FAQ

Welcome to the Ghostscript/GhostPDL FAQ (Frequently Asked Questions)

- Why is this FAQ so empty?

```
Because we've only just started it.
```

- What is Ghostscript?

```
Ghostscript is a high quality, high performance Postscript and PDF
interpreter and rendering engine.
```

- What is GhostPDL?

```
GhostPDL is a suite of interpreters for PCL, PXL and XPS implemented
using the Ghostscript graphics library. GhostPDL includes Ghostscript.
```

- Where can I download Ghostscript/GhostPDL?

```
You can get it here
There are also the following mirrors: sourceforge.net and code.google.com
```

- Can I install Ghostscript on Windows unattended?

```
Yes, give gsxxxw32/64.exe the /S (for silent) option.
In addition, you can add the /NCRC option (that will suppress
the CRC check, which, unfortunately, pops up a dialogue to show
progress). And /D=<install dir> to change the target directory
for the install.
```

- Why doesn't Ghostscript support production of PDF/A-1a ?

```
This is basically not possible when starting from a
source which is not itself PDF/A-1a compliant.

The differences between level b and level a are :

1) All fonts must be encoded using a standard encoding or they
   must include a ToUnicode CMap.

   There exist numerous PostScript (and a number of PDF) files for
   which it is not possible to construct a ToUnicode CMap, and which
   are not encoded in one of the specified ways. These files therefore
   cannot be converted into PDF/A1-a files. Note that where possible
   pdfwrite *does* emit ToUnicode information, so for those files where
   this is possible it is already conformant in this regard.

2) The file must contain 'tagged' textual data in order to recover the text
   information easily. This is simply impossible for a general purpose PDF
   converter. In the general case there is no way to infer which portions of
   text on a page are (for example) running heads, page numbers, footnote
   rules etc. Nor is it possible to reliably generate, for example, the
   reading order of the text.

   The ISO specification clearly says "PDF/A-1 writers should not add
   structural or semantic information that is not explicitly or implicitly
   present in the source material solely for the purpose of achieving
   conformance." and also "It is inadvisable for writers to generate
   structural or semantic information using automated processes without
   appropriate verification."

   There are a number of other requirements which are also not possible for
   a conversion application to deal with, for example:

    "6.8.6 Non-textual annotations. For annotation types that do not display
     text, the Contents key of an annotation dictionary should be specified with
     an alternative description of the annotation's contents in human-readable
     form."

   Its impossible for pdfwrite to create a textual representation for an
   arbitrary annotation.

   Also:

    "6.8.7 Replacement text. All textual structure elements that are represented
```

```
    in a non-standard manner, e.g., custom characters or inline graphics,
    should supply replacement text using the ActualText entry in the structure
    element dictionary, as described in PDF Reference 9.8.3."

  It is quite common to have text represented as an image, or drawn as a
  series of vectors, there is no way for pdfwrite to reconstruct an
  'ActualText' entry for these. In fact we probably can't even identify that
  they are 'textual'.

  We do note that Adobe Distiller 9 does offer the option of producing
  PDF/A1-a (2001) output. While it is possible that this simply aborts when
  ToUnicode information cannot be added, its not clear to me what it does
  about the limitations imposed by section 6.8.  I would guess that it simply
  tags all text as 'body', ignores the requirement for ActualText, and possibly
  aborts if annotations are non-textual. Its also possible that the 2001 revision
  does not contain section 6.8. I only have a copy of the 2005 revision. which
  was the spec adopted by the ISO and am unable to find any earlier revision.
  I imagine that anyone asking for PDF/A-1a means the 2005 ISO revision anyway.

  The main additional aim of PDF/A-1a seems to be for text extraction and
  accessibility usage (I imagine this is why government agencies are mandating
  it). So it is important that documents can have textual representation of all
  elements (for text to speech readers for example), that the reading order
  is correct and that the language can be identified. We have no way of
  adding any of this information if it is not already present in the source.

  Although we could produce a document which would pass an automated
  conformance checker, we could not guarantee that the file would be
  'correct' as regards the intention of the specification.

  For these reasons, at present we have no plans to implement PDF/A1-a in
  pdfwrite.
```

- How do I specify custom substitutions on Windows?

```
Since recent releases of Ghostscript compiles the initialisation files
(include Fontmap.GS) into the executable, you will need to download a
copy of the default Fontmap.GS file. The easiest way to do so is to
download it from here:
Fontmap.GS

Navigate to the directory into which you downloaded Fontmap.GS and
copy the file into a new directory under your ghostscript install
directory (for example, c:\Program Files\gs\gs-<version>\init). Open
your newly copied Fontmap.GS in a plain text editor (Windows Notepad,
for example - do not use Wordpad, or Word or other word processor
application). You can then add font name to file name mappings, for
example, you might add:
  /CourierNew         (C:/Windows/Fonts/cour.ttf) ;

So that Ghostscript will substitute the Windows TrueType Courier font
when a job requests a font names CourierNew.

To ensure that Ghostscript uses the new Fontmap.GS, when you call
Ghostscript, add -I"<gsInstallDir>\<directory containing fontmap>" to
your switches. So in the example given above, you would add:
  -I"c:/Program Files/gs/gs-<version>/init"

Ghostscript should now use the the font name to file mapping supplied.
```

- Is it possible to cross compile GhostPCL/GhostXPS

```
Yes, it is. The steps are as follows:
  1) Patch the configure script so the call to the libtiff configure has the required options for your target
  2) Pass configure the --enable-big-endian or --enable-little-endian as appropriate for your target
  3) Pass configure the --disable-sse2 as appropriate for your target
  4) generate arch.h manually and modify gs/base/lib.mak to copy your arch.h rather than generate a new one
  5) Set the CCAUX variable to the native host compiler
  6) Set the CC variable to your cross compiler
```

- Where can I see more Frequently Asked Questions about Ghostscript and Postscript in general.

  Wikibooks PostScript FAQ

Home Contents

Search: