XeLaTeX and the PDF archivable format

Claudio Beccari

Abstract

At this time, XeLaTeX produces a final PDF output file but it gets this output by means of the transformation of a XDV (extended DVI) intermediate file. This excludes most of the possibilities offered by pdfLaTeX that, since version 1.40.9 and with the help of an extension file `pdfx.sty`, can directly produce a PDF/A-1b compliant output. This paper explains how to overcome this by resorting to the ubiquitous Ghostscript program.

1 Introduction

Several papers have been already published in TeX-related journals, ArsTeXnica and *TUGboat* included, about producing PDF/A-compliant archivable files. Almost all of these papers focused on the various *caveats* that are necessary to observe in order to avoid the many pitfalls of this format. Some papers also discussed the fact that the color profiles are not so clearly defined, so sometimes a non-compliant file is obtained just because an unsuitable color profile file has been employed. Some papers pointed out that sometimes the Preflight program, the most authoritative one included in the Adobe Acrobat Pro suite, fails to recognize file compliance with the ISO standard labelled PDF/A-1a or PDF/A-1b. But to the best of my knowledge, no paper has dealt with producing a PDF/A-compliant file from a source intended to be composed with XeLaTeX.

Let us recall some pieces of information. The PDF/A ISO standard was devised in 2005, and regulated with the ISO-19005-1:2005 document. This standardizes two sub-formats, labelled 1a and 1b, with the latter being less stringent than the former; it requires that the level of the PDF language used in the PDF file is level 4; it requires the fonts to be outlines and that they be subset-embedded in the document file; it requires that the color profiles are clearly defined, and it requires the presence of certain metadata, in a non-encrypted form, so that library searches can be performed. The purpose is to have files that fulfill specific limitations, but that will be readable from now on for an unlimited length of time, in spite of the fact that in, say, fifty years the fonts and the programs available today may not exist any more. The former sub-format, 1a, is more stringent in the sense that the fonts must be completely embedded and also the document structure must be included in the file.

As far as I am aware, archivable files in the 1b subformat are generally sufficient for the purpose of long-term reproduction on screen of the archived documents, exactly as the authors intended them to be. Therefore I will concentrate on this "simpler" format; besides, as a practical matter, I did not find any means for producing the 1a format except the Preflight program of Adobe Acrobat Pro, to which I have no direct access.

Finally it must be noticed that a new standard has been issued in 2011, ISO 19005-2:2011, that slightly extends the previous PDF/A standard; with this new standard, PDF language level 6 may be used and JPEG2000 images may be included in the archivable documents. Although these are important enhancements in certain areas, I will not deal with them and just stick with the previous standard, for no other reason than that the Ghostscript program, which is needed for the task, is not yet capable of satisfying the new standard. I hope that in a short time Ghostscript will be updated and its documentation will show the small modifications that need to be introduced into the necessary scripts.

2 PDF/A requirements

2.1 The metadata

Any PDF/A compliant file must contain some metadata that describe some features of the document, from the color profile and the document title and author, to the keywords that ease library searches of the archived document. Some metadata are compulsory, some are optional.

As for the compulsory metadata, I show below how to prepare a suitable auxiliary file that contains all the necessary information in the PostScript language; Ghostscript will translate such information into the XML language and will insert this XML code into the output file.

In some sense this is the simplest part of the whole procedure; the problem consists of knowing what information to supply and in which form.

2.2 The color profile

One of the most mysterious pieces of information is the one that describes the color profile; Luigi Scarso already wrote a paper [9] where he discusses this problem in connection with the typesetting program ConTeXt MkIV. But the main problem is not the particular typesetting program, but rather the very concept of *color profile*. I won't go any deeper into this question, but rather redirect the reader to [9], where the question is thoroughly discussed. Here, I

Editor's note: First published in ArsTeXnica #11, April 2011. Reprinted with permission, in slightly different form.

remark that I have obtained satisfactory results by downloading the color profile file `ECI-RGB.V1.0.icc`, freely downloaded from the `www.eci.org`. This file may be saved anywhere that Ghostscript can find it, but I suggest saving the file in a system-wide folder and to specify an absolute path when dealing with this file.

This color profile is generic, and yields satisfactory results in most circumstances; it deals only with the RGB (red, green, blue) additive color model (used by, for instance, TV and computer screens) and the images inserted into the document file should be consistent with this color model. Therefore, no image in the CMYK (cyan, magenta, yellow, black) subtractive color model should be used in a PDF/A-compliant file when the color profile refers to the RGB color model.

2.3 Hyperlinks

PDF/A-compliant files may use internal hyperlinks in order to ease the document navigation. "Internal" links means that link targets are internal to the document, so the reader may use the bookmark pane of a PDF reader to jump from one point to another in the document. This possibility is particularly useful in a reference document.

External links are prohibited in PDF/A; external links refer to other documents on the same computer, or to targets in the Internet. It is evident that a long term archiving process cannot have links between objects that may exist today, but most likely will not exist in fifty years. Therefore, when archiving is a requirement, any document should be self contained; if one needs to refer to another document, include either a complete reference in the document bibliography, or the referenced document in its entirety. The choice depends on the author, but we must keep in mind the requirement that the archived document *must* be self contained.

This is not a trivial constraint; after all, it's our daily experience, for anyone using the Internet, that many Internet addresses valid yesterday are not valid today, not to mention fifty years from now!

In order to be sure to have the hyperlinks behave as they should with PDF/A-compliant documents, it is sufficient to specify the `pdfa` option either to the class itself, or to the `hyperref` package directly, or to the `\hypersetup` command with which the package may be customized.

2.4 Fonts

The default fonts of any TEX distribution are pretty good for most purposes, but they fail in some other instances. I already wrote a short paper on this subject [2] in order to find a patch to "heal" the `cmsy*` fonts that use some zero-width glyphs. This happened with the math family three fonts, the one that contains the math symbols, because some symbols, such as $\mapsto$ and all the negated relation operators $\neq$, $\notin$, etc., resorted to the superposition of a zero-width glyph over, or beside, another symbol. Zero-width glyphs are not PDF/A compliant.

With XELATEX the font choice is much wider, although where math typesetting is concerned, at this time there are only a few suitable OpenType fonts; but, as Unicode-encoded fonts, they are (or should be) safe under the point of view of compliance with the PDF/A requirements. I confess that I did not test for this conformity the recently available OpenType Latin Modern Math fonts, but I tested the XITS math fonts (with the widest choice of math glyphs) and I did not find any glitch.

On the other hand, I found out something that either I neglected in the past or that more modern checking programs can spot: some problems with the use of accents in the "normal" Type 1 Computer Modern `OT1`-encoded fonts.

Evidently if you use XELATEX you have no problem in avoiding the traditional Computer Modern fonts; if you like their shape, you'd rather use the OpenType CM-Unicode fonts (which contain the full accented set of Latin letters, Cyrillic letters, and Greek letters, suitable for the Greek polytonic orthography), and there would not be any problem. But the problem might show up when you include in your document PDF files or pages extracted from other documents, where the latter were typeset with the default CM fonts.

I may have overlooked this problem in the past, because I never use `OT1`-encoded fonts for typesetting my documents; moreover, in principle nobody should use the `OT1`-encoded fonts if the document they typeset contains even a single accented word. The poor performance of the `OT1`-encoded fonts depends on the fact that accents are superimposed on any letter that has to be accented by an overlay mechanism that is visually acceptable when a document is printed, but in effect is a poor patch that does not follow the best practice used by the OpenType fonts and the `T1` encoded "normal" TEX fonts: OpenType fonts use accented letters and `T1` encoded fonts translate the accenting process into a glyph substitution so that no overlay process is involved.

Therefore it is most important to check the type and quality of the fonts embedded into a PDF file to be included, be it a technical diagram or a stretch of text. For this purpose I find the free program Adobe Reader X (any recent version will do the

task) extremely useful, where pressing ctrl+D (or cmd+D on Mac computers) opens a dialog pane where the user can select the "Fonts" tag and get the full list of the fonts contained in the document. In particular, the user can require that no Type 3 (bitmapped) fonts are used and, if any CM font is used, the user may examine the document to find out if any accented letters have been used. In the latter case the user should process the document to be included by following the procedure shown in the next section.

2.5 Included documents and files

If the documents are in PDF format, the Adobe Reader procedure above may be used also for checking other document characteristics. This or other programs should be used also for controlling the color model of the documents or images to be included.

The user must distinguish between bitmap and vector images. The former may contain anything, from photographs to text and line art; they must be controlled only to determine the color model. Should it be a "wrong" model, almost any image processing program can be used to open the file and save it again with a different color model. Remember the only color model usable for PDF/A-compliant images is RGB, though the "gray" model is also admissible, since the RGB model can render gray nuances very well. But while bitmap format may be acceptable for photographs with a pixel density of at least 300 dpi, in general it is not valid for line art and textual files. The compression method used by the JPEG format (`.jpg`) is not lossless, therefore the reproduced image may exhibit unwanted artifacts, often very visible if the image contains periodic patterns. The Portable Network Graphics format (`.png`) uses a lossless compression method, and in general is more suited for line art. Vector graphics, in any case, are the best suited for line art, but sometimes it's not possible to have every line art picture available in such a format.

But stretches of text and vector graphics lose all quality in a terrible way if they are converted to bitmapped form; therefore the user should pay much attention to what s/he does with files to be included that do not comply with the PDF/A standard.

Concerning `.eps` vector files, it's easy to convert them to `.pdf`, but it's necessary to pay attention that every possible font glyph is embedded into the transformed file.

One way to correct the font problems of a PDF file is to open it with the free program `inkscape` and save it back to PDF format. In this process the vector fonts are converted into vector drawings that reproduce the same glyphs, but do not exploit or exhibit any font property. In this way, for example, it is possible to draw the glyphs of the fonts that were not embedded into the file, and it is possible to redraw the poor accent overlay of the CM fonts. I confess I myself have never tried this procedure, but it was suggested by Hàn Thế Thành [4].

3 The metadata auxiliary file

We are almost ready to produce a PDF/A-compliant file from a file typeset with XeLaTeX. I assume that XeLaTeX has been run the sufficient number of times so as to be sure that the final PDF document does not require any further adjustment. Of course we can repeat the transformation, but it's better to wait until the document is substantially stable. We are going to use Ghostscript; it must be at version 8.60 or higher; the more recent the better. At writing time I am using version 9.02.

As already mentioned, we need an auxiliary file to specify the metadata and to set up some special PostScript commands necessary for this task. The Ghostscript distribution contains a file named `PDFA_def.ps` contained in the sub-tree `.../ghostscript/`⟨*version*⟩`/lib/`; where this sub-tree is grafted into your general system tree depends on your platform and your operating system, but you can execute a search command and find out where all this resides if need be.

Copy the above-mentioned file `PDFA_def.ps` to the folder where you saved your document master file. Suppose this master file is named `mydoc.tex`; copy the above `.ps` file to `mydoc-def.ps` (notice I changed the underscore into a normal "hyphen sign" or "short dash").

Now open this file with an ASCII editor; since it is a `.ps` file you might right-click the file name, but you have to choose the editor name yourself, since you cannot use the default application for `.ps` files. On a Windows platform you might use `Wordpad`; on GNU/Linux, you might use `vim`, `emacs`, `gedit`; on a Mac, you might use `Textedit.app` or `TextWrangler.app` or `TeXworks.app`, but avoid using `TeXShop.app`, since TeXShop is capable of "distilling" a `.dvi` or `.ps` file into PDF format, which generally is a very useful feature — but not in this case!

The newly created `mydoc-def.ps` contains a few lines commented with a `% Customize` comment; you should edit these lines in the way that is best suited to introduce the necessary metadata information. To insert the metadata suitable for this article, I would use the file as shown on page 19.

As you can see, there are three sets of data that can be customized:

1. The `/ICCProfile`; I changed the default to the

The auxiliary .ps file

```
%!
% $Id: PDFA_def.ps 8284 2007-10-10 17:40:38Z giles $
% This is a sample prefix file for creating a PDF/A document.
% Feel free to modify entries marked with "Customize".

% This assumes an ICC profile to reside in the file (ISO Coated sb.icc),
% unless the user modifies the corresponding line below.

systemdict /ProcessColorModel known {
...
} if

% Define entries to the document Info dictionary:
/ICCProfile (/Users/claudio/icc/ECI-RGB.V1.0.icc) def % Customize

[ /Title (XeLaTeX and the PDF archivable format) % Customize.
/Author (Claudio Beccari) % Customize.
/Subject (How to produce a PDF/A compliant document typeset with XeLaTeX) % Customize.
  /DOCINFO pdfmark

% Define an ICC profile :
[/_objdef {icc_PDFA} /type /stream /OBJ pdfmark
[{icc_PDFA} <</N systemdict /ProcessColorModel get
          /DeviceGray eq {1} {4} ifelse >> /PUT pdfmark
[{icc_PDFA} ICCProfile (r) file /PUT pdfmark

% Define the output intent dictionary :
[/_objdef {OutputIntent_PDFA} /type /dict /OBJ pdfmark
[{OutputIntent_PDFA} <<
  /Type /OutputIntent             % Must be so (the standard requires).
  /S /GTS_PDFA1                   % Must be so (the standard requires).
  /DestOutputProfile {icc_PDFA}            % Must be so (see above).
  /OutputConditionIdentifier (CGATS TR001)      % Customize
>> /PUT pdfmark
[{Catalog} <</OutputIntents [ {OutputIntent_PDFA} ]>> /PUT pdfmark
```

The shell script

```
#!/bin/bash
file1=$1.pdf
file2=$1-a.pdf
file3=$1-def.ps
# WHAT FOLLOWS MUST BE WRITTEN ON JUST ONE LINE:
gs -dPDFA -dNOOUTERSAVE -dUseCIEColor -dCompatibilityLevel=1.4 -sDEVICE=pdfwrite
   -sProcessColorModel=DeviceCMYK -sPDFACompatibilityPolicy=1 -o "$file2" "./$file3" "$file1"
```

above-mentioned file `ECI-RGB.V1.0.icc`, giving the full path from the root of my disk to the file; probably I could have abbreviated my home folder with `~`, but the full path is more easily changeable on those Windows platforms that do not have the notion of "home".

2. The document `/Title`, `/Author` and `/Subject`; other similar metadata may be added in a similar way, but remember: these metadata have nothing to do with those you can specify in the input files to be processed with `pdfLaTeX` by means of specific `\pdf...` commands; some of those commands may have a meaning also for XeLaTeX, but their contents do not migrate to the proper PDF/A file section where metadata should be.
3. The `/OutputConditionIdentifier`; I did not modify it at all.

Now the fact that we named this auxiliary file with relation to the main document comes in handy,

because it is clear that every document needs its own auxiliary file. We thus avoid the possibility of fiddling with a myriad of `PDFA_def.ps` files, all with the same name, even if they are in different folders, and with Ghostscript looking for it starting with its own sub-tree; otherwise, you may get very frustrated trying to figure out why Ghostscript does not fetch your newly-created `.ps` file.

4 The script

Ghostscript requires a long series of specifications for doing its job; shell scripts (for Unix-like systems) or bat files (for Windows) are good for specifying the whole command string without errors and without the risk of forgetting some terms.

The shell script on page 19 may be copied for use; the only attention that must be paid is to eliminate the end-of-lines in the last long command; it must consist of just one line. The shell script may be changed into a bat file by using the Windows name for the Ghostscript executable (`gswin32c` in place of `gs`), using `%1` in place of `$1`, and changing the comment character `#` to the string `REM`. The first three `file` assignments are not really necessary; one might avoid them and use only one symbolic parameter, but they might be used for testing the presence of the files that are required and to issue the necessary messages in case they were missing. The bash file might be saved with the name `pdf2pdfa` (or `pdf2pdfa.bat`) without forgetting to give it the proper mode bits, for example, `chmod 755 pdf2pdfa`.

Now the user may simply issue on the terminal the command:

```
pdf2pdfa mydoc
```

and Ghostscript will do the work.

Of course the last step is to check PDF/A compliance with a reliable program, such as Preflight. Please do not trust the information issued by Adobe Reader X when it opens a PDF file that pretends to be PDF/A-compliant. The information is a wish, in the sense that the statement might be true, but the document has not been really tested in every remote corner of its compliance. So the real message should not be: "This file is PDF/A compliant", but: "This file might be PDF/A compliant".

5 Conclusion

I have converted a number of PDF documents produced with XeLaTeX using the procedure described in the previous sections. Of course I have observed all the caveats I listed above; in fact, they are the result of my failing efforts. I had to find out at every failure the cause, but eventually I could put together a reasonable list of caveats.

There is another thing that might be done: to write an extension file to be read by the main document file, that accepts LaTeX style commands in order to specify the metadata and write the auxiliary file without any specific intervention by the user; this extension could also run Ghostscript by means of a suitable "shell escape" command at the end of the processing by XeLaTeX, as the last task of the `\end{document}` statement. Apparently there already exist the necessary hooks, but I must leave this further task for another moment.

References

[1] Adobe. XMP toolkit SDK. `http://www.adobe.com/devnet/xmp/sdk/eula.html`, 2010.

[2] Claudio Beccari. Some PDF/A tricks. *The PracTeX Journal*, 2010. `http://tug.org/pracjourn/2010-1/beccari`.

[3] Olaf Drümmer, Alexandra Oettler, and Dietrich von Seggern. *PDF/A in a Nutshell: Long Term Archiving with PDF*. 2008. `http://www.pdfa.org/download/pdfa-in-a-nutshell`.

[4] Hàn Thế Thành. Generating PDF/A compliant PDFs from pdfTeX. `http://support.river-valley.com/wiki/index.php?title=Generating_PDF/A_compliant_PDFs_from_pdftex`, 2008.

[5] Donald E. Knuth. *The TeXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, Massachusetts, 1990.

[6] Emanuela Martini. Lo standard PDF/A: Sperimentazione di software per la verifica di conformità allo standard ISO 19005:2005, 2007.

[7] PDF/A. Technical notes. `http://www.pdfa.org/publications`, 2005–2010.

[8] CV Radhakrishnan and Hàn Thế Thành. Generation of PDF/X-1a and PDF/A-1b compliant PDF's with pdfTeX — `pdfx.sty`. `http://mirror.ctan.org/macros/latex/contrib/pdfx`, 2009.

[9] Luigi Scarso. Introduction to colours in ConTeXt MkIV. *TUGboat*, 31(3):203–207, 2010.

◇ Claudio Beccari
Villarbasse (TO), Italy
`claudio dot beccari (at) gmail dot com`