XeLaTeX and the PDF archivable format

Claudio Beccari

Sommario

Up to now XeLaTeX produces a final PDF output file but it gets this output by means of the transforamtion of a XDV (extended DVI) intermediate file. This excludes most of the possibilities offered by pdfLaTeX that, at least since version 1.40.9, and with the help of an extension file pdfx.sty, can directly producce a PDF/A-1b compliant output. This paper explains how to get through this bottle neck by resorting to the ubiquitous ghostscript program.

Sommario

Fino ad oggi XeLaTeX produce un file finale in formato PDF ma passando attraverso la trasformazione di un file intermedio in formato XDV (extended DVI). Questa trasformazione esclude che ci si possa servire delle funzionalità del programma pdfLaTeX che, almeno dalla versione 1.40.9 in poi, con l'aiuto di un file di estensione pdfx.sty, riesce a produrre direttamente un file di uscita conforme allo standard PDF/A-1b. Questo articolo spiega come superare questo intoppo ricorrendo al programma ghostscript presente su ogni piattaforma.

1 Introduction

Several papers have been already published on several TeX related magazines, ArsTeXnica included, about producing PDF/A compliant archivable files. Almost all these papers focused the reader's attention on the various *caveats* that is necessary to observe in order to avoid the many pitfalls of this format. Some papers focused the attention also on the fact that the color profiles are not so clearly defined so that sometimes a non compliant file is obtained just because an unsuitable color profile file has been employed. Some papers focused the attention also on the fact that sometimes the Preflight program, the most authoritative one included in the Adobe Acrobat Pro suite, fails to recognize the file compliance with the ISO standard labelled PDF/A-1a or PDF/A-1b. But to my best knowledge, no paper deals 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 has been devised in the year 2005 and regulated with the ISO-19005-1:2005 regulation. This regulation standardizes two subformats, labelled 1a and 1b; the latter is 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 vector ones and that they are subset-embedded in the document file; it requires that the color profiles are clearly defined, and it requires the presence of a certain number of *metadata* to be included into the file 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, the 1a one, 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 of, archivable files in the 1b sub format are generally sufficient for the purpose of the 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, also because I did not find any means for producing the 1a format, except the Preflight program of Adobe Acrobat Pro, which I have no direct access to.

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

2 The 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 the library search of the archived document. Some *metadata* are compulsory, some are optional.

For what concerns 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 a certain sense this is the simplest part of the whole procedure; the problem consists in knowing what information must be supplied and in which form.

2.2 The color profile

One of the most misterious pieces of information is the one that describes the color profile; Luigi Scarso already wrote a paper (Scarso, 2010) where he discusses this problem in connection with the typesetting program ConTEXt-mkiv. But the problem is not the particular typesetting program, rather it is the very concept of *color profile*. I won't go any deeper into this question, but I redirect the reader on the paper Scarso (2010), where the question is thoroughly discussed. I just remark that I have obtained satisfactory results by downloading the color profile file ECI-RGB.V1.0.icc, freely downloadable from the download area of the www.eci.org Internet site. This file may be saved on one's disk somewhere, where the ghostscript program can find it; but I suggest to save this file in a system wide folder and to specify the full address when dealing with this file.

The mentioned color profile is a generic one, and yields satisfactory results in most circumstances; of course it only deals with the RGB (Red, Green, Blue) additive color model (the one that is used by the TV and computer screens) and the pictures inserted into the document file should be consistent with this color model; therefore no picture 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 the link targets are internal to the document; therefore the reader may use the bookmark column of the PDF reader so as to jump from one point to another of the document, and this possibility is particularly useful in a reference document.

External links are prohibited; 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 do exist today, but most likely will not exist any more in fifty years. Therefore when archiving is a requirement, any document should be self contained; if one needs to refer to another document, either uses a complete reference in the document bibliography, or includes the referred document in his/her own one. The choice depends on the author, but s/he must keep in mind the requirement that her/his archived document *must* be self contained.

This is not a trivial constraint; after all it's our daily experience, if we are used to surf the net, that many Internet addresses 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, even better, by means of the \hypersetup command which the package performance is customized with.

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 (Beccari, 2010) in order to find a patch to "heal" the cmsy* fonts that use some zero-width glyphs. This happened with the third math family fonts, the one that contain only 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 besides, another symbol. Any zero-width glyph is not PDF/A compliant.

With XeLaTeX the font choice is much wider, although for what concerns math typesetting up to now there are few OpenType UNICODE encoded fonts suitable for the task; 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 (those that have the widest choice of math glyphs) and I did not find any glitch.

On the opposite I found out something that either I neglected in the past or that more modern checking programs can spot; it's the "normal" Type 1 Computer Modern OT1 encoded fonts that have some problems with the use of accents.

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 UNICODE encoded CM-Unicode fonts (that contain the full accented set of Latin letters, the Cyrillic ones and the full accented set of Greek letters, suitable for the Greek polytonic spelling), and there would not be any problem. But the problem might show up when you include in your document either PDF files or pages extracted from other documents, when the latter ones were typeset by means of 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 nobody should actually use the OT1 encoded fonts if the docu-

ment they typeset contains even a single accented word. In facts 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 apparently is acceptable when a document is printed, but in effects 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 extremely useful the free program Adobe Reader X (actually any recent version will do the task), where by pressing the Ctrl +D (or the cmd + D key on Mac computers) a dialog pane is opened 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 control that no Type 3 (bitmapped) fonts are used and, if any CM font is being used, the user may examine the document in order 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 indicated 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 pictures 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 the RGB one; it is admissible also the "gray" model, since the RGB one can render the gray nuances very well. But the bitmapped format may be acceptable for photographs with a pixel density of at least 300 dpi, but in general it is not valid for line art and textual files. The compression method used by the JPEG (`.jpg`) is not a lossless one, 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 different compression method and in general it is more suited for line art. Vector graphics, in any case, is more suited for line art even if sometimes it's not possible to have available every line art picture in such a format.

But stretches of text and vector graphics loose their 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.

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

One way to correct the font problem 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 nor 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, or it is possible to draw the poor accent overlay of the CM fonts. I confess I never tried this procedure, but it is Hàn Thế Thành himself (Hàn Thế Thành, 2008) who suggests this procedure.

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 version 8.60 or higher; the more recent the better. At writing time I am using version 9.02.

As already said, we need an auxiliary file in order to introduce the *metadata* and to set up some special PostScript commands necessary for this task. The ghostscript distribution contains a file named `PDFA_def.ps` and is contained in the ghostscript sub tree `.../ghostscript/`⟨*version*⟩`/lib/` (of course change the slash character into a backslash one on Windows platforms); where this sub tree is grafted onto your general system tree depend on your platform and your operating system, but you certainly can execute a search command and find out where all this resides.

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

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, because you cannot use the default application for `.ps` files. On a Window platform you might use

Wordpad; on a Linux one you might use vim, emacs, gedit; on a Mac one you might use Textedit.app, TextWrangler.app, but avoid using TeXShop.app (although you may use TeXworks.app) since this programs, besides the TEX system files, may open also the PDF files, and therefore it 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 `myXeLaTeXfile-def.ps` contains some lines commented with a `% Customize` comment; you should edit these lines the way that is best suited to introduce the necessary *metadata* information. In order to insert the *metadata* suitable for this article I would use the file on page 7.

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

1. The `/ICCProfile`; I changed the default one `ISO Coated sb.icc` into the above mentioned `ECI-RGB.V1.0.icc` and I expressed the full path from the root of my disk to the file; probably I might have used the home folder indicating it with `~`, but the full path is more easily changeable on those Windows platforms that do not have the notion of "home".

2. The document data `/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 does 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 this auxiliary file has the same name as the main file of the document comes handy because it is sure that every document has its own auxiliary file and there is no 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; 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; the bash scripts (for UNIX-like machines) or batch files for Windows platforms come handy for specifying the whole command string without errors and without the risk of forgetting some terms.

The script on page 7 is a bash script where the only attention to be made is to eliminate the end-of-lines in the last long command that must consist in just one line. In order to change the script into a batch file, it's sufficient to use the Windows name for the ghostscript executable (gswin32c in place of gs) and to use `%1`, `%2`, etc. in place of `$1`, `$2`. Comment signs `#` must be changed into 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 with Windows) without forgetting to assign it the proper mode codes such as, for example, `chmod 755 pdf2pdfa`.

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

```
pdf2pdfa myXeLaTeXfile
```

and ghostscript will do the whole work.

Of course the last step is to check the 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; actually 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 should 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 the necessary hooks already exist, but I must leave this further task for another moment.

References

ADOBE (2006). «Adobe's free xmp toolkit for reading/writing xmp». `http://www.adobe.com/devnet/xmp/sdk/eula.html`. In basso a

The auxiliary 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 {
  systemdict /ProcessColorModel get dup /DeviceGray ne exch /DeviceCMYK ne and
} {
  true
} ifelse
{ (ERROR: ProcessColorModel must be /DeviceGray or DeviceCMYK.)=
  /ProcessColorModel cvx /rangecheck signalerror
} 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 bash script

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

sinistra nella schermata puntata dall'URL c'è l'ancora con il link per il download.

Beccari, C. (2010). «Some PDF/A tricks». *The PracTEX Journal*.

Drümmer, O., Oettler, A. e von Seggern, D. (2008). *PDF/A in a Nutshell. Long Term Archiving with PDF*. Callas Software gmbh. In: `http://www.pdfa.org/doku.php?id=pdfa:en:pdfa_in_a_nutshell`.

Hàn Thế Thành (2008). «Generating PDFA compliant PDFs from pdftex». `http://support.river-valley.com/wiki/index.php?title=Generating_PDF/A_compliant_PDFs_from_pdftex`.

Knuth, D. E. (1996). *The TEXbook*. Addison Wesley, Reading, Mass., 16ª edizione.

Martini, E. (2007). «Lo standard PDF/A: Sperimentazione di software per la verifica di conformità allo standard ISO 19005:2005». `http://www.cnipa.gov.it/site/_files/ref02_RS_3.1_martini.pdf`.

PDF/A (2005–2008). «Technical notes». `http://www.pdfa.org/`. This site contains the Technical Notes included in the following files: `tn0001_pdfa-1_and_namespaces_2008-03-19.pdf`, `tn0003_metadata_in_pdfa-1_2008-03-18.pdf`, `tn0006_digital_signatures_in_pdfa-1_2008-03-14.pdf`, `tn0008_predefined_xmp_properties_in_pdfa-1_2008-03-20.pdf`, `tn0009_xmp_extension_schemas_in_pdfa-1_2008-03-20.pdf`.

Radhakrishnan, C. e Hàn Thế Thành (2008). «Generation of PDF/X-1a and PDF/A-1b compliant PDF's with PDFTEX — `pdfx.sty`». `http://tug.ctan.org/tex-archive/macros/latex/contrib/pdfx/`.

Scarso, L. (2010). «Introduction to colours in ConTeXt mkiv». *TUGboat*, **31** (3), pp. 203–207.

▷ Claudio Beccari
Villarbasse (TO)
`claudio dot beccari at gmail dot com`