

The hyperxmp package^{*}

Scott Pakin
scott+hyxmp@pakin.org

September 19, 2012

Abstract

`hyperxmp` makes it easy for an author to include XMP metadata in a PDF document produced by `LATEX`. `hyperxmp` integrates seamlessly with `hyperref` and requires virtually no modifications to a document that already specifies document metadata through `hyperref`'s mechanisms.

1 Introduction

Adobe Systems, Inc. has been promoting XMP [4]—eXtensible Metadata Platform—as a standard way to include metadata within a document. The idea behind XMP is that it is an XML-based description of various document attributes and is embedded as uncompressed, unencoded text within the document it describes. By storing the metadata this way it is independent of the document's file format. That is, regardless of whether a document is in PDF, JPEG, HTML, or any other format, it is trivial for a program (or human) to locate, extract, and—using any standard XML parser—process the embedded XMP metadata.

As of this writing there are few tools that actually do process XMP. However, it is easy to imagine future support existing in file browsers for displaying not only a document's filename but also its title, list of authors, description, and other metadata.

This is too abstract! Give me an example. Consider a `LATEX` document with three authors: Jack Napier, Edward Nigma, and Harvey Dent. The generated PDF file will contain, among other information, the following stanza of XMP code embedded within it:

```
<dc:creator>
  <rdf:Seq>
    <rdf:li>Jack Napier</rdf:li>
    <rdf:li>Edward Nigma</rdf:li>
    <rdf:li>Harvey Dent</rdf:li>
```

^{*}This document corresponds to `hyperxmp` v2.1, dated 2012/09/16.

```
</rdf:Seq>
</dc:creator>
```

In the preceding code, the `dc` namespace refers to the Dublin Core schema, a collection of metadata properties. The `dc:creator` property surrounds the list of authors. The `rdf` namespace is the Resource Description Framework, which defines `rdf:Seq` as an ordered list of values. Each author is represented by an individual list item (`rdf:li`), making it easy for an XML parser to separate the authors' names.

Remember that XMP code is stored as *metadata*. It does not appear when viewing or printing the PDF file. Rather, it is intended to make it easy for applications to identify and categorize the document.

What metadata does hyperxmp process? hyperxmp knows how to embed all of the following types of metadata within a document:

- authors (`dc:creator`)
- base URL (`xmp:BaseURL`)
- copyright (`dc:rights` and `xmpRights:Marked`)
- date (`dc:date`, `xmp:CreateDate`, `xmp:ModifyDate`, and `xmp:MetadataDate`)
- document identifier (`xmpMM:DocumentID`)
- document instance identifier (`xmpMM:InstanceID`)
- file format (`dc:format`)
- keywords (`pdf:Keywords` and `dc:subject`)
- language (`dc:language`)
- L^AT_EX file name (`dc:source`)
- license URL (`xmpRights:WebStatement`)
- metadata writer (`photoshop:CaptionWriter`)
- PDF-generating tool (`pdf:Producer` and `xmp:CreatorTool`)
- PDF version (`pdf:PDFVersion`)
- primary author's position/title (`photoshop:AuthorsPosition`)
- summary (`dc:description`)
- title (`dc:title`)

More types of metadata may be added in a future release.

How does `hyperxmp` compare to the `xmpincl` package? The short answer is that `xmpincl` is more flexible but `hyperxmp` is easier to use. With `xmpincl`, the author manually constructs a file of arbitrary XMP data and the package merely embeds it within the generated PDF file. With `hyperxmp`, the author specifies values for various predefined metadata types and the package formats those values as XMP and embeds the result within the generated PDF file.

`xmpincl` can embed XMP only when running under pdf \LaTeX and only when in PDF-generating mode. `hyperxmp` additionally works with a few other PDF-producing \LaTeX backends.

`hyperxmp` and `xmpincl` can complement each other. An author may want to use `hyperxmp` to produce a basic set of XMP code, then extract the XMP code from the PDF file with a text editor, augment the XMP code with any metadata not supported by `hyperxmp`, and use `xmpincl` to include the modified XMP code in the PDF file.

2 Usage

`hyperxmp` works by postprocessing some of the package options honored by `hyperref`. To use `hyperxmp`, merely put a `\usepackage{hyperxmp}` in your document's preamble. That line can appear anywhere before the `hyperref` PDF options are specified (i.e., with either `\usepackage[...]{hyperref}` or `\hypersetup{...}`). `hyperxmp` will construct its XMP data using the following `hyperref` options:

- `baseurl`
- `pdfauthor`
- `pdfkeywords`
- `pdflang`
- `pdfproducer`
- `pdfsubject`
- `pdftitle`

`hyperxmp` instructs `hyperref` also to accept the following options, which have meaning only to `hyperxmp`:

- `pdfauthortitle`
- `pdfcaptionwriter`
- `pdfcopyright`
- `pdflicenseurl`
- `pdfmetalang`

`pdfauthor` indicates the primary author's position or title. `pdfcaptionwriter` specifies the name of the person who added the metadata to the document. `pdfcopyright` defines the copyright text. `pdflicenseurl` identifies a URL that points to the document's license agreement. `pdfmetalang` indicates the natural language in which the metadata is written, typically as an IETF language tag [6], for example, “en” for English, “en-US” for specifically United States English, “de” for German, and so forth. If `pdfmetalang` is not specified, `hyperxmp` assumes the metadata language is the same as the document language (`hyperref`'s `pdflang` option). If neither `pdfmetalang` nor `pdflang` is specified, `hyperxmp` uses only “x-default” as the metadata language. Note that “x-default” metadata is always included in addition to the specified metadata language, as the user reading the document may not have specified a language preference.

It is usually more convenient to provide values for those options using `hyperref`'s `\hypersetup` command than on the `\usepackage` command line. See the `hyperref` manual for more information. The following is a sample L^AT_EX document that provides values for most of the metadata options that `hyperxmp` recognizes:

```
\documentclass{article}
\usepackage{hyperxmp}
\usepackage{hyperref}
\title{%
  On a heuristic viewpoint concerning the production and
  transformation of light}
\author{Albert Einstein}
\hypersetup{%
  pdftitle={%
    On a heuristic viewpoint concerning the production and
    transformation of light},
  pdfauthor={Albert Einstein},
  pdfauthortitle={Technical Assistant, Level III},
  pdfcopyright={Copyright (C) 1905, Albert Einstein},
  pdfsubject={photoelectric effect},
  pdfkeywords={energy quanta, Hertz effect, quantum physics},
  pdflicenseurl={http://creativecommons.org/licenses/by-nc-nd/3.0/},
  pdfcaptionwriter={Scott Pakin},
  pdflang={en},
  baseurl={http://www.ctan.org/tex-archive/macros/latex/contrib/hyperxmp/}}
\begin{document}
\maketitle
A profound formal difference exists between the theoretical
concepts that physicists have formed about gases and other
ponderable bodies, and Maxwell's theory of electromagnetic
processes in so-called empty space\dots
\end{document}
```

Compile the document to PDF using any of the following approaches:

- pdf \LaTeX
- Lua \LaTeX
- \LaTeX + Dvipdfm
- \LaTeX + Dvips + Adobe Acrobat Distiller
- X \LaTeX

Unfortunately, the \LaTeX + Dvips + Ghostscript path doesn't work. Ghostscript bug report #690066, closed with "WONTFIX" status on 2012-05-28, explains that Ghostscript doesn't honor the Metadata tag needed to inject a custom XMP packet. Instead, Ghostscript fabricates an XMP packet of its own based on the metadata it finds in the PDF file's Info dictionary (Author, Title, Subject, and Keywords).

Once the document is compiled, the resulting PDF file will contain an XMP packet that looks something like this:

```
<?xpacket begin="\357\273\277" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="3.1-702">
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    <rdf:Description rdf:about=""
      xmlns:pdf="http://ns.adobe.com/pdf/1.3/"
      <pdf:Keywords>
        energy quanta, Hertz effect, quantum physics
      </pdf:Keywords>
      <pdf:Producer>pdfTeX-1.40.10</pdf:Producer>
    </rdf:Description>
    <rdf:Description rdf:about=""
      xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/"
      <xmpRights:Marked>True</xmpRights:Marked>
      <xmpRights:WebStatement>
        http://creativecommons.org/licenses/by-nc-nd/3.0/
      </xmpRights:WebStatement>
    </rdf:Description>
    <rdf:Description rdf:about=""
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      <dc:format>application/pdf</dc:format>
      <dc:title>
        <rdf:Alt>
          <rdf:li xml:lang="en">
            On a heuristic viewpoint concerning the production and
            transformation of light
          </rdf:li>
          <rdf:li xml:lang="x-default">
            On a heuristic viewpoint concerning the production and
            transformation of light
          </rdf:li>
        </rdf:Alt>
      </dc:title>
    </rdf:Description>
  </rdf:RDF>
</x:xmpmeta>
```

```

</dc:title>
<dc:description>
  <rdf:Alt>
    <rdf:li xml:lang="en">photoelectric effect</rdf:li>
    <rdf:li xml:lang="x-default">photoelectric effect</rdf:li>
  </rdf:Alt>
</dc:description>
<dc:rights>
  <rdf:Alt>
    <rdf:li xml:lang="en">
      Copyright (C) 1905, Albert Einstein
    </rdf:li>
    <rdf:li xml:lang="x-default">
      Copyright (C) 1905, Albert Einstein
    </rdf:li>
  </rdf:Alt>
</dc:rights>
<dc:creator>
  <rdf:Seq>
    <rdf:li>Albert Einstein</rdf:li>
  </rdf:Seq>
</dc:creator>
<dc:subject>
  <rdf:Bag>
    <rdf:li>energy quanta</rdf:li>
    <rdf:li>Hertz effect</rdf:li>
    <rdf:li>quantum physics</rdf:li>
  </rdf:Bag>
</dc:subject>
<dc:date>
  <rdf:Seq>
    <rdf:li>2012-08-26</rdf:li>
  </rdf:Seq>
</dc:date>
<dc:language>en</dc:language>
<dc:source>einstein.tex</dc:source>
</rdf:Description>
<rdf:Description rdf:about=""
  xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/">
  <photoshop:AuthorsPosition>
    Technical Assistant, Level III
  </photoshop:AuthorsPosition>
  <photoshop:CaptionWriter>Scott Pakin</photoshop:CaptionWriter>
</rdf:Description>
<rdf:Description rdf:about=""
  xmlns:xmp="http://ns.adobe.com/xap/1.0/">
  <xmp:CreateDate>2012-08-26</xmp:CreateDate>
  <xmp:ModifyDate>2012-08-26</xmp:ModifyDate>
  <xmp:MetadataDate>2012-08-26</xmp:MetadataDate>
  <xmp:CreatorTool>LaTeX with hyperref package</xmp:CreatorTool>

```

```

<xmp:BaseUrl>
  http://www.ctan.org/tex-archive/macros/latex/contrib/hyperxmp/
</xmp:BaseUrl>
</rdf:Description>
<rdf:Description rdf:about=""
  xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"
  <xmpMM:DocumentID>uuid:0595fdce-41dc-e4c4-6c418dc4ce46</xmpMM:DocumentID>
  <xmpMM:InstanceID>uuid:efd754c4-1d7f-200a-ef754ce413ea</xmpMM:InstanceID>
</rdf:Description>
</rdf:RDF>
</x:xmpmeta>
<?xpacket end="w"?>

```

Figure 1 is a screenshot of the preceding packet as it appears in Adobe Acrobat’s “Advanced” metadata dialog box.

Note 1: Acrobat Author bug A bug in Adobe Acrobat—at least in versions 10.0.1 and earlier—causes that PDF reader to confuse the XMP and non-XMP author lists when displaying the document’s metadata. Specifically, the first author is displayed as the concatenated list of authors from the non-XMP data (`Author`) while the remaining authors are displayed from the XMP data (`dc:creator`). For example, suppose that a document’s authors are Jack Napier, Edward Nigma, and Harvey Dent. When displaying the document properties, Adobe Acrobat replaces “Jack Napier” with a single author named “Jack Napier, Edward Nigma, Harvey Dent” and leaves “Edward Nigma” and “Harvey Dent” as the second and third authors, respectively.

`\XMPTruncateList` The `hyperxmp` package provides a workaround for this bug in the form of the `\XMPTruncateList` macro. `\XMPTruncateList` takes the name of a list (a `hyperref` option name) and replaces the list with the value of its first element. Currently, the only meaningful usage is to put

```
\XMPTruncateList{pdfauthor}
```

in your document’s preamble. This will cause Adobe Acrobat to properly display all of the authors but at the cost of other PDF readers likely displaying only the first author.

Note 2: \LaTeX object compression \LaTeX (or, more precisely, the `xdvipdfmx` back end), compresses *all* PDF objects, including the ones containing XMP metadata. While Adobe Acrobat can still detect and utilize the XMP metadata, non-PDF-aware applications are unlikely to see the metadata. Three options to consider are to (1) use a different program (e.g., `Lua \LaTeX`), (2) pass the `--output-driver="xdvipdfmx -z0"` option to \LaTeX to instruct `xdvipdfmx` to turn off all compression (which will of course make the PDF file substantially larger), or (3) postprocess the generated PDF file by loading it into the commercial version of Adobe Acrobat and re-saving it with the Save As... menu option.

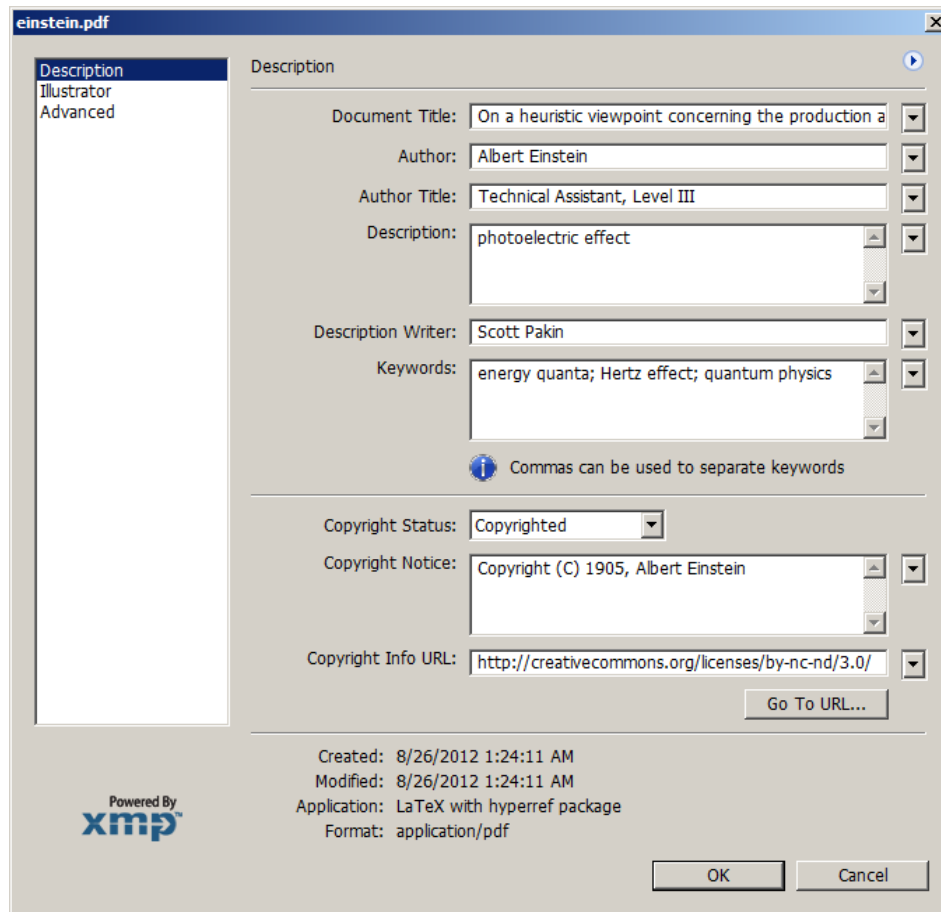


Figure 1: XMP metadata as it appears in Adobe Acrobat

Note 3: Literal commas hyperxmp splits the `pdfauthor` and `pdfkeywords` lists at commas. Therefore, when specifying `pdfauthor` and `pdfkeywords`, you should separate items with commas. Also, omit “and” and other text that does not belong to any list item. The following examples should serve as clarification:

Wrong: `pdfauthor={Jack Napier, Edward Nigma, and Harvey Dent}`

Wrong: `pdfauthor={Jack Napier; Edward Nigma; Harvey Dent}`

Right: `pdfauthor={Jack Napier, Edward Nigma, Harvey Dent}`

`\xmpcomma` If you need to include a comma within an author or keyword list, use the `\xmpcomma` macro to represent it, and wrap the entire entry containing the comma within `\xmpquote{...}` as shown below:


```
pdfauthor={\xmpquote{Jack Napier\xmpcomma\ Jr.},
\xmpquote{Edward Nigma\xmpcomma\ PhD},
\xmpquote{Harvey Dent\xmpcomma\ Esq.}}
```

3 Implementation

This section presents the commented L^AT_EX source code for `hyperxmp`. Read this section only if you want to learn how `hyperxmp` is implemented.

3.1 Initial preparation

`\hyxmp@dq@code` The `ngerman` package redefines “ ” as an active character, which causes problems for `hyperxmp` when it tries to use that character. We therefore save the double-quote character’s current category code in `\hyxmp@dq@code` and mark the character as category code 12 (“other”). The original category code is restored at the end of the package code (Section 3.7).

```
1 \edef\hyxmp@dq@code{\the\catcode'\"}
2 \catcode'\ "=12
```

`\hyxmp@at@end` The `\hyxmp@at@end` macro includes code at the end of the document. For pdfT_EX, the standard `\AtEndDocument` works well enough. For all the other backends we use `\AtEndDvi` from the `atenddvi` package, which is more robust but requires an additional L^AT_EX run.

`\hyxmp@driver`

```
3 \def\hyxmp@driver{hpdfTeX}
4 \ifx\hyxmp@driver\Hy@driver
5   \let\hyxmp@at@end=\AtEndDocument
6 \else
7   \RequirePackage{atenddvi}
8   \let\hyxmp@at@end=\AtEndDvi
9 \fi
```

3.2 Integration with hyperref

An important design decision underlying `hyperxmp` is that the package should integrate seamlessly with `hyperref`. To that end, `hyperxmp` takes its XMP metadata from `hyperref`’s `pdftitle`, `pdfauthor`, `pdfsubject`, `pdfkeywords`, and `pdflang` options. It also introduces five new options: `pdfcopyright`, `pdflicenseurl`, `pdfauthortitle`, `pdfcaptionwriter`, and `pdfmetalang`. For consistency with `hyperref`’s document-metadata naming conventions (which are in turn based on L^AT_EX’s document-metadata naming conventions), we do not prefix metadata-related macro names with our package-specific `\hyxmp@` prefix. That is, we use names like `\@pdfcopyright` instead of `\hyxmp@pdfcopyright`.

We load a bunch of helper packages: `kvoptions` for package-option processing, `pdfescape` and `stringenc` for re-encoding Unicode strings, `intcalc` for performing integer calculations (division and modulo), and `ifxetex` for detecting Xe_{La}TeX.

```
10 \RequirePackage{kvoptions}
11 \RequirePackage{pdfescape}
12 \RequirePackage{stringenc}
13 \RequirePackage{intcalc}
14 \RequirePackage{ifxetex}
```

`\@pdfcopyright` Prepare to store the document's copyright statement.

```
15 \def\@pdfcopyright{}
16 \define@key{Hyp}{pdfcopyright}{\pdfstringdef\@pdfcopyright{#1}}
```

`\@pdflicenseurl` Prepare to store the URL containing the document's license agreement.

```
17 \def\@pdflicenseurl{}
18 \define@key{Hyp}{pdflicenseurl}{\pdfstringdef\@pdflicenseurl{#1}}
```

`\@pdfauthortitle` Prepare to store the author's position/title (e.g., Staff Writer).

```
19 \def\@pdfauthortitle{}
20 \define@key{Hyp}{pdfauthortitle}{\pdfstringdef\@pdfauthortitle{#1}}
```

`\@pdfcaptionwriter` Prepare to store the name of the person who inserted the `hyperxmp` metadata.

```
21 \def\@pdfcaptionwriter{}
22 \define@key{Hyp}{pdfcaptionwriter}{\pdfstringdef\@pdfcaptionwriter{#1}}
```

`\@pdfmetalang` Prepare to store the natural language of the document's metadata, typically as an ISO 639-1 two-letter abbreviation.

```
23 \def\@pdfmetalang{}
24 \define@key{Hyp}{pdfmetalang}{\pdfstringdef\@pdfmetalang{#1}}
```

We need to capture list arguments (viz. `pdfauthor` and `pdfkeywords`) before `hyperref` converts them to PDFDocEncoding. Otherwise, `\xmpcomma` is permanently replaced with a comma, and we lose our ability to change it to a `\hyxmp@comma`. We therefore need to augment `hyperref`'s option processing with our own. Because `hyperref` has not yet been loaded we need to ensure that our augmentation gets loaded in the future: after the `\usepackage{hyperref}` but before options are passed to that package.

For lack of a better approach, `hyperxmp` redefines `\ProcessKeyvalOptions` to alter the way `hyperref` processes `pdfauthor` and `pdfkeywords`. This is somewhat heavy-handed as it gets executed for *every* subsequently loaded package that uses `\ProcessKeyvalOptions`, but at least it does what we need. `hyperxmp` also redefines `\hypersetup` to do the same thing. This is required in case `hyperref` is loaded before `hyperxmp`.

`\hyxmp@redefine@Hyp` If not already redefined, redefine `hyperref`'s `pdfauthor` and `pdfkeywords` options to properly handle `\xmpcomma` and `\xmpquote`.

```
25 \newcommand*{\hyxmp@redefine@Hyp}{%
```

`\hyxmp@Hyp@pdfauthor` Store the old definition of `\KV@Hyp@pdfauthor` in `\hyxmp@Hyp@pdfauthor`, but only if we see that `\KV@Hyp@pdfauthor` is defined and `\hyxmp@Hyp@pdfauthor` isn't. Otherwise, we'd be defining `\hyxmp@Hyp@pdfauthor` in terms of itself and creating an infinite loop.

```

26 \ifundefined{KV@Hyp@pdfauthor}{}{%
27   \ifundefined{hyxmp@Hyp@pdfauthor}{%
28     \expandafter\let\expandafter\hyxmp@Hyp@pdfauthor
29       \csname KV@Hyp@pdfauthor\endcsname
30   }{}%
31 }%
```

`\KV@Hyp@pdfauthor` Redefine `\KV@Hyp@pdfauthor` to process its argument twice. The first time, `\xmpcomma` is defined as a placeholder character (`\hyxmp@comma`) and `\xmpquote` as the identity function. The result is stored in `\hyxmp@pdfauthor` for use in structured lists (those surrounding each entry with `<rdf:li>`). The second time, `\hyxmp@pdfauthor` `\xmpcomma` is defined as an ordinary comma, and `\xmpquote` is defined as a macro that puts its argument within double quotes. The result is stored in `\@pdfauthor` for use in unstructured lists (those in which the entire list appears within a single pair of tags).

```

32 \define@key{Hyp}{pdfauthor}{%
33   \let\xmpcomma=\hyxmp@comma
34   \def\xmpquote####1{####1}%
35   \hyxmp@Hyp@pdfauthor{##1}%
36   \global\let\hyxmp@pdfauthor=\@pdfauthor
37   \def\xmpcomma{,%}
38   \def\xmpquote####1{"####1"%
39   \hyxmp@Hyp@pdfauthor{##1}%
40   \let\xmpcomma=\relax
41   \let\xmpquote=\relax
42 }%
```

`\hyxmp@Hyp@pdfkeywords` The previous block of code now repeats for the keyword list, starting by storing the old definition of `\KV@Hyp@pdfkeywords` in `\hyxmp@Hyp@pdfkeywords`.

```

43 \ifundefined{KV@Hyp@pdfkeywords}{}{%
44   \ifundefined{hyxmp@Hyp@pdfkeywords}{%
45     \expandafter\let\expandafter\hyxmp@Hyp@pdfkeywords
46       \csname KV@Hyp@pdfkeywords\endcsname
47   }{}%
48 }%
```

`\KV@Hyp@pdfkeywords` Redefine `\KV@Hyp@pdfkeywords` to process its argument twice. The first time, `\xmpcomma` is defined as a placeholder character (`\hyxmp@comma`) and `\xmpquote` as the identity function. The result is stored in `\hyxmp@pdfkeywords` for use in structured lists (those surrounding each entry with `<rdf:li>`). The second time, `\hyxmp@pdfkeywords` `\xmpcomma` is defined as an ordinary comma, and `\xmpquote` is defined as a macro that puts its argument within double quotes. The result is stored in `\@pdfkeywords` for use in unstructured lists (those in which the entire list appears within a single pair of tags).

```

49 \define@key{Hyp}{pdfkeywords}{%
50   \let\xmpcomma=\hyxmp@comma
51   \def\xmpquote####1{####1}%
52   \hyxmp@Hyp@pdfkeywords{##1}%
53   \global\let\hyxmp@pdfkeywords=\@pdfkeywords
54   \def\xmpcomma{,%}
55   \def\xmpquote####1{"####1"%
56   \hyxmp@Hyp@pdfkeywords{##1}%
57   \let\xmpcomma=\relax
58   \let\xmpquote=\relax
59 }%
60 }

```

`\hyxmp@ProcessKeyvalOptions` Redefine `kvoptions's \ProcessOptions` command to invoke `\hyxmp@redefine@Hyp` before performing its normal option processing.

```

61 \let\hyxmp@ProcessKeyvalOptions=\ProcessKeyvalOptions
62 \renewcommand*{\ProcessKeyvalOptions}{%
63   \hyxmp@redefine@Hyp
64   \hyxmp@ProcessKeyvalOptions
65 }

```

`\hyxmp@hypersetup` Redefine `hyperref's \hypersetup` command to invoke `\hyxmp@redefine@Hyp` before performing its normal option processing.

```

66 \let\hyxmp@hypersetup=\hypersetup
67 \def\hypersetup{%
68   \hyxmp@redefine@Hyp
69   \hyxmp@hypersetup
70 }

```

`\hyxmp@find@metadata` Issue a warning message if the author failed to include any metadata at all. Note that we don't consider `\pdfmetlang` as metadata as that value is meaningful only when used in conjunction with other information.

```

71 \newcommand*{\hyxmp@find@metadata}{%
72   \edef\hyxmp@concat@metadata{%
73     \@baseurl
74     \@pdfauthor
75     \@pdfauthortitle
76     \@pdfcaptionwriter
77     \@pdfcopyright
78     \@pdfkeywords
79     \@pdflang
80     \@pdflicenseurl
81     \@pdfsubject
82     \@pdftitle
83   }%
84   \ifx\hyxmp@concat@metadata\@empty
85     \PackageWarningNoLine{hyperxmp}{%
86 \jobname.tex did not specify any metadata to\MessageBreak
87 include in the XMP packet.\space\space Please see the hyperxmp\MessageBreak

```

```

88 documentation for instructions on how to provide\MessageBreak
89 metadata values to hyperxmp}%
90 \fi
91 }

```

Rather than load `hyperref` ourselves we let the author do it then verify he actually did. This approach gives the author the flexibility to load `hyperxmp` and `hyperref` in either order and to call `\hypersetup` anywhere in the document's preamble, not just before `hyperxmp` is loaded.

```

92 \AtBeginDocument{%
93   \ifpackageloaded{hyperref}{%

```

If the user explicitly specified the language to use for the document's metadata, we use that. If not, we use the document language, specified to `hyperref` with the `pdflang` option. If the author did not specify a language, we use `x-default` as the metadata language.

```

94     \ifx\@pdflang\@empty
95       \let\@pdfmetalang=\hyxmp@x@default
96     \else
97       \edef\@pdfmetalang{\@pdflang}%
98     \fi
99     \hyxmp@xmlify\@pdfmetalang

```

We wait until the end of the document to construct the XMP packet and write it to the PDF document catalog. This gives the author ample opportunity to provide metadata to `hyperref` and thereby `hyperxmp`.

```

100     \hyxmp@at@end{%
101       \hyxmp@find@metadata
102       \hyxmp@embed@packet
103     }%
104   }%
105   {\PackageWarningNoLine{hyperxmp}{%
106 \jobname.tex failed to include a\MessageBreak
107 \string\usepackage\string{hyperref\string}
108 in the preamble.\MessageBreak
109 Consequently, all hyperxmp functionality will be\MessageBreak
110 disabled}%
111   }%
112 }

```

3.3 Manipulating author-supplied data

The author provides metadata information to `hyperxmp` via package options to `hyperref` or via `hyperref`'s `\hypersetup` command. The functions in this section convert author-supplied lists (e.g., `pdfkeywords={foo, bar, baz}`) into L^AT_EX lists (e.g., `\@elt {foo} \@elt {bar} \@elt {baz}`) that can be more easily manipulated (Section 3.3.1); trim spaces off the ends of strings (Section 3.3.2); and, in Section 3.3.3, convert text to XML (e.g., from `<scott+hyxmp@pakin.org>` to `<scott+hyxmp@pakin.org>`).

3.3.1 List manipulation

We define a macro for converting a list of comma-separated elements (e.g., the list of PDF keywords) to a list of L^AT_EX \@elt-separated elements.

```
\hyxmp@commas@to@list  Given a macro name (#1) and a comma-separated list (#2), define the macro name
                        as the elements of the list, each preceded by \@elt. (Executing the macro therefore
                        applies \@elt to each element in turn.)
113 \newcommand*{\hyxmp@commas@to@list}[2]{%
114   \gdef#1{%
115     \expandafter\hyxmp@commas@to@list@i\expandafter#1#2,,%
116   }

\hyxmp@commas@to@list@i  Recursively construct macro #1 from comma-separated list #2. Stop if #2 is empty.
\next
117 \def\hyxmp@commas@to@list@i#1#2,{%
118   \gdef\hyxmp@sublist{#2}%
119   \ifx\hyxmp@sublist\@empty
120     \let\next=\relax
121   \else
122     \hyxmp@trimspaces\hyxmp@sublist
123     \@cons{#1}{\hyxmp@sublist}%
124     \def\next{\hyxmp@commas@to@list@i{#1}}%
125   \fi
126   \next
127 }

\xmpcomma  Because hyperxmp splits lists at commas, a comma cannot normally be used within
           a list. We there provide an \xmpcomma macro that can expand to either a true
           comma or a placeholder character depending on the situation. Here, we bind it to
           \relax to prevent it from expanding to either prematurely.
128 \let\xmpcomma=\relax

\hyxmp@comma  This is what \xmpcomma maps to during list construction. We assume that doc-
              uments will never otherwise use an ETX (^C) character in their XMP metadata.

129 \bgroup
130 \catcode'\^C=11
131 \gdef\hyxmp@comma{^C}
132 \egroup

\xmpquote  Adobe Acrobat likes to see double quotes around list elements that contain commas
           when the entire list appears within a single XMP tag (e.g., <pdf:Keywords>).
           However, it doesn't like to see double quotes around list elements that contain
           commas when the list is broken up into individual components (i.e., using <rdf:li>
           tags). We therefore introduce an \xmpquote macro that quotes or doesn't quote
           its argument based on context. Here, we bind \xmpquote to \relax to prevent it
           from prematurely quoting or not quoting.
133 \let\xmpquote=\relax
```

`\XMPTruncateList` As a workaround for Adobe Acrobat’s inability to display author lists correctly
`\hyxmp@temp@str` (cf. “Acrobat Author bug” on page 7) we introduce a hack that replaces a list with
`\hyxmp@temp@list` its first element. One can then write “`\XMPTruncateList{pdfauthor}`” and have
`\@elt` Adobe Acrobat display the author list correctly. It’s sad that this is necessary, though.

```

134 \newcommand{\XMPTruncateList}[1]{%
135   \edef\hyxmp@temp@str{\csname hyxmp@#1\endcsname}%
136   \hyxmp@commas@to@list{\hyxmp@temp@list}{\hyxmp@temp@str}%
137   \def\@elt##1{%
138     \expandafter\gdef\csname @#1\endcsname{##1}%
139     \let\@elt=\@gobble
140   }
141   \hyxmp@temp@list
142 }
```

3.3.2 Trimming leading and trailing spaces

To make it easier for XMP processors to manipulate our output we define a `\hyxmp@trimspaces` macro to strip leading and trailing spaces from various data fields.

`\hyxmp@trimspaces` Redefine a macro as its previous value but without leading or trailing spaces. This code—as well as that for its helper macros, `\hyxmp@trimb` and `\hyxmp@trimc`—was taken almost verbatim from a solution to an *Around the Bend* puzzle [5]. Inline comments are also taken from the solution text.

```

143 \catcode'\Q=3

\hyxmp@trimspaces\x redefines \x to have the same replacement text sans leading
and trailing space tokens.
144 \newcommand{\hyxmp@trimspaces}[1]{%
  Use grouping to emulate a multi-token afterassignment queue.
145   \begingroup
  Put “\toks 0 {” into the afterassignment queue.
146   \aftergroup\toks\aftergroup0\aftergroup{%
  Apply \hyxmp@trimb to the replacement text of #1, adding a leading \noexpand
  to prevent brace stripping and to serve another purpose later.
147   \expandafter\hyxmp@trimb\expandafter\noexpand#1Q Q}%
  Transfer the trimmed text back into #1.
148   \edef#1{\the\toks0}%
149 }
```

`\hyxmp@trimb` `\hyxmp@trimb` removes a trailing space if present, then calls `\hyxmp@trimc` to clean up any leftover bizarre Qs, and trim a leading space. In order for `\hyxmp@trimc` to work properly we need to put back a Q first.

```

150 \def\hyxmp@trimb#1 Q{\hyxmp@trimc#1Q}
```

`\hyxmp@trimc` Execute `\vfuzz` assignment to remove leading space; the `\noexpand` will now prevent unwanted expansion of a macro or other expandable token at the beginning of the trimmed text. The `\endgroup` will feed in the `\aftergroup` tokens after the `\vfuzz` assignment is completed.

```
151 \def\hyxmp@trimc#1Q#2{\afterassignment\endgroup \vfuzz\the\vfuzz#1}
152 \catcode'\Q=11
```

3.3.3 Converting text to XML

The “<”, “>”, and “&” characters are significant to XML. We therefore need to escape them in any author-supplied text.

`\ifhyxmp@unicodetex` \TeX and \LuaTeX natively support Unicode. We define the conditional `\ifhyxmp@unicodetex` to check for these so we can properly handle encoding conversions. The trick here is that Unicode \TeX implementations compare decimal 64 to hexadecimal 40 (decimal 64), specified with four carets, and take the TRUE branch; non-Unicode \TeX implementations compare decimal 64 to character “^” (decimal 94), ignore the “^^0040” and the rest of the TRUE branch, and take the FALSE branch.

```
153 \newif\ifhyxmp@unicodetex
154 \ifnum64='^^^^0040\relax
155   \hyxmp@unicodetextrue
156 \else
157   \hyxmp@unicodetexfalse
158 \fi
```

`\hyxmp@reencode` This is now a placeholder macro needed only for `\@pdfmetalang` in the `\begin{document}`.

```
159 \newcommand*{\hyxmp@reencode}[1]{}
```

`\SE->pdfdoc@03` Preserve ETX (`^^C`), which is normally an invalid character in PDFDocEncoding. We use it in `hyperxmp` (and specifically in `\hyxmp@xmlify` below) as a list-element separator.

```
160 \expandafter\def\csname SE->pdfdoc@03\endcsname{0003}
```

`\hyxmp@xmlify` Given a piece of text defined using `\pdfstringdef` (i.e., with many special characters redefined to have category code 11), set `\hyxmp@xmlified` to the same text `\hyxmp@text` but with all occurrences of “<” replaced with `<`; all occurrences of “>” replaced with `>`; and all occurrences of “&” replaced with `&`.

```
161 \newcommand*{\hyxmp@xmlify}[1]{%
162   \gdef\hyxmp@xmlified{}
```

Escaped PDF string \rightarrow PDFDocEncoding/Unicode

```
163   \EdefUnescapeString\hyxmp@text{#1}%
164   \ifhyxmp@unicodetex
```

PDFDocEncoding/Unicode \rightarrow UTF-32BE

```
165     \hyxmp@is@unicode\hyxmp@text{%
```



```

166     \StringEncodingConvert
167     \hyxmp@text\hyxmp@text{utf16be}{utf32be}%
168 }{%
169     \ifxetex
170     \hyxmp@xetex@crap
171     \else
172     \StringEncodingConvert
173     \hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
174     \fi
175 }%

UTF-32BE → UTF-32BE as hex string
176     \EdefEscapeHex\hyxmp@text{\hyxmp@text}%

UTF-32BE → XML in ASCII
177     \edef\hyxmp@text{%
178     \expandafter
179     }\expandafter\hyxmp@toxml@unicodetex\hyxmp@text
180     \relax\relax\relax\relax\relax\relax\relax\relax
181     \else

PDFDocEncoding/Unicode → UTF-8
182     \hyxmp@is@unicode\hyxmp@text{%
183     \StringEncodingConvert
184     \hyxmp@text\hyxmp@text{utf16be}{utf8}%
185     }{%
186     \StringEncodingConvert
187     \hyxmp@text\hyxmp@text{pdfdoc}{utf8}%
188     }%

UTF-8 → UTF-8 as hex string
189     \EdefEscapeHex\hyxmp@text{\hyxmp@text}%

UTF-8 as hex string → XML in UTF-8 as hex string
190     \edef\hyxmp@text{%
191     \expandafter\hyxmp@toxml\hyxmp@text\@empty\@empty
192     }%

XML in UTF-8 as hex string → XML in UTF-8
193     \EdefUnescapeHex\hyxmp@text{\hyxmp@text}%
194     \fi
195     \global\let\hyxmp@xmlified\hyxmp@text
196 }

```

`\hyxmp@is@unicode` Given a string and two expressions, evaluate the first expression if the string is
`\hyxmp@@is@unicode` UTF-16BE-encoded and the second expression if not.

```

197 \begingroup
198 \lccode'\<=254 %
199 \lccode'\>=255 %
200 \catcode254=12 %
201 \catcode255=12 %
202 \lowercase{\endgroup

```

```

203 \def\hyxmp@is@unicode#1{%
204   \expandafter\hyxmp@@is@unicode#1<>\@nil
205 }%
206 \def\hyxmp@@is@unicode#1<>#2\@nil{%
207   \ifx\#1\%
208     \expandafter\@firstoftwo
209   \else
210     \expandafter\@secondoftwo
211   \fi
212 }%
213 }

```

`\hyxmp@toxml` Replace the characters “<”, “&”, and “>” with XML entities when using a non-native-Unicode T_EX (T_EX or pdfT_EX).

```

214 \def\hyxmp@toxml#1#2{%
215   \ifx#1\@empty
216   \else
217     \ifnum"#1#2='\& %
218       26616D703B% &amp;
219     \else\ifnum"#1#2='\< %
220       266C743B% &lt;
221     \else\ifnum"#1#2='\> %
222       2667743B% &gt;
223   \else

```

`dvips` wraps text when generating most PostScript code but preserves line breaks within strings. Unfortunately, `dvips` fails to observe the special case in the PostScript specification that “[b]alanced pairs of parentheses in the string require no special treatment” [2]. Consequently, XMP data containing parentheses (e.g., “Copyright (C) 1605 Miguel de Cervantes”) confuse `dvips` into thinking that the string has ended after the closing parenthesis and that line breaks can subsequently be injected safely into the document at arbitrary points for formatting purposes. This leads to erroneous display by PDF viewers, which honor line breaks within XMP tags. The solution is to insert a backslash before all parentheses when in `pdfmark`-generating mode to convince `dvips` that the entire XMP packet must be treated as a single, not-to-be-modified string.

```

224   \@ifundefined{pdfmark}{%
225     #1#2%
226   }{%
227     \ifnum"#1#2='\( %
228       5C28% \(
229     \else\ifnum"#1#2='\) %
230       5C29% \)
231     \else
232       #1#2%
233     \fi\fi
234   }%
235   \fi\fi\fi
236   \expandafter\hyxmp@toxml

```

```

237 \fi
238 }

\hyxmp@toxml@unicodetex Replace the characters "<", "&", and ">" with XML entities when using a native-
\hyxmp@text Unicode TEX (XYTEX or LuaTEX).
239 \def\hyxmp@toxml@unicodetex#1#2#3#4#5#6#7#8{%
240 \ifx#1\relax
241 \else
242 \ifnum"#1#2#3#4#5#6#7#8>127 %
243 \uccode'\*="#1#2#3#4#5#6#7#8\relax
244 \uppercase{%
245 \edef\hyxmp@text{\hyxmp@text *}%
246 }%
247 \else\ifnum"#7#8='< %
248 \edef\hyxmp@text{\hyxmp@text &lt;}%
249 \else\ifnum"#7#8='& %
250 \edef\hyxmp@text{\hyxmp@text &amp;}%
251 \else\ifnum"#7#8='> %
252 \edef\hyxmp@text{\hyxmp@text &gt;}%
253 \else\ifnum"#7#8='\ %
254 \edef\hyxmp@text{\hyxmp@text\space}%
255 \else
256 \uccode'\*="#7#8\relax
257 \uppercase{%
258 \edef\hyxmp@text{\hyxmp@text *}%
259 }%
260 \fi\fi\fi\fi\fi
261 \expandafter\hyxmp@toxml@unicodetex
262 \fi
263 }

\hyxmp@skipzeros Skip over leading zeroes in the input argument.
264 \def\hyxmp@skipzeros#1{%
265 \ifx#10%
266 \expandafter\hyxmp@skipzeros
267 \fi
268 }

\hyxmp@xetex@crap \x In the case of XYTEX, the strings defined by \pdfstringdef can contain big
\hyxmp@xetex@crap characters. In this case, the string is treated as Unicode.
\hyxmp@try 269 \begingroup
\hyxmp@crap@result 270 \def\x#1{\endgroup
\hyxmp@text 271 \def\hyxmp@xetex@crap{%
272 \edef\hyxmp@try{%
273 \expandafter\hyxmp@SpaceOther\hyxmp@text#1\@nil
274 }%
275 \let\hyxmp@crap@result=N%
276 \expandafter\hyxmp@crap@test\hyxmp@try\relax
277 \ifx\hyxmp@crap@result Y%

```

```

278     \let\hyxmp@text\@empty
279     \expandafter\hyxmp@crap@convert\hyxmp@try\relax
280   \else
281     \StringEncodingConvert\hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
282   \fi
283 }%
284 }
285 \x{ }

```

`\hyxmp@SpaceOther` Re-encode all spaces in a string with category code 12 (“other”).

```

286 \begingroup
287   \catcode'\~=12 %
288   \lccode'\~= '\ %
289 \lowercase{\endgroup
290   \def\hyxmp@SpaceOther#1 #2\@nil{%
291     #1%
292     \ifx\relax#2\relax
293       \expandafter\@gobble
294     \else
295       ~%
296       \expandafter\@firstofone
297     \fi
298     {\hyxmp@SpaceOther#2\@nil}%
299   }%
300 }

```

`\hyxmp@crap@test` Determine if we need to treat a string as Unicode.

```

301 \def\hyxmp@crap@test#1{%
302   \ifx#1\relax
303   \else
304     \ifnum'#1>127 %
305       \let\hyxmp@crap@result=Y%
306       \expandafter\expandafter\expandafter\hyxmp@skiptorelax
307     \else
308       \expandafter\expandafter\expandafter\hyxmp@crap@test
309     \fi
310   \fi
311 }

```

`\hyxmp@skiptorelax` Discard all tokens up to and including the first `\relax`.

```

312 \def\hyxmp@skiptorelax#1\relax{}

```

`\hyxmp@crap@convert` Convert a hexadecimal string to a number.

```

\hyxmp@num 313 \def\hyxmp@crap@convert#1{%
\hyxmp@text 314   \ifx#1\relax
315   \else
316     \edef\hyxmp@num{\number'#1}%
317     \ifnum\hyxmp@num>"FFFFFF %
318       \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"1000000}\relax

```

```

319     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}}%
320     \edef\hyxmp@num{\intcalMod{\hyxmp@num}{\number"1000000}}}%
321   \else
322     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
323   \fi
324   \ifnum\hyxmp@num>"FFFF %
325     \lccode'\!=\intcalDiv{\hyxmp@num}{\number"10000}\relax
326     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}}%
327     \edef\hyxmp@num{\intcalMod{\hyxmp@num}{\number"10000}}}%
328   \else
329     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
330   \fi
331   \ifnum\hyxmp@num>"FF %
332     \lccode'\!=\intcalDiv{\hyxmp@num}{\number"100}\relax
333     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}}%
334     \edef\hyxmp@num{\intcalMod{\hyxmp@num}{\number"100}}}%
335   \else
336     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
337   \fi
338   \ifnum\hyxmp@num>0 %
339     \lccode'\!=\hyxmp@num\relax
340     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}}%
341   \else
342     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
343   \fi
344   \expandafter\hyxmp@crap@convert
345 \fi
346 }

```

`\hyxmp@zero` Define a null character with category code 12 (“other”).

```

347 \begingroup
348   \catcode0=12 %
349   \gdef\hyxmp@zero{^^00}%
350 \endgroup

```

3.4 UUID generation

We use a linear congruential generator to produce pseudorandom UUIDs. True, this method has its flaws but it’s simple to implement in T_EX and is good enough for producing the XMP xmpMM:DocumentID and xmpMM:InstanceID fields.

`\hyxmp@modulo@a` Replace the contents of `\@tempcnta` with the contents modulo #1. Note that `\@tempcntb` is overwritten in the process.

```

351 \def\hyxmp@modulo@a#1{%
352   \@tempcntb=\@tempcnta
353   \divide\@tempcntb by #1
354   \multiply\@tempcntb by #1
355   \advance\@tempcnta by -\@tempcntb
356 }

```

`\hyxmp@big@prime` Define a couple of large prime numbers that can still be stored in a \TeX counter.

```

\hyxmp@big@prime@ii 357 \def\hyxmp@big@prime{536870923}
358 \def\hyxmp@big@prime@ii{536870027}

```

`\hyxmp@seed@rng` Seed hyperxmp's random-number generator from a given piece of text.

```

\hyxmp@one@token 359 \def\hyxmp@seed@rng#1{%
360   \@tempcnta=\hyxmp@big@prime
361   \futurelet\hyxmp@one@token\hyxmp@seed@rng@i#1\@empty
362 }

```

`\hyxmp@seed@rng@i` Do all of the work for `\hyxmp@seed@rng`. For each character code c of the input text, assign $\text{\@tempcnta} \leftarrow 3 \cdot \text{\@tempcnta} + c \pmod{\text{\hyxmp@big@prime}}$.

```

\hyxmp@one@token 363 \def\hyxmp@seed@rng@i{%
364   \ifx\hyxmp@one@token\@empty
365     \let\next=\relax
366   \else
367     \def\next##1{%
368       \multiply\@tempcnta by 3
369       \advance\@tempcnta by '##1
370       \hyxmp@modulo@a{\hyxmp@big@prime}%
371       \futurelet\hyxmp@one@token\hyxmp@seed@rng@i
372     }%
373   \fi
374   \next
375 }

```

`\hyxmp@set@rand@num` Advance `\hyxmp@rand@num` to the next pseudorandom number in the sequence. Specifically, we assign $\text{\hyxmp@rand@num} \leftarrow 3 \cdot \text{\hyxmp@rand@num} + \text{\hyxmp@big@prime@ii} \pmod{\text{\hyxmp@big@prime}}$. Note that both `\@tempcnta` and `\@tempcntb` are overwritten in the process.

```

\hyxmp@rand@num 376 \def\hyxmp@set@rand@num{%
377   \@tempcnta=\hyxmp@rand@num
378   \multiply\@tempcnta by 3
379   \advance\@tempcnta by \hyxmp@big@prime@ii
380   \hyxmp@modulo@a{\hyxmp@big@prime}%
381   \xdef\hyxmp@rand@num{\the\@tempcnta}%
382 }

```

`\hyxmp@append@hex` Append a randomly selected hexadecimal digit to macro #1. Note that both `\@tempcnta` and `\@tempcntb` are overwritten in the process.

```

383 \def\hyxmp@append@hex#1{%
384   \hyxmp@set@rand@num
385   \@tempcnta=\hyxmp@rand@num
386   \hyxmp@modulo@a{16}%
387   \ifnum\@tempcnta<10
388     \xdef#1{#1\the\@tempcnta}%
389   \else

```

There *must* be a better way to handle the numbers 10–15 than with `\ifcase`.

```

390     \advance\@tempcnta by -10
391     \ifcase\@tempcnta
392         \xdef#1{#1a}%
393         \or\xdef#1{#1b}%
394         \or\xdef#1{#1c}%
395         \or\xdef#1{#1d}%
396         \or\xdef#1{#1e}%
397         \or\xdef#1{#1f}%
398     \fi
399 \fi
400 }
```

`\hyxmp@append@hex@iv` Invoke `\hyxmp@append@hex` four times.

```

401 \def\hyxmp@append@hex@iv#1{%
402     \hyxmp@append@hex#1%
403     \hyxmp@append@hex#1%
404     \hyxmp@append@hex#1%
405     \hyxmp@append@hex#1%
406 }
```

`\hyxmp@create@uuid` Define macro `#1` as a UUID of the form “`uuid:xxxxxxxx-xxxx-xxxx-xxxxxxxxxxxx`” in which each “`x`” is a lowercase hexadecimal digit. We assume that the random-number generator is already seeded. Note that `\hyxmp@create@uuid` overwrites both `\@tempcnta` and `\@tempcntb`.

```

407 \def\hyxmp@create@uuid#1{%
408     \def#1{uuid:}%
409     \hyxmp@append@hex@iv#1%
410     \hyxmp@append@hex@iv#1%
411     \g@addto@macro#1{-}%
412     \hyxmp@append@hex@iv#1%
413     \g@addto@macro#1{-}%
414     \hyxmp@append@hex@iv#1%
415     \g@addto@macro#1{-}%
416     \hyxmp@append@hex@iv#1%
417     \hyxmp@append@hex@iv#1%
418     \hyxmp@append@hex@iv#1%
419 }
```

`\hyxmp@def@DocumentID` Seed the random-number generator with a function of the current filename, PDF document title, and PDF author, then invoke `\hyxmp@create@uuid` to define `\hyxmp@DocumentID` as a random UUID.

```

420 \newcommand*{\hyxmp@def@DocumentID}{%
421     \edef\hyxmp@seed@string{\jobname:\@pdftitle:\@pdfauthor}%
422     \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
423     \edef\hyxmp@rand@num{\the\@tempcnta}%
424     \hyxmp@create@uuid\hyxmp@DocumentID
425 }
```

`\hyxmp@def@InstanceID` Seed the random-number generator with a function of the current filename, PDF document title, PDF author, and the current day, month, year, and minutes since midnight, then invoke `\hyxmp@create@uuid` to define `\hyxmp@InstanceID` as a random UUID.

```

426 \newcommand*{\hyxmp@def@InstanceID}{%
427   \edef\hyxmp@seed@string{%
428     \jobname:\@pdftitle:\@pdfauthor:%
429     \the\year/\the\month/\the\day:%
430     \the\time
431   }%
432   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
433   \edef\hyxmp@rand@num{\the\@tempcnta}%
434   \hyxmp@create@uuid\hyxmp@InstanceID
435 }
```

3.5 Constructing the XMP packet

An XMP packet “shall consist of the following, in order: a header PI, the serialized XMP data model (the XMP packet) with optional white-space padding, and a trailer PI” [4]. (“PI” is an abbreviation for “processing instructions”). The serialized XMP includes blocks of XML for various XMP schemata: Adobe PDF (Section 3.5.2), Dublin Core (Section 3.5.3), XMP Rights Management (Section 3.5.4), and XMP Media Management (Section 3.5.5). The `\hyxmp@construct@packet` macro constructs the XMP packet into `\hyxmp@xml`. It first writes the appropriate XML header, then calls the various schema-writing macros, then injects `\hyxmp@padding` as padding, and finally writes the appropriate XML trailer.

3.5.1 XMP utility functions

`\hyxmp@add@to@xml` Given a piece of text, replace all underscores with category-code 11 (“other”) spaces and all `^C` characters with commas, then append the result to the `\hyxmp@xml` macro.

```

436 \newcommand*{\hyxmp@add@to@xml}[1]{%
437   \bgroup
438     \@tempcnta=0
439     \loop
440       \lccode\@tempcnta=\@tempcnta
441       \advance\@tempcnta by 1
442       \ifnum\@tempcnta<256
443         \repeat
444         \lccode'\_='\ \relax
445         \lccode'\^C='\,\relax
446         \lowercase{\xdef\hyxmp@xml{\hyxmp@xml#1}}%
447       \egroup
448 }
```

`\hyxmp@hash` Define a category-code 11 (“other”) version of the “#” character.

```

449 \bgroup
```



```

450 \catcode'\#=11
451 \gdef\hyxmp@hash{#}
452 \egroup

\hyxmp@padding The XMP specification recommends leaving approximately 2000 bytes of whites-
\hyxmp+xml    pace at the end of each XMP packet to facilitate editing the packet in place [4].
                \hyxmp@padding is defined to contain 32 lines of 50 spaces and a newline apiece
                for a total of 1632 characters of whitespace.

453 \bgroup
454 \xdef\hyxmp+xml{%
455 \hyxmp@add@to+xml{%
456 ----- ^^J%
457 }
458 \xdef\hyxmp@padding{\hyxmp+xml}%
459 \egroup
460 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
461 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
462 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
463 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
464 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}

\hyxmp@today Define today's date in YYYY-MM-DD format.

465 \xdef\hyxmp@today{\the\year}%
466 \ifnum\month<10
467 \xdef\hyxmp@today{\hyxmp@today-0\the\month}%
468 \else
469 \xdef\hyxmp@today{\hyxmp@today-\the\month}%
470 \fi
471 \ifnum\day<10
472 \xdef\hyxmp@today{\hyxmp@today-0\the\day}%
473 \else
474 \xdef\hyxmp@today{\hyxmp@today-\the\day}%
475 \fi

\hyxmp@x@default Define an x-default string that we can use in comparisons with \@pdfmetalang.
476 \newcommand*{\hyxmp@x@default}{x-default}

3.5.2 The Adobe PDF schema

\hyxmp@pdf@schema Add properties defined by the Adobe PDF schema to the \hyxmp+xml macro.
477 \newcommand*{\hyxmp@pdf@schema}{%

\hyxmp@have@any Include an Adobe PDF schema block if at least one of \@pdfkeywords and
\@pdfproducer is defined.

478 \let\hyxmp@have@any=!%
479 \ifx\@pdfkeywords\@empty
480 \ifx\@pdfproducer\@empty
481 \let\hyxmp@have@any=\@empty
482 \fi

```

```

483 \fi
484 \ifx\hyxmp@have@any\@empty
485 \else
    Add a block of XML to \hyxmp@xml that lists the document's keywords (the
    pdf:Keywords property) and the tools used to produce the PDF file (the pdf:Producer
    property).
486 \hyxmp@add@to@xml{%
487 -----<rdf:Description rdf:about=""^^J%
488 -----xmlns:pdf="http://ns.adobe.com/pdf/1.3/">^^J%
489 }%
490 \hyxmp@add@simple{pdf:Keywords}{\@pdfkeywords}%
491 \hyxmp@add@simple{pdf:Producer}{\@pdfproducer}%
492 \@ifundefined{pdfminorversion}{}{%
493 \hyxmp@add@simple{pdf:PDFVersion}{1.\the\pdfminorversion}%
494 }%
495 \hyxmp@add@to@xml{%
496 -----</rdf:Description>^^J%
497 }%
498 \fi
499 }

```

\hyxmp@add@simple Given an XMP tag (#1) and a string (#2), if the string is nonempty, add a begin tag, the string, and an end tag to the packet. The “simple” in the macro name indicates that the string is output without variations for different languages.

```

\hyxmp@string
500 \newcommand*{\hyxmp@add@simple}[2]{%
501 \edef\hyxmp@string{#2}%
502 \ifx\hyxmp@string\@empty
503 \else
504 \hyxmp@xmlify{\hyxmp@string}%
505 \hyxmp@add@to@xml{%
506 -----<#1>\hyxmp@xmlified</#1>^^J%
507 }%
508 \fi
509 }

```

3.5.3 The Dublin Core schema

\hyxmp@rdf@dc Given a Dublin Core property (#1) and a macro containing some \pdfstringdef-defined text (#2), append the appropriate block of XML to the \hyxmp@xml macro but only if #2 is non-empty.

```

510 \newcommand*{\hyxmp@rdf@dc}[2]{%
511 \ifx#2\@empty
512 \else
513 \hyxmp@xmlify{#2}%
514 \hyxmp@add@to@xml{%
515 -----<dc:#1>^^J%
516 -----<rdf:Alt>^^J%
517 }%

```

```

518 \ifx\@pdfmetalang\hyxmp@x@default
519 \else
520 \hyxmp@add@to@xml{%
521 -----<rdf:li xml:lang="\@pdfmetalang">\hyxmp@xmlified</rdf:li>^^J%
522 }%
523 \fi
524 \hyxmp@add@to@xml{%
525 -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@xmlified</rdf:li>^^J%
526 -----</rdf:Alt>^^J%
527 -----</dc:#1>^^J%
528 }%
529 \fi%
530 }%

```

`\hyxmp@list@to@xml` Given a Dublin Core property (#1), an RDF array (#2), and a macro containing a comma-separated list (#3), append the appropriate block of XML to the `\hyxmp@xml` macro but only if #3 is non-empty.

```

531 \newcommand*{\hyxmp@list@to@xml}[3]{%
532 \ifx#3\empty
533 \else
534 \hyxmp@add@to@xml{%
535 -----<dc:#1>^^J%
536 -----<rdf:#2>^^J%
537 }%
538 \bgroup

```

`\@elt` Re-encode the text from Unicode if necessary. Then redefine `\@elt` to XML-ify each element of the list and append it to `\hyxmp@xmlified`.

```

539 \hyxmp@xmlify{#3}%
540 \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
541 \def\@elt##1{%
542 \hyxmp@add@to@xml{%
543 -----<rdf:li>##1</rdf:li>^^J%
544 }%
545 }%
546 \hyxmp@list
547 \egroup
548 \hyxmp@add@to@xml{%
549 -----</rdf:#2>^^J%
550 -----</dc:#1>^^J%
551 }%
552 \fi
553 }

```

`\hyxmp@dc@schema` Add properties defined by the Dublin Core schema to the `\hyxmp@xml` macro. Specifically, we add entries for the `dc:title` property if the author specified a `pdftitle`, the `dc:description` property if the author specified a `pdfsubject`, the `dc:rights` property if the author specified a `pdfcopyright`, the `dc:creator` property if the author specified a `pdfauthor`, the `dc:subject` property if the author specified

pdfkeywords, and the dc:language property if the author specified pdflang. We also specify the dc:date property using the date the document was run through L^AT_EX and the dc:source property using the base name of the source file with .tex appended.

```

554 \newcommand*{\hyxmp@dc@schema}{%
555   \hyxmp@add@to@xml{%
556     <rdf:Description rdf:about=""^^J%
557     _____xmlns:dc="http://purl.org/dc/elements/1.1/">^^J%
558     _____<dc:format>application/pdf</dc:format>^^J%
559   }%
560   \hyxmp@rdf@dc{title}{\@pdftitle}%
561   \hyxmp@rdf@dc{description}{\@pdfsubject}%
562   \hyxmp@rdf@dc{rights}{\@pdfcopyright}%
563   \hyxmp@list@to@xml{creator}{Seq}{\hyxmp@pdfauthor}%
564   \hyxmp@list@to@xml{subject}{Bag}{\hyxmp@pdfkeywords}%
565   \hyxmp@list@to@xml{date}{Seq}{\hyxmp@today}%
566   \hyxmp@add@simple{dc:language}{\@pdflang}%
567   \hyxmp@add@simple{dc:source}{\jobname.tex}%
568   \hyxmp@add@to@xml{%
569     _____</rdf:Description>^^J%
570   }%
571 }
```

3.5.4 The XMP Rights Management schema

\hyxmp@xmpRights@schema Add properties defined by the XMP Rights Management schema to the \hyxmp@xml macro. Currently, these are only the xmpRights:Marked property and the xmpRights:WebStatement property. If the author specified a copyright statement we mark the document as copyrighted. If the author specified a license statement we include the URL in the metadata.

```

572 \newcommand*{\hyxmp@xmpRights@schema}{%
```

\hyxmp@legal Set \hyxmp@rights to YES if either pdfcopyright or pdflicenseurl was specified.

```

573   \let\hyxmp@rights=\@empty
574   \ifx\@pdflicenseurl\@empty
575   \else
576     \def\hyxmp@rights{YES}%
577   \fi
578   \ifx\@pdfcopyright\@empty
579   \else
580     \def\hyxmp@rights{YES}%
581   \fi
```

Include the license-statement URL and/or the copyright indication. The copyright statement itself is included by \hyxmp@dc@schema in Section 3.5.3.

```

582   \ifx\hyxmp@rights\@empty
583   \else
```

Header

```

584 \hyxmp@add@to@xml{%
585 -----<rdf:Description rdf:about=""^^J%
586 -----xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/">^^J%
587 }%

```

Copyright indication

```

588 \ifx\@pdfcopyright\@empty
589 \else
590 \hyxmp@add@to@xml{%
591 -----<xmpRights:Marked>True</xmpRights:Marked>^^J%
592 }%
593 \fi

```

License URL

```

594 \hyxmp@add@simple{xmpRights:WebStatement}{\@pdflicenseurl}%

```

Trailer

```

595 \hyxmp@add@to@xml{%
596 -----</rdf:Description>^^J%
597 }%
598 \fi
599 }

```

3.5.5 The XMP Media Management schema

`\hyxmp@mm@schema` Add properties defined by the XMP Media Management schema to the `\hyxmp@xml` macro. According to the XMP specification, the `xmpMM:DocumentID` property is supposed to uniquely identify a document, and the `xmpMM:InstanceID` property is supposed to change with each save operation [4]. As seen in Section 3.4, we do what we can to honor this intention from within a \TeX -based workflow.

```

600 \gdef\hyxmp@mm@schema{%
601 \hyxmp@def@DocumentID
602 \hyxmp@def@InstanceID
603 \hyxmp@add@to@xml{%
604 -----<rdf:Description rdf:about=""^^J%
605 -----xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/">^^J%
606 -----<xmpMM:DocumentID>\hyxmp@DocumentID</xmpMM:DocumentID>^^J%
607 -----<xmpMM:InstanceID>\hyxmp@InstanceID</xmpMM:InstanceID>^^J%
608 -----</rdf:Description>^^J%
609 }%
610 }

```

3.5.6 The XMP Basic schema

`\hyxmp@xmp@basic@schema` Add properties defined by the XMP Basic schema to the `\hyxmp@xml` macro. These include a bunch of dates (all set to the same value) and the base URL for the document if specified with `baseurl`.

```

611 \newcommand*{\hyxmp@xmp@basic@schema}{%
612 \hyxmp@add@to@xml{%
613 -----<rdf:Description rdf:about=""^^J%

```

```

614 -----xmlns:xmp="http://ns.adobe.com/xap/1.0/">^^J%
615  }%
616   \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@today}%
617   \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@today}%
618   \hyxmp@add@simple{xmp:MetadataDate}{\hyxmp@today}%
619   \hyxmp@add@simple{xmp:CreatorTool}{\@pdfcreator}%
620   \hyxmp@add@simple{xmp:BaseURL}{\@baseurl}%
621   \hyxmp@add@to@xml{%
622 -----</rdf:Description>^^J%
623  }%
624 }

```

3.5.7 The Photoshop schema

`\hyxmp@photoshop@schema` Add properties defined by the Photoshop schema to the `\hyxmp@xml` macro. We
`\hyxmp@photoshop@data` currently support only the `photoshop:AuthorsPosition` and `photoshop:CaptionWriter`
properties.

```

625 \gdef\hyxmp@photoshop@schema{%
626   \edef\hyxmp@photoshop@data{\@pdfauthortitle\@pdfcaptionwriter}%
627   \ifx\hyxmp@photoshop@data\@empty
628     \else
629       \hyxmp@add@to@xml{%
630 -----<rdf:Description rdf:about=""^^J%
631 -----xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/">^^J%
632   }%
633   \fi
634   \hyxmp@add@simple{photoshop:AuthorsPosition}{\@pdfauthortitle}%
635   \hyxmp@add@simple{photoshop:CaptionWriter}{\@pdfcaptionwriter}%
636   \ifx\hyxmp@photoshop@data\@empty
637     \else
638       \hyxmp@add@to@xml{%
639 -----</rdf:Description>^^J%
640   }%
641   \fi
642 }

```

3.5.8 Constructing the XMP packet

`\hyxmp@bom` Define a macro for the Unicode byte-order marker (BOM).

```

643 \begingroup
644   \ifhyxmp@unicodetex
645     \lccode'\!="FEFF %
646     \lowercase{%
647       \gdef\hyxmp@bom{!}
648     }%
649   \else
650     \catcode'\^^ef=12
651     \catcode'\^^bb=12
652     \catcode'\^^bf=12

```

```

653     \gdef\hyxmp@bom{^^ef^^bb^^bf}%
654     \fi
655 \endgroup

\hyxmp@construct@packet  Successively add XML data to \hyxmp+xml until we have something we can insert
\hyxmp+xml              into the document's PDF catalog.
656 \def\hyxmp@construct@packet{%
657     \gdef\hyxmp+xml{%
658         \hyxmp@add@to+xml{<?xpacket begin="\hyxmp@bom" %
659 id="W5MOMpCehiHzreSzNTczkc9d"?>^^J%
660 <x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="3.1-702">^^J%
661 ___<rdf:RDF
662 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\hyxmp@hash">^^J%
663 }%
664 \hyxmp@pdf@schema
665 \hyxmp@xmpRights@schema
666 \hyxmp@dc@schema
667 \hyxmp@photoshop@schema
668 \hyxmp@xmp@basic@schema
669 \hyxmp@mm@schema
670 \hyxmp@add@to+xml{%
671 ___</rdf:RDF>^^J%
672 </x:xmpmeta>^^J%
673 \hyxmp@padding
674 <?xpacket end="w"?>^^J%
675 }%
676 }

```

3.6 Embedding the XMP packet

The PDF specification says that “a metadata stream may be attached to a document through the Metadata entry in the document catalogue” [3] so that’s what we do here.

```

\hyxmp@embed@packet  Determine which hyperref driver is in use and invoke the appropriate embedding
\hyxmp@driver         function.
677 \newcommand*{\hyxmp@embed@packet}{%
678     \hyxmp@construct@packet
679     \def\hyxmp@driver{hpdfTeX}%
680     \ifx\hyxmp@driver\Hy@driver
681         \hyxmp@embed@packet@pdfTeX
682     \else
683         \def\hyxmp@driver{hdvipdfm}%
684         \ifx\hyxmp@driver\Hy@driver
685             \hyxmp@embed@packet@dviPDFm
686         \else
687             \def\hyxmp@driver{hXeTeX}%
688             \ifx\hyxmp@driver\Hy@driver
689                 \hyxmp@embed@packet@XeTeX

```

```

690     \else
691     \ifundefined{pdfmark}{%
692     \PackageWarningNoLine{hyperxmp}{%
693     Unrecognized hyperref driver ‘\Hy@driver’.\MessageBreak
694     \jobname.tex’s XMP metadata will *not* be\MessageBreak
695     embedded in the resulting file}%
696     }{%
697     \hyxmp@embed@packet@pdfmark
698     }%
699     \fi
700     \fi
701     \fi
702 }

```

3.6.1 Embedding using pdfTeX

`\hyxmp@embed@packet@pdftex` Embed the XMP packet using pdfTeX primitives.

```

703 \newcommand*{\hyxmp@embed@packet@pdftex}{%
704   \bgroup
705   \pdfcompresslevel=0
706   \immediate\pdfobj stream attr {%
707     /Type /Metadata
708     /Subtype /XML
709   }\hyxmp@xml}%
710   \pdfcatalog {/Metadata \the\pdflastobj\space 0 R}%
711   \egroup
712 }

```

3.6.2 Embedding using any pdfmark-based backend

`\hyxmp@embed@packet@pdfmark` Embed the XMP packet using hyperref’s `\pdfmark` command. I believe `\pdfmark` is used by the `dvipdf`, `dvipsone`, `dvips`, `dviwindo`, `nativepdf`, `pdfmark`, `ps2pdf`, `textures`, and `vtexpdfmark` options to `hyperref` but I’ve tested only a few of those.

```

713 \newcommand*{\hyxmp@embed@packet@pdfmark}{%
714   \pdfmark{%
715     pdfmark=/NamespacePush
716   }%
717   \pdfmark{%
718     pdfmark=/OBJ,
719     Raw={/_objdef \string\hyxmp@Metadata\string} /type /stream}%
720   }%
721   \pdfmark{%
722     pdfmark=/PUT,
723     Raw={\string\hyxmp@Metadata\string}
724     2 dict begin
725       /Type /Metadata def
726       /Subtype /XML def
727       currentdict
728     end

```



```

729     }%
730 }%
731 \pdfmark{%
732     pdfmark=/PUT,
733     Raw={\string{hyxmp@Metadata\string} (\hyxmp@xml)}}%
734 }%
735 \pdfmark{%
736     pdfmark=/Metadata,
737     Raw={\string{Catalog\string} \string{hyxmp@Metadata\string}}%
738 }%
739 \pdfmark{%
740     pdfmark=/NamespacePop
741 }%
742 }

```

3.6.3 Embedding using dvipdfm

`\hyxmp@embed@packet@dvipdfm` Embed the XMP packet using dvipdfm-specific `\special` commands. Note that dvipdfm rather irritatingly requires us to count the number of characters in the `\hyxmp@xml` stream ourselves.

```

743 \newcommand*{\hyxmp@embed@packet@dvipdfm}{%
744     \hyxmp@string@len{\hyxmp@xml}%
745     \special{pdf: object @hyxmp@Metadata
746         <<
747             /Type /Metadata
748             /Subtype /XML
749             /Length \the\@tempcnta
750         >>
751         stream^^J\hyxmp@xml endstream%
752     }%
753     \special{pdf: docview
754         <<
755             /Metadata @hyxmp@Metadata
756         >>
757     }%
758 }

```

`\hyxmp@string@len` Set `\@tempcnta` to the number of characters in a given string (`#1`). The approach is first to tally the number of space characters then to tally the number of non-space characters. While this is rather sloppy I haven't found a better way to achieve the same effect, especially given that all of the characters in `#1` have already been assigned their category codes.

```

759 \newcommand*{\hyxmp@string@len}[1]{%
760     \@tempcnta=0
761     \expandafter\hyxmp@count@spaces#1 {} %
762     \expandafter\hyxmp@count@non@spaces#1{}%
763 }

```

`\hyxmp@count@spaces` Count the number of spaces in a given string. We rely on the built-in pattern

matching of T_EX’s `\def` primitive to pry one word at a time off the head of the input string.

```

764 \def\hyxmp@count@spaces#1 {%
765   \def\hyxmp@one@token{#1}%
766   \ifx\hyxmp@one@token\empty
767     \advance\@tempcnta by -1
768   \else
769     \advance\@tempcnta by 1
770     \expandafter\hyxmp@count@spaces
771   \fi
772 }
```

`\hyxmp@count@non@spaces` Count the number of non-spaces in a given string. Ideally, we’d count both spaces and non-spaces but T_EX won’t bind `#1` to a space character (category code 10). Hence, in each iteration, `#1` is bound to the next non-space character only.

```

773 \newcommand*\hyxmp@count@non@spaces}[1]{%
774   \def\hyxmp@one@token{#1}%
775   \ifx\hyxmp@one@token\empty
776   \else
777     \advance\@tempcnta by 1
778     \expandafter\hyxmp@count@non@spaces
779   \fi
780 }
```

3.6.4 Embedding using X_YT_EX

`\hyxmp@embed@packet@xetex` Embed the XMP packet using `xdvipdfmx`-specific `\special` commands. I don’t know how to tell `xdvipdfmx` always to leave the Metadata stream uncompressed, so the XMP metadata is likely to be missed by non-PDF-aware XMP viewers.

```

781 \newcommand*\hyxmp@embed@packet@xetex{%
782   \special{pdf:stream @hyxmp@Metadata (\hyxmp@xml)
783     <<
784       /Type /Metadata
785       /Subtype /XML
786     >>
787   }%
788   \special{pdf:put @catalog
789     <<
790       /Metadata @hyxmp@Metadata
791     >>
792   }%
793 }
```

3.7 Final clean-up

Having saved the category code of “ ” at the start of the package code (Section 3.1), we now restore that character’s original category code.

```

794 \catcode'\="=\hyxmp@dq@code
```

4 Future Work

Help wanted Ideally, `\xmpquote` should automatically replace all commas with `\xmpcomma`. Unfortunately, my \TeX skills are insufficient to pull that off. If you know a way to make `\xmpquote{Hello, world}` work with both Unicode and non-Unicode encodings and with all \TeX engines (\pdfTeX , \LuaTeX , \XeTeX , etc.), please send me a code patch.

References

- [1] Adobe Systems, Inc., San Jose, California. *Adobe Acrobat X SDK Help, pdfmark Reference*. Available from <http://www.adobe.com/devnet/acrobat/documentation.html>.
- [2] Adobe Systems, Inc. *PostScript Language Reference Manual*. Addison-Wesley, 2nd edition, January 1996, ISBN: 0-201-18127-4.
- [3] Adobe Systems, Inc., San Jose, California. *Document Management—Portable Document Format—Part 1: PDF 1.7*, July 2008. ISO 32000-1 standard document. Available from http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000_2008.pdf.
- [4] Adobe Systems, Inc., San Jose, California. *XMP Specification Part 1: Data model, Serialization, and Core Properties*, July 2010. Available from <http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/xmp/pdfs/XMPSpecificationPart1.pdf>.
- [5] Michael Downes. Around the bend #15, answers, 4th (last) installment. `comp.text.tex` newsgroup posting, January 3, 1994. Archived by Google at <http://groups.google.com/group/comp.text.tex/msg/7da7643b9e8f3b48>.
- [6] Internet Assigned Numbers Authority. Language subtag registry, January 11, 2011. Available from <http://www.iana.org/assignments/language-subtag-registry>.

Change History

v1.0		Daniel Schömer for the bug report	31
General: Initial version	1		
v1.1		v1.2	
<code>\hyxmp@construct@packet</code> : Explicitly set the category codes of characters $\langle EF \rangle$, $\langle BB \rangle$, and $\langle BF \rangle$ to “letter”. Thanks to		General: Made the package compatible with <code>ngerman</code> . Thanks to Tobias Mueller for the bug report.	9

\hyxmp@embed@packet@xetex:		\hyxmp@construct@packet:	Modified by Heiko Oberdiek to use an appropriate BOM representation via \hyxmp@bom	31
Added support for the Xe _{La} TeX backend (xdvipdfmx)	34	\hyxmp@crap@convert:	Added by Heiko Oberdiek	20
\hyxmp@photoshop@data:	Added support for the Photoshop schema	\hyxmp@crap@test:	Added by Heiko Oberdiek	20
	30	\hyxmp@dc@schema:	Added support for dc:language and dc:source	28
v1.3		\hyxmp@is@unicode:	Added by Heiko Oberdiek	17
General:	Introduced the pdfmetalang package option, which enables an author to specify the language in which he wrote the document's metadata	\hyxmp@list@to+xml:	Modified by Heiko Oberdiek to use the new Unicode-processing macros	27
	13	\hyxmp@photoshop@data:	Simplified using \hyxmp@add@simple	30
\hyxmp@reencode:	Introduced this macro to re-encode Unicode strings as 8-bit strings before manipulating them into XMP schema. This change addresses a bug reported by Martin Münch	\hyxmp@ProcessKeyvalOptions:	Added this macro	12
	16	\hyxmp@reencode:	Replaced with an empty macro by Heiko Oberdiek	16
v1.4		\hyxmp@skiptorelax:	Added by Heiko Oberdiek	20
\hyxmp@mm@schema:	Renamed the xapMM namespace prefix to xmpMM	\hyxmp@skipzeros:	Added by Heiko Oberdiek	19
	29	\hyxmp@SpaceOther:	Added by Heiko Oberdiek	20
\hyxmp@rdf@dc:	Included metadata in the x-default language regardless of the specified metadata language	\hyxmp@string:	Added this macro	26
	26	\hyxmp@toxml:	Added by Heiko Oberdiek	18
\hyxmp@xmpRights@schema:	Renamed the xapRights namespace prefix to xmpRights		Escaped parentheses written with pdfmarks to prevent dvips from line-wrapping the XMP packet	18
	28	\hyxmp@toxml@unicodetex:	Added by Heiko Oberdiek	19
v1.5		\hyxmp@xetex@crap:	Added by Heiko Oberdiek	19
General:	Made the XMP inclusion more robust. Thanks to Heiko Oberdiek for the bug report and suggested modifications.	\hyxmp+xmlify:	Completely rewritten by Heiko Oberdiek to better support Unicode-enabled TeX programs	16
	9	\hyxmp@xmp@basic@schema:	Added this macro	29
v2.0		\hyxmp@xmpRights@schema:	Modified to include xmpRights:Marked only when pdfcopyright is specified and xmpRights:WebStatement only	
General:	Added support for the XMP Basic schema and miscellaneous other bits of metadata			
	1			
	Heiko Oberdiek's major rewrite of the code to better support native-Unicode TeX implementations (Xe _{La} TeX and Lua _{La} TeX)			
	1			
	New \AtBeginDocument code from Heiko Oberdiek to properly encode \pdfmetalang			
	13			
\hyxmp@add@to+xml:	Updated also to replace commas			
	24			
\hyxmp@bom:	Added by Heiko Oberdiek			
	30			
\hyxmp@comma:	Added this macro			
	14			

G

\g@addto@macro 411, 413, 415

Ghostsript 5

H

\Hy@driver 4, 680, 684, 688, 693

hyperref . . 1, 3, 4, 7, 9, 10, 12, 13, 31, 32, 37

\hypersetup 66

hyperxmp . . . 1–4, 7–10, 13, 14, 16, 22, 37

\hyxmp@is@unicode . 197

\hyxmp@add@simple 490, 491, 493, 500, 566, 567, 594, 616–620, 634, 635

\hyxmp@add@to@xml 436, 455, 486, 495, 505, 514, 520, 524, 534, 542, 548, 555, 568, 584, 590, 595, 603, 612, 621, 629, 638, 658, 670

\hyxmp@append@hex 383, 402–405

\hyxmp@append@hex@iv . 401, 409, 410, 412, 414, 416–418

\hyxmp@at@end . . . 3, 100

\hyxmp@big@prime . . . 357, 360, 370, 380

\hyxmp@big@prime@ii 357, 379

\hyxmp@bom 643, 658

\hyxmp@comma 33, 50, 129

\hyxmp@commas@to@list 113, 136, 540

\hyxmp@commas@to@list@i 115, 117

\hyxmp@concat@metadata 71

\hyxmp@construct@packet 656, 678

\hyxmp@count@non@spaces 762, 773

\hyxmp@count@spaces 761, 764

\hyxmp@crap@convert 279, 313

\hyxmp@crap@result 269, 305

\hyxmp@crap@test 276, 301

\hyxmp@create@uuid 407, 424, 434

\hyxmp@dc@schema 554, 666

\hyxmp@def@DocumentID 420, 601

\hyxmp@def@InstanceID 426, 602

\hyxmp@DocumentID 420, 606

\hyxmp@dq@code . . 1, 794

\hyxmp@driver . . . 3, 677

\hyxmp@embed@packet 102, 677

\hyxmp@embed@packet@dvi@pdfm 685, 743

\hyxmp@embed@packet@pdfmark 697, 713

\hyxmp@embed@packet@pdftex 681, 703

\hyxmp@embed@packet@xetex 689, 781

\hyxmp@find@metadata 71, 101

\hyxmp@hash . . . 449, 662

\hyxmp@have@any . . . 478

\hyxmp@Hyp@pdfauthor 26

\hyxmp@Hyp@pdfkeywords 43

\hyxmp@hypersetup . . 66

\hyxmp@InstanceID 426, 607

\hyxmp@is@unicode 165, 182, 197

\hyxmp@legal 573

\hyxmp@list . . . 540, 546

\hyxmp@list@to@xml 531, 563–565

\hyxmp@mm@schema 600, 669

\hyxmp@modulo@a . . . 351, 370, 380, 386

\hyxmp@num 313

\hyxmp@one@token 359, 363, 765, 766, 774, 775

\hyxmp@padding 453, 673

\hyxmp@pdf@schema 477, 664

\hyxmp@pdfauthor 32, 563

\hyxmp@pdfkeywords 49, 564

\hyxmp@photoshop@data 625

\hyxmp@photoshop@schema 625, 667

\hyxmp@ProcessKeyvalOptions 61

\hyxmp@rand@num . . . 376, 385, 423, 433

\hyxmp@rdf@dc 510, 560–562

\hyxmp@redefine@Hyp 25, 63, 68

\hyxmp@reencode . . . 159

\hyxmp@rights 573, 576, 580, 582

\hyxmp@seed@rng 359, 422, 432

\hyxmp@seed@rng@i 361, 363

\hyxmp@seed@string . . . 421, 422, 427, 432

\hyxmp@set@rand@num 376, 384

\hyxmp@skiptorelax 306, 312

\hyxmp@skipzeros . . . 264

\hyxmp@Space@other 273, 286

\hyxmp@string 500

\hyxmp@string@len 744, 759

\hyxmp@sublist 118, 119, 122, 123

\hyxmp@temp@list . . . 134

\hyxmp@temp@str . . . 134

\hyxmp@text 161, 239, 269, 313

\hyxmp@today 465, 565, 616–618

\hyxmp@toxml . . . 191, 214

<code>\hyxmp@toxml@unicodetex</code> 179, 239	L	<code>pdflicenseurl</code> (option) 3, 4, 9, 28, 37
<code>\hyxmp@trimb</code> .. 147, 150	<code>\lccode</code> 198, 199, 288, 318,	<code>pdfmark</code> (option) 32
<code>\hyxmp@trimc</code> .. 150, 151	325, 332, 339,	<code>\pdfmark</code> .. 714, 717, 721, 731, 735, 739
<code>\hyxmp@trimspaces</code> 122, 143	440, 444, 445, 645	<code>pdfmetalang</code> (option) 3, 4, 9
<code>\hyxmp@try</code> 269	<code>\lowercase</code> 202, 289, 319, 326,	<code>\pdfminorversion</code> ... 493
<code>\hyxmp@unicodetexfalse</code> 153	333, 340, 446, 646	<code>\pdfobj</code> 706
<code>\hyxmp@unicodetextrue</code> 153	Lua \LaTeX 5, 7	<code>pdfproducer</code> (option) .. 3
<code>\hyxmp@x@default</code> 95, 476 , 518, 525	Lua \TeX .. 16, 19, 35, 36	<code>\pdfstringdef</code> 16, 18, 20, 22, 24
<code>\hyxmp@xetex@crap</code> 170, 269	M	<code>pdfsubject</code> (option) 3, 9, 27
<code>\hyxmp+xml</code> 446, 453 , 656 , 709, 733, 744, 751, 782	Metadata 5, 31, 34	<code>pdf\TeX</code> ... 9, 18, 32, 35
<code>\hyxmp+xmlified</code> 161 , 506, 521, 525, 540	<code>\month</code> 429, 466, 467, 469	<code>pdftitle</code> (option) . 3, 9, 27
<code>\hyxmp+xmlify</code> .. 99, 161 , 504, 513, 539	N	<code>photoshop:AuthorsPosition</code> 2, 30
<code>\hyxmp@xmp@basic@schema</code> 611 , 668	<code>nativepdf</code> (option) ... 32	<code>photoshop:CaptionWriter</code> 2, 30
<code>\hyxmp@xmpRights@schema</code> 572 , 665	<code>\newif</code> 153	PI 24
<code>\hyxmp@zero</code> 322, 329, 336, 342, 347	<code>\next</code> 117 , 363	<code>\ProcessKeyvalOptions</code> 61
I	<code>ngerman</code> 9, 35	<code>ps2pdf</code> (option) 32
IETF 4	<code>\number</code> 316, 318, 320, 325, 327, 332, 334	Q
<code>\ifhyxmp@unicodetex</code> 153 , 164, 644	P	<code>\Q</code> 143, 152
<code>ifxetex</code> 10	<code>\PackageWarningNoLine</code> 85, 105, 692	R
<code>\ifxetex</code> 169	PDF 1–5, 7, 13, 14, 18, 23–26, 31, 34	<code>rdf:li</code> 2
<code>Info</code> 5	<code>pdf:Keywords</code> 2, 26	<code>rdf:Seq</code> 2
<code>intcalc</code> 10	<code>pdf:PDFVersion</code> 2	<code>\renewcommand</code> 62
<code>\intcalcDiv</code> 318, 325, 332	<code>pdf:Producer</code> 2, 26	<code>\RequirePackage</code> 7, 10–14
<code>\intcalcMod</code> 320, 327, 334	<code>pdfauthor</code> (option) 3, 8–10, 27	S
ISO 10	<code>pdfauthor</code> (option) 3, 4, 9	<code>\SE->pdfdoc@03</code> 160
J	<code>pdfcaptionwriter</code> (op- tion) 3, 4, 9	<code>\special</code> 745, 753, 782, 788
<code>\jobname</code> ... 86, 106, 421, 428, 567, 694	<code>\pdfcatalog</code> 710	<code>stringenc</code> 10
K	<code>\pdfcompresslevel</code> .. 705	<code>\StringEncodingConvert</code> 166, 172, 183, 186, 281
Keywords 5	<code>pdfcopyright</code> (option) . . 3, 4, 9, 27, 28, 36	Subject 5
<code>\KV@Hyp@pdfauthor</code> .. 32	PDFDocEncoding 10, 16, 17	T
<code>\KV@Hyp@pdfkeywords</code> 49	<code>pdfescape</code> 10	<code>\TeX</code> 16, 18, 19, 21, 22, 29, 34–36
<code>kvoptions</code> 10, 12	<code>pdfkeywords</code> (option) 3, 8–10, 28	<code>textures</code> (option) 32
	<code>pdflang</code> (option) 3, 4, 9, 13, 28	<code>\time</code> 430
	<code>\pdflastobj</code> 710	Title 5
	<code>pdf\LaTeX</code> 3, 5	U
		<code>\uccode</code> 243, 256

Unicode	10, 16–20, 27, 30, 35, 36				
<code>\uppercase</code>	244, 257				
URL	2, 4, 10, 28, 29				
<code>\usepackage</code>	107				
UTF-16BE	17				
UTF-32BE	16, 17				
UTF-8	17				
UUID	21, 23, 24				
V					
<code>\vfuzz</code>	151				
<code>vtexpdfmark</code> (option) .	32				
X					
<code>\x</code>	269				
<code>xdvipdfmx</code>	7, 34				
$X_{\text{L}}^{\text{LATEX}}$	5, 7				
$X_{\text{F}}^{\text{TEX}}$	10, 16, 19, 34–36				
XML	1, 2, 13, 16–19, 24, 26, 27, 31				
XMP	1–3, 5, 7– 9, 13–15, 18, 21, 24–26, 29, 32–34, 36				
XMP	36				
<code>xmp:BaseURL</code>	2				
<code>xmp:CreateDate</code>	2				
<code>xmp:CreatorTool</code>	2				
<code>xmp:MetadataDate</code>	2				
<code>xmp:ModifyDate</code>	2				
<code>\xmpcomma</code>	32, 49, 128				
<code>xmpincl</code>	3				
<code>xmpMM:DocumentID</code>	2, 21, 29				
<code>xmpMM:InstanceID</code>	2, 21, 29				
<code>\xmpquote</code>	32, 49, 133				
<code>xmpRights:Marked</code>	2, 28, 36				
<code>xmpRights:WebStatement</code>	2, 28, 36				
<code>\XMPTruncateList</code>	134				
Y					
<code>\year</code>	429, 465				