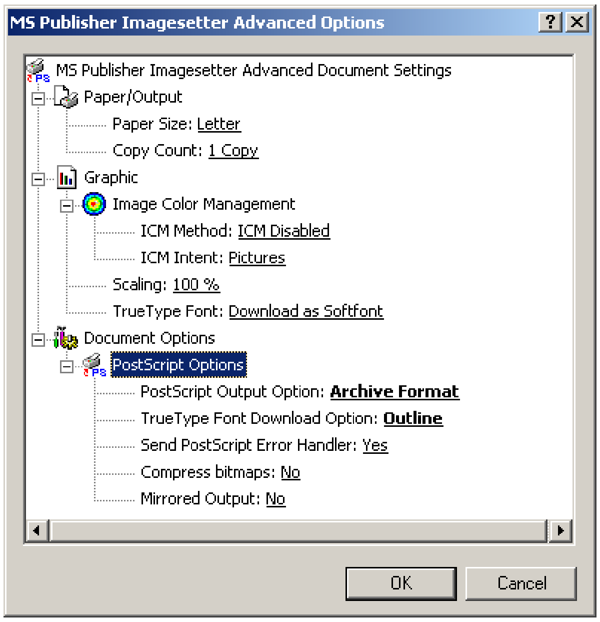
The Windows setup program. For Windows, epspdftk is available as a Windows setup program. This includes a Ruby/Tk subset, so there is no need for a full Ruby and Tcl/Tk install. This Ruby/Tk subset is adapted from one generated with RubyScript2Exe[3].

But Windows users can also manually install from a zipfile if they already have Ruby and Tcl/Tk.

Mac OS X. Under Mac OS X, epspdf mostly duplicates functionality from the Preview application. However, when faced with problem files it is nice to have an alternative converter. From the epspdf homepage you can download a double-clickable and dockable AppleScript applet for starting up epspdftk.

Appendix: exporting PostScript from Windows programs

Using a printerdriver. Often, the only way to get EPS or PostScript from a Windows program is by 'printing' to a PostScript file.


MS Publisher Imagesetter Advanced Options

MS Publisher Imagesetter Advanced Document Settings
- Paper/Output
  - Paper Size: Letter
  - Copy Count: 1 Copy
- Graphic
  - Image Color Management
    - ICM Method: ICM Disabled
    - ICM Intent: Pictures
  - Scaling: 100 %
  - TrueType Font: Download as Softfont
- Document Options
  - PostScript Options
    - PostScript Output Option: Archive Format
    - TrueType Font Download Option: Outline
    - Send PostScript Error Handler: Yes
    - Compress bitmaps: No
    - Mirrored Output: No

Figure 3. MS Publisher Imagesetter – printer settings

For this, you need to have a PostScript printer driver installed. You can pick 'FILE' for printer port. A suitable driver which comes with Windows is Generic / MS Publisher Imagesetter.

Pay attention to printer settings: in the Print dialog, click 'Properties', then 'Advanced' (on either tab). In the 'Advanced Document Settings' tree, navigate to first 'Document Options' then 'PostScript Options' (figure 3).

For 'PostScript Output Option' the default setting is 'Optimize for speed'. Change that to 'Optimize for Portability' or 'Archive Format', or, for single pages only, 'Encapsulated PostScript'. These non-default options presumably produce cleaner PostScript code, without printer-specific hacks. Experiment with this and other options if you run into problems (e.g. bad-looking screen output, or part of a graphic getting cut off, or conversion to bitmap).

What works best may depend on your Windows version: under Windows 2000, Archive worked best for me, but Taco Hoekwater warns me that this option was unusable in older Windows versions.

Another setting here to pay attention to is 'TrueType Font Downloading Option'. Pick 'Outline', not 'Automatic' or 'Bitmap'.

Using a program. Other possibilities for generating PostScript or pdf are the TpX and wmf2eps programs, which both can read Windows wmf- and emf files and also have options to write clipboard contents to an emf file. Wmf2eps uses its own virtual PostScript printer driver in the background. For faithful conversion, pick wmf2eps; for subsequent editing, choose TpX. Both programs are availbable from CTAN[4].

URLs

1. `http://tex.aanhet.net/epspdf/`
2. `http://www.foolabs.com/xpdf/`
3. `http://rubyforge.org/projects/rubyscript2exe/`
4. `http://www.tug.org/ctan.html`

Siep Kroonenberg
siepo at cybercomm dot nl