Adding XMP metadata in LaTeX

Ulrike Fischer, Frank Mittelbach

Abstract

One task of the "LaTeX Tagged PDF Project" [6] is to evaluate existing solutions to add XMP metadata to a PDF, and if needed, to design and implement a new standard interface for this. In this article we will describe the current state of this task.

Contents

1 Introduction

The PDF format offers two places to store metadata. For one there is the *Info dictionary*. It is directly at the root of the PDF structure and contains key–value pairs representing document data, such as the PDF creation date (e.g., `/CreationDate (D:2022100515 3151+02'00')`) and title (e.g., `/Title (Bearwear)`). (Recall that string constants in PDF and PostScript are enclosed in parentheses.) Various standard keys exist, such as `/Title`, `/Author` and `/CreationDate`, but the dictionary can also hold private keys. pdfTeX for example adds its banner: `/PTEX.Fullbanner (This is pdfTeX, ...)`. PDF viewers normally show a selection of the standard keys in the properties of a PDF. All TeX engines offer tools to add content to this dictionary and to change the values added automatically by the engines. In LaTeX the title and the author are normally added with the help of the hyperref package and its `pdftitle` and `pdfauthor` keys.

While the Info dictionary can hold arbitrary data, it is nevertheless only an unordered list and not well suited as a serious data container. So in PDF 1.4 a second option was added to the format: one can embed an XML file with the data and reference this file from the PDF catalog through the `/Metadata` key.[1] The format of the XML is defined as part of a framework called the *Extensible Metadata Platform* (XMP), first described in an Adobe document and now an ISO standard [3], and so commonly these metadata are referred to as *XMP metadata*.

In PDF 2.0, the XMP metadata replaces the deprecated Info dictionary. In earlier PDF versions the metadata has already been required by standards like the various PDF/A and PDF/X versions, PDF/UA (the standard for accessible PDF), and standards for electronic invoice data exchange like ZUGFeRD 2.2/ Factur-X 1.0 [1]. It is therefore quite important to have tools and interfaces to add them to a PDF. In the following we will describe various existing options and give our outlook on future plans in this area.

We assume that all source files are UTF-8 encoded and won't mention places where 8-bit encoded files need perhaps additional care (XMP metadata in the PDF are always UTF-8 encoded).

2 Creating the XMP metadata

It is quite easy to add an XML file to a PDF and to reference it in the catalog. In all engines this can be done with a few lines of code; the small package xmpincl [9] demonstrates it for pdfLaTeX. The challenge is to correctly build the content.

- At first, as always with XML there is quite a large amount of formal syntax to understand and follow.
- PDF standards contain further demands on the content and the structure of the XML. As an example, properties that don't count as predefined [8] must be declared in extension schemas, and if data like a title is present in the Info dictionary and in the XMP metadata they must match — something that is not easy to ensure as they use different encodings and formatting and so this requires concrete tests with PDF viewers and validators.
- Then one must decide which XMP metadata should be supported (various more or less standard name spaces exist here) and devise user interfaces for the data that can't be detected automatically.
- User input must be sanitized, properly escaped for use in an XML file and converted to UTF-8.

In the past, two LaTeX packages took on this task. They don't produce exactly the same XMP metadata, but the differences are small; on the whole, they settled more or less on the same set.

[1] It is possible to add more XML files and to reference them from other parts of the PDF but in this article we restrict the discussion to the document-wide data container.

doi.org/10.47397/tb/43-3/tb135fischer-xmp

Listing 1: A selection of XMP metadata added automatically by hyperxmp

```
<pdf:PDFVersion>1.5</pdf:PDFVersion>
<dc:format>application/pdf</dc:format>
<xmp:CreateDate>2022-10-06T10:27:33+02:00
  </xmp:CreateDate>
<xmp:CreatorTool>LaTeX with hyperref
  </xmp:CreatorTool>
<xmpMM:DocumentID>
  uuid:aef2b675-9b18-4d18-97f7-a3339b139000
</xmpMM:DocumentID>
```

Listing 2: hyperxmp example with additional \hypersetup keys

```
\documentclass{article}
\usepackage{hyperxmp}
\usepackage{hyperref}
\hypersetup
 {
  pdftitle = {Über einen die Erzeugung und
    Verwandlung des Lichtes betreffenden
    heuristischen Gesichtspunkt},
  pdfauthor={Albert Einstein},
  pdflang = {de},
  pdfmetalang={de},
  pdfdate={1905-03-17},
  pdfcontactcity={Bern},
  pdfcontactcountry={Switzerland},
  pdfissn={0003-3804},
  pdfdoi={10.1002/andp.19053220607},
 }
\begin{document}
Text
\end{document}
```

2.1 The hyperxmp package

Simply loading the hyperxmp package from Scott Pakin [7] will add XMP metadata. Listing 1 shows some selected lines.

hyperxmp supports all major compilation routes: pdfLaTeX, LuaLaTeX, XeLaTeX, LaTeX with `dvips`,[2] (u)(p)LaTeX with DVIPDFMx.

For the user interface, the package hooks into the `\hypersetup` command of hyperref. It retrieves the values of native hyperref keys like `pdftitle` and `pdfauthor` and defines new keys for a variety of additional XMP metadata. Listing 2 shows a few examples. The hyperxmp package supports quite a large set of metadata tags but has no interface to extend this set.

hyperref is used by hyperxmp not only for the user interface but also to sanitize the user input: All user input is first converted by `\pdfstringdef` to the format used in bookmarks and then back to UTF-8; it can contain arbitrary Unicode characters and all commands supported by hyperref in the bookmarks. If hyperref is missing it is loaded automatically by hyperxmp at the end of the preamble. If the document should contain XMP metadata but not links or bookmarks, load hyperref with the package option `draft`.

Individual items in the author and keyword lists should be separated by commas; if a real comma is wanted `\xmpcomma` must be used.

Some keys allow adding language variants with the command `\XMPLangAlt`:

```
\hypersetup{pdflang=de,pdftitle=Mein Titel}
\XMPLangAlt{en}{pdftitle={My title}}
```

This results in metadata using the `xml:lang` attribute:

```
<dc:title>
 <rdf:Alt>
  <rdf:li xml:lang="x-default">Mein Titel
    </rdf:li>
  <rdf:li xml:lang="de">Mein Titel</rdf:li>
  <rdf:li xml:lang="en">My title</rdf:li>
 </rdf:Alt>
</dc:title>
```

2.2 The pdfx package

The goal of the pdfx package [2], currently maintained by Ross Moore, is to support the generation of PDF/X-, PDF/A- and PDF/E-compliant documents. As XMP metadata are required by the standards they are created by the package, but it will (in part depending on the requested standard) also embed a color profile, change ToUnicode values, redefine math accents and more.

pdfx can only be used with pdfLaTeX, LuaLaTeX and XeLaTeX. (It makes use of the xmpincl package written for pdfLaTeX for the actual embedding of the XMP data and tweaks it a bit to make it compatible with the two other engines). The compilation with XeLaTeX requires the use of `--shell-escape` as pdfx calls LuaLaTeX to retrieve the creation date of the file; this can be avoided by defining `\pdfcreationdate` manually before loading pdfx, e.g.[3]

```
\def\pdfcreationdate
  {\string D:20221002224824+10'00'}
```

[2] But sadly Ghostscript has no option to add the XMP metadata as an uncompressed stream.

[3] The `\string` is needed as pdfx expects a `D` with catcode other.

Listing 3: Example input with the pdfx package

```
\begin{filecontents}[force]{\jobname.xmpdata}
 \Title{Baking through the ages}
 \Author{A. Baker\sep C. Kneader}
 \Language{en-GB}
 \Keywords{cookies\sep muffins\sep cakes}
 \Publisher{Baking International}
\end{filecontents}
\documentclass{article}
\usepackage[a-1b]{pdfx}
\begin{document}
 some text
\end{document}
```

When using pdfx the user must provide metadata in an external file with the extension `.xmpdata`. It is recommended to create this file at the beginning of the document with a `filecontents` environment. In this `.xmpdata` file the data are then given as arguments of various commands—Listing 3 shows an example. The arguments can contain arbitrary Unicode characters, and don't need to be printable in the document. Commands for non-Latin characters like `\CYRD` (cyrillic), `\hebzayin` (hebrew) or `\textarabicfa` can also be used, if enabled through package options like `cyrxmp` and `hebxmp`.

pdfx loads hyperref as it, like hyperxmp, uses `\pdfstringdef` to sanitize some of the input. It also uses hyperref to set the PDF version. As hyperref normally should be loaded late but the PDF version must be set early, getting the loading order right can get tricky.

The default setup for the XMP metadata is stored in external templates,[4] `pdfa.xmp`, `pdfx.xmp`, `pdfe.xmp`, ... By copying these templates and adjusting them it is possible to add private data. As a side effect of the processing the resulting XMP data are written to a file, `pdfa.xmpi` or `pdfx.xmpi`, which allows to feed them to an RDF validator [11].

Lists are created if the items are separated by the command `\sep`. Language support is offered through optional arguments:

```
\Title[en]{Baking through the ages}
\Keywords[de]{Kekse%
   \sep[en]Cookies%
   \sep[fr]Biscuits}
```

This will result in:

```
<dc:title>
 <rdf:Alt>
  <rdf:li xml:lang="en">
    Baking through the ages
  </rdf:li>
 </rdf:Alt>
</dc:title>
<dc:subject>
 <rdf:Bag>
  <rdf:li xml:lang="de">Kekse</rdf:li>
  <rdf:li xml:lang="en">Cookies</rdf:li>
  <rdf:li xml:lang="fr">Biscuits</rdf:li>
 </rdf:Bag>
</dc:subject>
```

2.3 hyperxmp and pdfx clash

The two packages can't be used in the same document. As both use low-level primitive commands to reference their XMP stream, the PDF catalog would contain two `/Metadata` entries if both packages are used, and this is invalid in a PDF:

```
/Metadata 3 0 R /Metadata 6 0 R
```

3 XMP metadata with the LaTeX PDF management support

Over the past years the LaTeX Project Team has in connection with the Tagged PDF project [6] written and released code targeting various PDF-related tasks. Some of this code is already included in the kernel through the l3pdf module of the L3 programming layer [4], the rest is provided through the external bundle pdfmanagement-testphase [5].

The goals of this code are, first, to provide abstracted, backend independent commands. So for example instead of setting the pdf version directly (depending on the engine: with `\pdfminorversion`, or `\pdfvariable minorversion` or with a `\special`), you can use `\pdf_gset_version:n{1.7}` with all supported backends. hyperref already makes use of this: when the PDF management is active the same generic driver is loaded for all backends.

The second goal of the new code (and the reason why it is called PDF *management*) is to prevent clashes like the one between hyperxmp and pdfx mentioned above, by providing interfaces for *managed* access to central PDF resources. If for example two packages try to add a `/Metadata` reference or some other resources to the catalag, then the code will ensure that the conflict is correctly resolved — either by merging the resources or by rejecting one of the requests.

The PDF management doesn't work with packages that bypass the interfaces by using primitive

[4] Depending on the standard, pdfx will create slightly different metadata.

commands. This means neither hyperxmp nor pdfx are usable with it, and a replacement to add the important XMP metadata was needed. As an intermediate solution hyperxmp was patched but as part of task 2.3.4 of the feasibility study [6] this patch has now been replaced by proper support in the l3pdfmeta package of the pdfmanagement-testphase bundle.

3.1 Design goals for the XMP metadata support

After reviewing the existing packages the following main design goals of the new XMP metadata support have been identified:

- The dependency to hyperref should be removed. XMP metadata are not directly related to links and other interactive features and should work also in documents which don't use them.
- The standard interface should be a key–value system with `\DocumentMetadata` as the default interface command. To ease the transition from hyperxmp the `\hypersetup` keys should continue to work where possible.
- The input should support the full range of Unicode and standard commands.
- The default set of supported XMP tags should be similar to the set of the existing packages.
- There should be no need in the user input for special commands like `\xmpcomma`, `\XMLlangalt` or `\sep`. List items should be input as comma lists where "real commas" are protected by braces as usual. The language alternatives can be set with optional arguments.
- As with pdfx it should be possible to export the XMP metadata in an external document, but this should be a debug option.
- The XMP metadata should be extensible. As a proof of concept, an example document showing how to add the metadata needed for a ZUGFeRD document should be developed.

3.2 Implementation of the design goal

The new XMP metadata support has been implemented in the l3pdfmeta module and is loaded together with PDF management code. This is done by using the `\DocumentMetadata` command with key–value pairs at the beginning of the document. XMP metadata are then automatically added to the PDF they can be suppressed by setting the key `xmp` to `false` (see listing 4). This works with all engines supported by the L3 layer backend.

In accordance with the goals specified above the minimal document doesn't require hyperref: The code relies on `\text_purify:n` and other functions from the L3 layer to sanitize the input.

Listing 4: Minimal input for XMP metadata with the PDF management

```
\DocumentMetadata
  {
   %xmp=false, % no XMP
    xmp=true   % optional as default
  }
\documentclass{article}
\begin{document}
abc
\end{document}
```

XMP metadata for the PDF standard,[5] the PDF version and the language are already retrieved from keys set in `\DocumentMetadata`:

```
\DocumentMetadata
  {
   pdfstandard = a-2b,
   pdfversion  = 1.7,
   lang        = de
  }
```

At the moment other metadata still requires the use of the hyperref and the `\hypersetup` interface with the hyperxmp keys known from listing 2 — we haven't decided yet how to name and organize suitable keys in the `\DocumentMetadata` command.

As outlined in the design goals, lists are input as comma lists with real commas protected by braces as usual. Where sensible it is possible to add a language tag with an optional argument before the item. The next listing demonstrates both for the title:

```
\hypersetup
 { pdftitle=
    {[en]Baking,
     [de]{Kekse, Kuchen und Torten backen}}}
```

This results in this metadata:

```
<dc:title>
  <rdf:Alt>
   <rdf:li xml:lang="en">Baking</rdf:li>
   <rdf:li xml:lang="de">Kekse,
     Kuchen und Torten backen</rdf:li>
  </rdf:Alt>
</dc:title>
```

[5] As with pdfx, declaring a standard can have further effects; for example, embed a color profile or do some validations. Be aware that LaTeX can neither ensure nor check all requirements of any given standard, and an external validator like veraPDF [10] should be used.

The XMP metadata can be exported to an external file — the default name is `\jobname.xmpi` — with a `debug` setting in `\DocumentMetadata`:

```
\DocumentMetadata
  {
   debug = { xmp-export, more debug options ... }
  }
```

Finally, first steps have been undertaken to extend the XMP metadata. To support, for example, the ZUGFeRD standard, additions to the XMP metadata are needed in three places:

- A new XML namespace with a suitable prefix must be declared.
- A new schema with declarations for the new tags must be added to the `pdfaExtension:schemas` section.
- And the data itself must be added.

For all these tasks internal functions have been defined, and a first prototype that implements the ZUGFeRD 2.2 standard exists, but it isn't yet clear what the public interface should look like.

3.3 Status and outlook

The new code supports XMP metadata at a comparable level to that provided by the existing packages. Almost all of the above design goals are already implemented. There remains some work to do to provide suitable public interfaces for certain parts. Also needed are more tests with PDF viewers and validators to check if their interpretation of the standards agrees with the code. Feedback and comments are welcome!

References

[1] Forum for Electronic Invoicing Germany. What is ZUGFeRD? `www.ferd-net.de/standards/what-is-zugferd`

[2] Hàn Thế Thành, P. Selinger, et al. *The pdfx package*. `ctan.org/pkg/pdfx`

[3] International Organization for Standardization. *ISO 16684-1:2019: Graphic technology — Extensible metadata platform (XMP) specification — Part 1: Data model, serialization and core properties*. 2nd ed., 2019. `www.iso.org/obp/ui/#!iso:std:75163:en`

[4] LaTeX Project Team. *The l3kernel package*. `ctan.org/pkg/l3kernel`

[5] LaTeX Project Team. *The pdfmanagement-testphase package*. `ctan.org/pkg/pdfmanagement-testphase`

[6] F. Mittelbach, U. Fischer, C. Rowley. LaTeX tagged PDF feasibility evaluation. `latex-project.org/publications/2020-tagged-pdf-feasibility.pdf`

[7] S. Pakin. *The hyperxmp package*. `ctan.org/pkg/hyperxmp`

[8] PDF competence center. Technote 0008: Predefined XMP properties in PDF/A-1. Technical report, 2008. `www.pdfa.org/resource/technical-note-tn0008-predefined-xmp-properties-in-pdfa-1/`

[9] M. Sneep. *The xmpincl package*. `ctan.org/pkg/xmpincl`

[10] veraPDF consortium. veraPDF—industry supported PDF/A validation. `verapdf.org`

[11] W3C. W3C RDF validation service. `www.w3.org/RDF/Validator/`

◇ Ulrike Fischer
LaTeX Project Team
Bonn
Germany
`ulrike.fischer (at) latex-project.org`

◇ Frank Mittelbach
LaTeX Project Team
Mainz
Germany
`frank.mittelbach (at) latex-project.org`