Converting a PDF to PNG

I'm trying to convert a PDF to a PNG image (at least the cover of one). I'm successfully extracting the first page of the PDF with pdftk. I'm using imagemagick to do the conversion:

```
convert cover.pdf cover.png
```

This works, but unfortunately the cover.png comes through incorrectly rendered (some of the alpha object in the PDF aren't rendered properly). I know ImageMagick uses GhostScript to do the conversion and if I do it directly with gs I can get the desired results, but I'd rather use the convert library as it has other tools I'd like to leverage.

This command in GhostScript accomplishes the desired image:

```
gs -sDEVICE=pngalpha -sOutputFile=cover.png -r144 cover.pdf
```

I'm wondering is there any way to pass arguments through convert to GhostScript or am I stuck with calling GhostScript directly?

image pdf png imagemagick ghostscript

edited Dec 9 '16 at 18:30
comecme
2,758 ●7 ●26 ●52

asked Mar 17 '09 at 8:18
Adam
713 ●1 ●9 ●13

1 Why is calling GhostScript directly a problem? – kquinn Mar 17 '09 at 8:37

It really isn't that big of a deal. I'd like to run some other params through convert at the same time and it'd be nice if I could keep it all in one command. Keeps my code cleaner and more consistent. It also means one less temporary file. – Adam Mar 17 '09 at 9:18

See also PDFBox: Problem with converting pdf page into image and Use Apache PDFBox convert PDF to image. – Petr Pudlák Jul 5 '13 at 16:15

What's the difference between how you call gs and how ImageMagick calls it? Might be worth reporting something upstream to ImageMagick (note to followers, updating ghostscript can help as well...) – rogerdpack Mar 3 at 19:09

8 Answers

You can use one commandline with two commands (`gs` , `convert`) connected through a pipe, if the first command can write its output to stdout, and if the second one can read its input from stdin.

1. Luckily, gs can write to stdout (`... -o %stdout ...`).
2. Luckily, convert can read from stdin (`convert -background transparent - output.png`).

Problem solved:

- GS used for alpha channel handling a special image,
- convert used for creating transparent background,

- pipe used to avoid writing out a temp file on disk.

Complete solution:

```
gs -sDEVICE=pngalpha       \
   -o %stdout              \
   -r144 cover.pdf         \
    |                      \
convert                    \
   -background transparent \
   -                       \
    cover.png
```

Update

If you want to have a separate PNG per PDF page, you can use the `%d` syntax:

```
gs -sDEVICE=pngalpha -o file-%03d.png -r144 cover.pdf
```

This will create PNG files named `page-000.png`, `page-001.png`, ... (Note that the `%d`-counting is zero-based -- `file-000.png` corresponds to page 1 of the PDF, `001` to page 2...

Or, if you want to keep your transparent background, for a 100-page PDF, do

```
for i in {1..100}; do        \
                             \
  gs -sDEVICE=pngalpha       \
     -dFirstPage="${i}"      \
     -dLastPage="${i}"       \
     -o %stdout              \
     -r144 input.pdf         \
      |                      \
  convert                    \
     -background transparent \
     -                       \
      page-${i}.png ;        \
                             \
done
```

edited Jun 2 at 2:06
Miro
136 ● 1 ● 9

answered Jul 31 '10 at 20:14
Kurt Pfeifle
53.9k ● 14 ● 147 ● 234

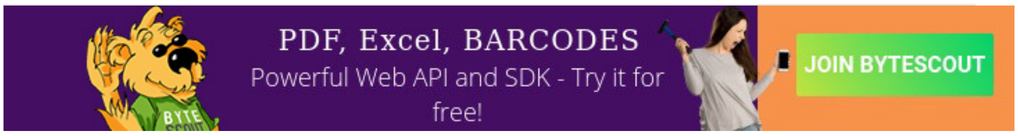
6 This only works for me if I add `-dBATCH -dNOPAUSE -dQUIET` to the gs options. – ford Nov 1 '13 at 13:51

@ford: That means you have an old version of Ghostscript. Recent versions can do `-o output.file` and this automatically and silently also sets `-dBATCH -dNOPAUSE -dQUIET` at the same time. – Kurt Pfeifle Nov 16 '14 at 19:58

@ford: However, I had a serious typo elsewhere in the above answer. I wonder why it got 22 upvotes despite of that :-) – Kurt Pfeifle Nov 16 '14 at 20:00

1 @Tarass: see my updated answer – Kurt Pfeifle Jun 21 '15 at 14:06

1 @Tarass: Did you see the updated Update? – Kurt Pfeifle Jun 21 '15 at 19:48



Out of all the available alternatives I found Inkscape to produce the most accurate results when converting PDFs to PNG. Especially when the source file had transparent layers, Inkscape succeeded where Imagemagick and other tools failed.

This is the command I use:

```
inkscape "$pdf" -z --export-dpi=600 --export-area-drawing --export-png="$pngfile"
```

And here it is implemented in a script:

```
#!/bin/bash

while [ $# -gt 0 ]; do

pdf=$1
echo "Converting "$pdf" ..."
pngfile=`echo "$pdf" | sed 's/\.\w*$/.png/'`
inkscape "$pdf" -z --export-dpi=600 --export-area-drawing --export-png="$pngfile"
echo "Converted to "$pngfile""
shift

done

echo "All jobs done. Exiting."
```

answered Mar 18 '13 at 18:59
Glutanimate
731 ● 8 ● 11

To convert pdf to image files use following commands:

For PNG `gs -sDEVICE=png16m -dTextAlphaBits=4 -r300 -o a.png a.pdf`

For JPG `gs -sDEVICE=jpeg -dTextAlphaBits=4 -r300 -o a.jpg a.pdf`

If you have multiple pages add to name **%03d** `gs -o a%03d.jpg a.pdf`

What each option means:

- sDEVICE={jpeg,pngalpha,png16m...} - filetype
- -o - output file (%stdout to stdout)
- -dTextAlphaBits=4 - font antialiasing.
- -r300 - 300 dpi

edited Nov 1 '16 at 4:16
Michael Mügge
2,650 ● 1 ● 17 ● 22

answered Nov 4 '15 at 17:52
yanpas
745 ● 1 ● 7 ● 15

Here is a german discussion about a problem like this for SVG files where it is solved by using

```
convert -background transparent
```

Perhaps this works for you, too.

answered Mar 17 '09 at 8:34
schnaader
37.1k ● 5 ● 75 ● 107

Sadly no that doesn't solve my problem. It's actually an image in the PDF that has an alpha channel that sits on top of everything. – Adam Mar 17 '09 at 8:58

I'll add my solution, even thought his thread is old. Maybe this will help someone anyway.

First, I need to generate the PDF. I use XeLaTeX for that:

```
xelatex test.tex
```

Now, ImageMagick and GraphicMagic both parse parameters from left to right, so the leftmost parameter, will be executed first. I ended up using this sequence for optimal processing:

```
gm convert -trim -transparent white -background transparent -density 1200x1200 -
resize 25% test.pdf test.png
```

It gives nice graphics on transparent background, trimmed to what is actually on the page. The `-density` and `-resize` parameters, give a better granularity, and increase overall resolution.

I suggest checking if the density can be decreased for you. It'll cut down converting time.

answered Jul 11 '12 at 12:04
polemon
2,702 ● 2 ● 21 ● 41

For a PDF that ImageMagick was giving inaccurate colors I found that GraphicsMagick did a better job:

```
$ gm convert -quality 100 -thumbnail x300 -flatten journal.pdf\[0\] cover.jpg
```

answered Sep 2 '15 at 7:00
aorth
113 ● 1 ● 5

My solution is much simpler and more direct. At least it works that way on my PC (with the following specs):

```
me@home: my.folder$ uname -a
Linux home 3.2.0-54-generic-pae #82-Ubuntu SMP Tue Sep 10 20:29:22 UTC 2013 i686
i686 i386 GNU/Linux
```

with

```
me@home: my.folder$ convert --version
Version: ImageMagick 6.6.9-7 2012-08-17 Q16 http://www.imagemagick.org
Copyright: Copyright (C) 1999-2011 ImageMagick Studio LLC
Features: OpenMP
```

So, here's what I run on my `file.pdf`:

```
me@home: my.folder$ convert -density 300 -quality 100 file.pdf file.png
```

answered Nov 16 '13 at 13:41
polarise
864 ● 7 ● 14

Yeah this is what the OP tried initially but couldn't get something err other to work underneath when ImageMagick calls through to ghostscript...but if it works go for it :) – rogerdpack Mar 3 at 19:12

Couldn't get the accepted answer to work. Then found out that actually the solution is much simpler anyway as Ghostscript not just natively supports PNG but even multiple different "encodings":

- `png256`
- `png16`
- `pnggray`
- `pngmono`
- ...

The shell command that works for me is:

```
gs -dNOPAUSE -q -sDEVICE=pnggray -r500 -dBATCH -dFirstPage=2 -dLastPage=2 -
sOutputFile=test.png test.pdf
```

It will save page 2 of test.pdf to test.png using the `pnggray` encoding and 500 DPI.

answered Mar 17 '15 at 19:50
Raffael
10.8k ● 9 ● 54 ● 119