AcroTEX.Net

The aebXMP Package

Updating XMP using E4X and LaTeX

D. P. Story

Prepared: March 22, 2017

www.acrotex.net
Version 2.5a, 2017/02/17

Table of Contents

1. Introduction

The motivation for the development of this package came from Herr Jürgen Gilg, who had a need to fill in the metadata fields beyond those normally populated by using hyperref: Title, Author, Subject and Keywords. Of particular interest to him were the metadata fields Copyright Status, Copyright Notice and Copyright Info URL.

After doing some research on the CTAN archives, I came across the hyperxmp package by Scott Pakin.[1] The package works well with pdftex and dvipdfm, but has a bit of a problem when using the distiller. For this reason, I sought my own solution to the problem.

As a beta tester of Acrobat 8 Professional, I had the opportunity to use some of the new capabilities of the JavaScript interpreter as an alternate approach to the one used by Mr. Patkin. The JavaScript version 1.6 interpreter, the one used in version 8, comes with E4X, an XML parser, built in. I could see that E4X could be exploited to manipulate the XMP data, and this was my approach.

2. Requirements

The techniques used by the aebxmp package to update the XMP data require the AcroTEX eDucation Bundle (AeB), freely available from www.acrotex.net. Because E4X is used, we also require Acrobat 8 Pro or later, and, since you have Acrobat 8 or later, my LaTeXing friend, this package will work for you with all workflows: dvips/Distiller, pdflatex (including lualatex), and xelatex. To emphasize, for non-Distiller workflow, the full Acrobat application is still required to be your default PDF viewer on your own computer.

3. The Test File

The package aebxmp has a simple test file, aebxmp_test.tex, which is found in the examples folder. After you build the PDF and open the document in Acrobat for the first time, the new metadata is imported; *it is important to save the document* after the data is imported.

save the document

To use this package, you must have, in addition to Acrobat 8 Pro (or later), installed on your computer a standard TeX system, including the latest version of AeB.[2]

4. Metadata

The hyperref package provides basic metadata support, providing a mechanism for providing the title, author, subject, and keywords. Beyond these, we can include additional metadata.

[1] The reader is invited to read the documentation of the hyperxmp, as contained therein is a good discussion of XMP (eXtensible Metadata Platform).

[2] AeB can be downloaded from any CTAN server, from www.math.uakron.edu/~dpstory/webeq.html, or from www.acrotex.net.

4.1. Author, Author

The hyperref package allows you to specify an author or authors; the Web package uses this through the \title command, which passes its argument on to hyperref of inclusion in the PDF Info dictionary. The authors names are not individually assessable through JavaScript. In Acrobat 9, I believe, the JavaScript API includes the Doc.info.Authors property, its value is an array of authors, We can access each author using array notation: First author is this.info.Authors[0], second author is this.info.Authors[1], and so on. The number of authors is obtained using the length property of arrays, this.info.Authors.length.

The aebxmp interface to this is through the \Authors command. It takes a list of authors, each enclosed in braces, no commas (,) between the authors. Like so:

```
\Authors{{D. P. Story}{J\u00FCrgen Gilg}}
```

Using \Authors will overwrite the author(s) named in the \author command (of web), or more generally, passed to \hypersetup{pdfauthor={⟨*author*⟩}} of hyperref.[3]

If you do use \Authors and overwrite the author(s) as passed through hyperref, the first author listed will be the one returned by this.info.Author (a JavaScript property) in the Document Properties dialog box (Ctrl+D), all authors are listed in a semi-colon delimited list.

4.2. The \Keywords Command

The \Authors command utilizes Doc.info.Authors, which takes an array of authors names. There is no such property available for Doc.info.Keywords, the value of this property takes only a string of keywords. The keywords are stored by Acrobat in three ways, (1) pdf:Keywords; (2) in the **Info** dictionary; and (3) in dc:subject. In the latter case, dc:subject takes a Bag of subjects (keywords). A Bag, in XMP parlance, is an unordered array.

The command \Keywords takes a comma delimited list of keywords. Notice the word is capitalized to distinguish it from \keywords, which is defined in the Web package as the interface to inserting the keywords, via hyperref, into the **Info** dictionary.

```
\Keywords{AcroTeX.Net,XMP,E4X,Adobe Acrobat,JavaScript}
```

The command takes each comma-delimited list of keywords and inserts each into the dc:subject part of the metadata. For this document, the keywords appear in the XML metadata as

```
<dc:subject>
   <rdf:Bag>
      <rdf:li>AcroTeX.Net</rdf:li>
      <rdf:li>XMP</rdf:li>
      <rdf:li>E4X</rdf:li>
      <rdf:li>Adobe Acrobat</rdf:li>
```

[3] This may be to your advantage when using Web; the value of \author is used to display the author(s) of the document, you may want to present the names one way on the title page, for example, and another way in the Description tab of the Document Properties dialog box.

```
        <rdf:li>JavaScript</rdf:li>
      </rdf:Bag>
   </dc:subject>
```

When the document is first opened, the following command is executed

```
this.info.Keywords="AcroTeX.Net; XMP; E4X; Adobe Acrobat; JavaScript"
```

Here, I have broken the string across lines for readability. This inserts the same list of keywords into the **Info** dictionary.

When you executed `this.info.Keywords` in the console, you'll get

```
AcroTeX.Net; XMP; E4X; Adobe Acrobat; JavaScript
```

as expected.

To access the individual keywords, I've defined an array of keywords, `aKeywords`. It takes as its argument the index of the keyword you want to get; For example, executing `aKeywords[0]` in the console returns a value of `"AcroTeX.Net"`, while `aKeywords[4]` returns a value of `"JavaScript"`. If you execute `aKeywords[5]`, a value of `undefined` is returned. The number of keywords is `aKeywords.length`.

Listing the keywords may be of interest to someone, it is an exercise to me.[4]

The command `\xmpDoNotInsKWScript`, when expanded in the preamble, will turn off the creation of the array `aKeywords`.

4.3. Additional Metadata

As mentioned previously, the `aebxmp` package addresses three areas of interest: Setting the Copyright Status, Copyright Notice, and the Copyright Info URL. Obviously, other elements of the XMP can be addressed. To that end, the `aebxmp` package defines five new LaTeX commands to populate the values of the five metadata fields Copyright Status, Copyright Notice, Copyright Info URL, Author Title, and Writer Description. Values for the arguments of these commands are documented below.

`\copyrightStatus{True|False}`: If `True`, Copyright Status is set to `Copyrighted`; if `False`, Copyright Status is set to Public Domain. If left empty, the status is set to `Unknown`.

Unless you've executed `\xmpDoNotInsKWScript` in the preamble, the `aebxmp` defined JavaScript function `getCopyrightStatus()` is available. The function returns the copyright status: `Copyrighted`, `Public Domain`, and `Unknown`.

`\copyrightNotice{`⟨*text*⟩`}`: The ⟨*text*⟩ of the Copyright Notice is defined

`\copyrightInfoURL{`⟨*URL*⟩`}`: The ⟨*URL*⟩ to the copyright information

Unless you've executed `\xmpDoNotInsKWScript` in the preamble, the `aebxmp` defined JavaScript function `getCopyrightInfoURL()` is available. The function returns the copyrightinfo URL (⟨*URL*⟩).

[4] Actually, this is the way **Acrobat** handles a comma-delimited list of keywords when the words are entered through the user interface; it puts them in a `Bag`.

`\authortitle{`⟨*text*⟩`}`: The ⟨*text*⟩ appears in the Author Title line on the Additional Metadata dialog box. This is a Photoshop property. (See the Advanced category in the left panel.)

Unless you've executed `\xmpDoNotInsKWScript` in the preamble, the `aebxmp` defined JavaScript function `getAuthorTitle()` is available. The function returns the authortitle (⟨*text*⟩).

`\descriptionwriter{`⟨*text*⟩`}`: The ⟨*text*⟩ appears in the Description Writer line on the Additional Metadata dialog box. This is a Photoshop property. (See the Advanced category in the left panel.)

Unless you've executed `\xmpDoNotInsKWScript` in the preamble, the `aebxmp` defined JavaScript function `getDescriptionWriter()` is available. The function returns the descriptionwriter (⟨*text*⟩).

For example, for this document, we have in the preamble,

```
\copyrightStatus{True}
\copyrightNotice{Copyright (C) 2006-\the\year, D. P. Story}
\copyrightInfoURL{http://www.acrotex.net}
\authortitle{Programming and Development, AcroTeX.Net}
\descriptionwriter{Testing and Promotions Department, AcroTeX.Net}
```

Enter unicode (`\uXXXX`) directly into the `\copyrightNotice` and `\copyrightInfoURL` fields; for example,

```
\copyrightNotice{Copyright (C) 2006-\the\year, J\u00FCrgen Gilg}
```

Unicode can be used all the metadata commands discussed in the manual.

The `\copyrightNotice` can take multiple arguments, one for each language. The syntax is

```
\copyrightNotice{%
    {[⟨lang1⟩] copyright in this language}
    {[⟨lang2⟩] copyright in this language}
    ...
}
```

The first copyright item listed is also counted as the default language and will be marked as `x-default` as the value of the `xml:lang` attribute. Contrary to LaTeX custom, the brackets do not indicate optional arguments, they are required except for the first item in the list. Each group, which are enclosed in braces (`{}`), represents a copyright notice; the part enclosed on brackets (`[]`) contains the language designator. This is a two-letter code to indicate the language; it can also have a sub-tag to indicate a country (see `en-US` in example below). See the ISO 639-1 standard, and the RFC 3066 standard, referenced at the end of the manual, for more information on language codes.

For example,

```
\copyrightNotice{%
    {[en-US]Copyright (C) \the\year, D. P. Story}
    {[fr]Copyright (C) \the\year, D. P. Story}
    {[de]Copyright (C) \the\year, D. P. Story}
}
```

There is also a \sourceFile command that takes one argument. If \sourceFile does not appear in the preamble, the \jobname.tex is written to the metadata (as part of the Dublin Core Properties). If \sourceFile{} is expanded in the preamble, no source file data will be inserted into the metadata. Finally, executing the command \sourceFile{hw01_1100.tex} causes the string hw01_1100.tex to be written as the value of the dc:source key.

4.4. \Title, \Subject, and \metaLang

The Title and Subject keys can also be recorded with alternate languages; for this reason, aebxmp defines \Title and \Subject. The syntax of these two are similar to \copyrightNotice, described above

```
\Title{%
    {[⟨lang1⟩] title in this language}
    {[⟨lang2⟩] title in this language}
    ...
}
\Subject{%
    {[⟨lang1⟩] subject in this language}
    {[⟨lang2⟩] subject in this language}
    ...
}
```

The first one listed is also designated as the default language, marked with x-default.

For example,

```
\Title{%
    {[en-US]Testing the aebxmp Package}
    {[fr]Test du paquet aebxmp}
    {[de]Testen des aebxmp Pakets}
}
\Subject{%
    {[en-US]Test file for using E4X to update the XMP Data Model}
    {[fr]Fichier de test utilisant E4X pour mettre à jour
            le modèle de données XMP}
    {[de]Testdatei für die Verwendung von E4X, um das XMP Daten
            Modell zu aktualisieren}
}
```

Note that literal characters such as ü are also recognized so that unicode is not needed here.

The arrays aTitle and aSubject are defined in the document, unless the command \xmpDoNotInsKWScript is executed in the preamble. For example, if you executed aTitle[0] in the console (or part of a JavaScript action of a button), the array element is seen to be "[x-default]: Testing the aebxmp Package", while aTitle[1] is "[en-US]: Testing the aebxmp Package". The array aSubject behaves similarly.

The data passed by \Title and \Subject overrides the data passed by the web commands \title and \subject, and overrides the data passed by the hyperref keys pdftitle and pdfsubject.

Special characters need to be entered using unicode (\uXXXX), not the octal or unicode generated by hyperref. Do not use LaTeX markup that expands to special characters inside the arguments of any of the commands defined in this package.

The \metaLang command allows you to document the languages used in the metadata. Multiple languages may be specified.

```
\metaLang{⟨lang1⟩,⟨lang2⟩,..,⟨langn⟩}
```

For example

```
\metaLang{en,en-US,fr,de}
```

4.5. Custom Document Properties

You can define custom properties using the \customProperties command.

Standard Syntax

```
\customProperties
{%
    {name=⟨name1⟩,value=⟨value1⟩}
    {name=⟨name2⟩,value=⟨value2⟩}
}
```

Colon Syntax

```
\customProperties
{%
    {name:⟨name1⟩,value:⟨value1⟩}
    {name:⟨name2⟩,value:⟨value2⟩}
}
```

The 'colon' syntax is also recognized, but do not mix the two syntaxes together, use either all equal signs or all colons.

The value of the name key requires a unique name, which must not be one of the standard property names Title, Author, Subject, Keywords, Creator, Producer, CreationDate, ModDate, and Trapped. The value of name shall consist of the characters A–Z, a–z, and 0–9, and beginning with a letter. The value may contain unicode, for example, in the preamble of this document we have,

```
\customProperties
{%
    {name=Developer,value={D. P. Story, Esq.}}
    {name=Motivator,value=J\u00FCrgen Gilg}
}
```

Instead of unicode, this same set of custom properties can be defined as follows:

```
\customProperties
{%
    {name=Developer,value={D. P. Story, Esq.}}
    {name=Motivator,value=Jürgen Gilg}
}
```

That is, using literal characters, if your editor supports it.

Info The properties may be accessed through the info property of the Doc object. The button (on the left) opens the console debugger window and displays all the document properties.

The custom properties may be viewed using the user interface; press Ctrl+D and choose the Custom tab.

For more information on this topic, see Part 3, Storage in Files, section 3.2 on **Native metadata in PDF files**, in particular, see section 3.2.1 concerning user-defined document properties.

5. Checking for validity

While looking at this file in Acrobat (or Adobe Reader), press Ctrl+D to get the Document Properties dialog box. Select the Description tab and click Additional Metadata. Since this document was built using the aebxmp package, with the declarations

```
\copyrightStatus{True}
\copyrightNotice{Copyright (C) 2006-\the\year, D. P. Story}
\copyrightInfoURL{http://www.acrotex.net}
\authortitle{Programming and Development, AcroTeX.Net}
\descriptionwriter{Testing and Promotions Department, AcroTeX.Net}
```

In the Advanced Metadata dialog box, you should see,

Copyright Status: Copyrighted

Copyright Notice: Copyright (C) 2006-2017, D. P. Story

Copyright Info URL: http://www.acrotex.net

Author Title: Programming and Development, AcroTeX.Net

Description Writer: Testing and Promotions Department, AcroTeX.Net

in addition to the usual Document Title, Author, Description, and Keywords. I promise you that I did not enter these values through the user interface. :-)

6. Resources

The resources for the development of this package are

- *Standard ECMA-357: ECMAScript for XML (E4X) Specification*, http://www.ecma-international.org/publications/standards/Ecma-357.htm
- *XMP Specification*, http://www.adobe.com/devnet/xmp.html
- *Acrobat JavaScript Scripting Reference*, Version 8.0 http://www.adobe.com/go/acrobat_developer
- hyperxmp package by Scott Pakin, http://ctan.org/pkg/hyperxmp
- The AcroTeX System Tools, available for free download at www.acrotex.net. This is a LaTeX-based system.
- ISO 639-1 two-letter abbreviation. http://www.loc.gov/standards/iso639-2/php/English_list.php
- IETF RFC 3066 http://www.ietf.org/rfc/rfc3066.txt

Now, I simply must get back to my retirement. DPS