

# The hyperxmp package<sup>\*</sup>

Scott Pakin  
scott+hyxmp@pakin.org

September 12, 2012

## Abstract

`hyperxmp` makes it easy for an author to include XMP metadata in a PDF document produced by  $\text{\LaTeX}$ . `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  $\text{\LaTeX}$  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>
```

---

<sup>\*</sup>This document corresponds to `hyperxmp` v2.0, dated 2012/09/10.

```

    </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<sup>A</sup>T<sub>E</sub>X 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 pdfL<sup>A</sup>T<sub>E</sub>X and only when in PDF-generating mode. `hyperxmp` additionally works with a few other PDF-producing L<sup>A</sup>T<sub>E</sub>X 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}` somewhere in your document's preamble. `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<sup>A</sup>T<sub>E</sub>X 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 $\text{\LaTeX}$
- Lua $\text{\LaTeX}$
- $\text{\LaTeX}$  + Dvipdfm
- $\text{\LaTeX}$  + Dvips + Adobe Acrobat Distiller
- X $\text{\LaTeX}$

Unfortunately, the  $\text{\LaTeX}$  + 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: X<sub>g</sub>L<sup>A</sup>T<sub>E</sub>X object compression** X<sub>g</sub>L<sup>A</sup>T<sub>E</sub>X (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<sup>A</sup>T<sub>E</sub>X), (2) pass the `--output-driver="xdvipdfmx -z0"` option to X<sub>g</sub>L<sup>A</sup>T<sub>E</sub>X 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.

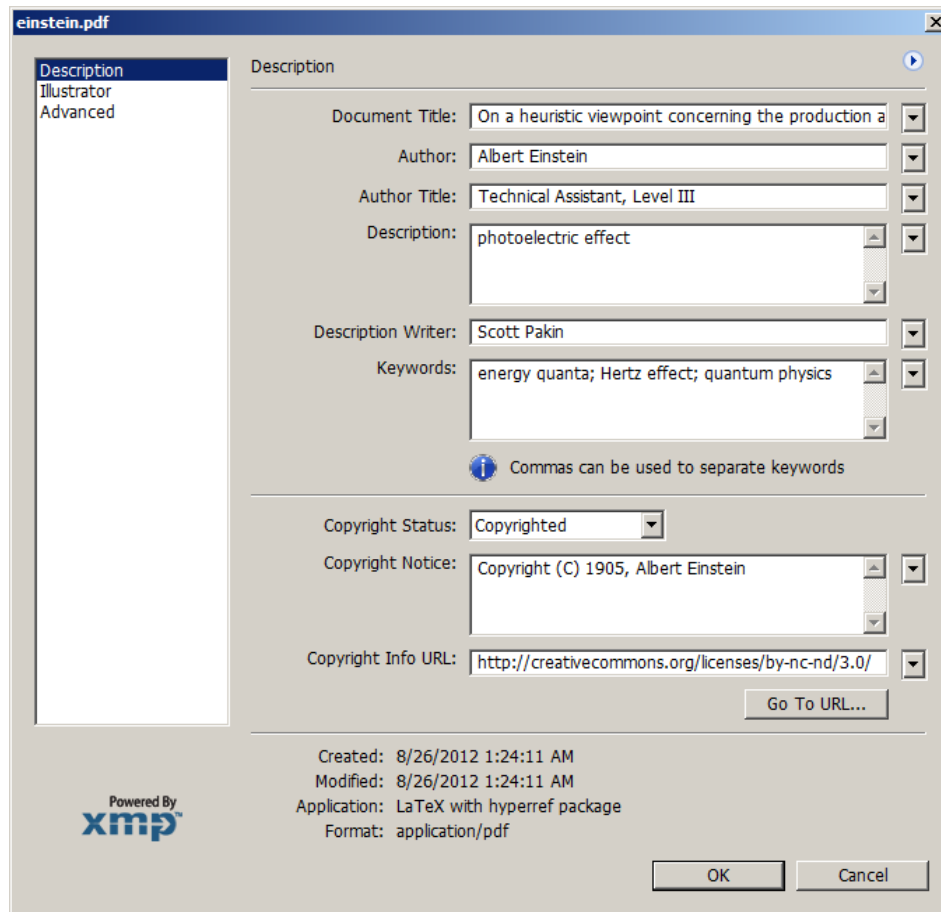


Figure 1: XMP metadata as it appears in Adobe Acrobat

**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<sup>A</sup>T<sub>E</sub>X 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<sub>E</sub>X, 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 addition L<sup>A</sup>T<sub>E</sub>X 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<sup>A</sup>T<sub>E</sub>X’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<sub>La</sub>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}}
```

`\hyxmp@ProcessKeyvalOptions` 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.

```
25 \let\hyxmp@ProcessKeyvalOptions=\ProcessKeyvalOptions
26 \renewcommand*{\ProcessKeyvalOptions}{%
```

`\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.

```

27 \ifundefined{KV@Hyp@pdfauthor}{-}{%
28   \ifundefined{hyxmp@Hyp@pdfauthor}{-}{%
29     \expandafter\let\expandafter\hyxmp@Hyp@pdfauthor
30       \csname KV@Hyp@pdfauthor\endcsname
31   }-}%
32 }%
```

`\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).

```

33 \define@key{Hyp}{pdfauthor}{%
34   \let\xmpcomma=\hyxmp@comma
35   \def\xmpquote####1{####1}%
36   \hyxmp@Hyp@pdfauthor{##1}%
37   \global\let\hyxmp@pdfauthor=\@pdfauthor
38   \def\xmpcomma{,%}
39   \def\xmpquote####1{"####1"%}
40   \hyxmp@Hyp@pdfauthor{##1}%
41   \let\xmpcomma=\relax
42   \let\xmpquote=\relax
43 }%
```

`\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`.

```

44 \ifundefined{KV@Hyp@pdfkeywords}{-}{%
45   \ifundefined{hyxmp@Hyp@pdfkeywords}{-}{%
46     \expandafter\let\expandafter\hyxmp@Hyp@pdfkeywords
47       \csname KV@Hyp@pdfkeywords\endcsname
48   }-}%
49 }%
```

`\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).

```

50 \define@key{Hyp}{pdfkeywords}{%
51   \let\xmpcomma=\hyxmp@comma
```

```

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

```

Process the calling package's options as normal but with the preceding substitutions.

```

61 \hyxmp@ProcessKeyvalOptions
62 }

```

```

\hyxmp@find@metadata Issue a warning message if the author failed to include any metadata at all. Note
\hyxmp@concat@metadata that we don't consider \@pdfmetalang as metadata as that value is meaningful
                        only when used in conjunction with other information.
63 \newcommand*{\hyxmp@find@metadata}{%
64 \edef\hyxmp@concat@metadata{%
65 \@baseurl
66 \@pdfauthor
67 \@pdfauthortitle
68 \@pdfcaptionwriter
69 \@pdfcopyright
70 \@pdfkeywords
71 \@pdflang
72 \@pdflicenseurl
73 \@pdfsubject
74 \@pdftitle
75 }%
76 \ifx\hyxmp@concat@metadata\@empty
77 \PackageWarningNoLine{hyperxmp}{%
78 \jobname.tex did not specify any metadata to\MessageBreak
79 include in the XMP packet.\space\space Please see the hyperxmp\MessageBreak
80 documentation for instructions on how to provide\MessageBreak
81 metadata values to hyperxmp}%
82 \fi
83 }

```

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.

```

84 \AtBeginDocument{%
85 \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.

```

86 \ifx\@pdflang\@empty
87 \let\@pdfmetalang=\hyxmp@x@default
88 \else
89 \edef\@pdfmetalang{\@pdflang}%
90 \fi
91 \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.

```

92 \hyxmp@at@end{%
93 \hyxmp@find@metadata
94 \hyxmp@embed@packet
95 }%
96 }%
97 {\PackageWarningNoLine{hyperxmp}{%
98 \jobname.tex failed to include a\MessageBreak
99 \string\usepackage\string{hyperref\string}
100 in the preamble.\MessageBreak
101 Consequently, all hyperxmp functionality will be\MessageBreak
102 disabled}%
103 }%
104 }

```

### 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<sup>A</sup>T<sub>E</sub>X 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 `&lt;scott+hyxmp@pakin.org&gt;`).

#### 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<sup>A</sup>T<sub>E</sub>X `\@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.)
105 \newcommand*\hyxmp@commas@to@list}[2]{%
106 \gdef#1{%
107 \expandafter\hyxmp@commas@to@list@i\expandafter#1#2,,%
108 }

\hyxmp@commas@to@list@i Recursively construct macro #1 from comma-separated list #2. Stop if #2 is empty.
\next 109 \def\hyxmp@commas@to@list@i#1#2,{%

```

```

110 \gdef\hyxmp@sublist{#2}%
111 \ifx\hyxmp@sublist\@empty
112   \let\next=\relax
113 \else
114   \hyxmp@trimspaces\hyxmp@sublist
115   \@cons{#1}{\hyxmp@sublist}%
116   \def\next{\hyxmp@commas@to@list@i{#1}}%
117 \fi
118 \next
119 }

```

`\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.

```
120 \let\xmpcomma=\relax
```

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

```

121 \bgroup
122 \catcode'\^^C=11
123 \gdef\hyxmp@comma{^^C}
124 \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.

```
125 \let\xmpquote=\relax
```

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

```

\hyxmp@temp@str
\hyxmp@temp@list
\@elt
126 \newcommand{\XMPTruncateList}[1]{%
127   \edef\hyxmp@temp@str{\csname hyxmp@#1\endcsname}%
128   \hyxmp@commas@to@list{\hyxmp@temp@list}{\hyxmp@temp@str}%
129   \def\@elt##1{%
130     \expandafter\gdef\csname @#1\endcsname{##1}%
131     \let\@elt=\@gobble
132   }
133   \hyxmp@temp@list
134 }

```

### 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.

```
135 \catcode'\Q=3
\hyxmp@trimspaces\x redefines \x to have the same replacement text sans leading
and trailing space tokens.
136 \newcommand{\hyxmp@trimspaces}[1]{%
    Use grouping to emulate a multi-token afterassignment queue.
137 \begingroup
    Put “\toks 0 {” into the afterassignment queue.
138 \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.
139 \expandafter\hyxmp@trimb\expandafter\noexpand#1Q Q}%
    Transfer the trimmed text back into #1.
140 \edef#1{\the\toks0}%
141 }
```

`\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.

```
142 \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.

```
143 \def\hyxmp@trimc#1Q#2{\afterassignment\endgroup \vfuzz\the\vfuzz#1}
144 \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` X<sub>Y</sub>TeX and LuaTeX natively support Unicode. We define the conditional `\ifhyxmp@unicodetex` to check for these so we can properly handle encoding

`\hyxmp@unicodetexttrue`

`\hyxmp@unicodetexfalse`

conversions. The trick here is that Unicode T<sub>E</sub>X implementations compare decimal 64 to hexadecimal 40 (decimal 64), specified with four carets, and take the TRUE branch; non-Unicode T<sub>E</sub>X implementations compare decimal 64 to character “^” (decimal 94), ignore the “^^0040” and the rest of the TRUE branch, and take the FALSE branch.

```

145 \newif\ifhyxmp@unicodetex
146 \ifnum64='^^^^0040\relax
147   \hyxmp@unicodetextrue
148 \else
149   \hyxmp@unicodetexfalse
150 \fi

```

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

```

151 \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.

```

152 \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@xmlified` but with all occurrences of “<” replaced with `&lt;`; all occurrences of “>” replaced  
`\hyxmp@text` with `&gt;`; and all occurrences of “&” replaced with `&amp;`.

```

153 \newcommand*{\hyxmp@xmlify}[1]{%
154   \gdef\hyxmp@xmlified{}%

```

Escaped PDF string → PDFDocEncoding/Unicode

```

155   \EdefUnescapeString\hyxmp@text{#1}%
156   \ifhyxmp@unicodetex

```

PDFDocEncoding/Unicode → UTF-32BE

```

157     \hyxmp@is@unicode\hyxmp@text{%
158       \StringEncodingConvert
159       \hyxmp@text\hyxmp@text{utf16be}{utf32be}%
160     }{%
161       \ifxetex
162         \hyxmp@xetex@crap
163       \else
164         \StringEncodingConvert
165         \hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
166       \fi
167     }%

```

UTF-32BE → UTF-32BE as hex string

```

168     \EdefEscapeHex\hyxmp@text{\hyxmp@text}%

```



UTF-32BE → XML in ASCII

```

169 \edef\hyxmp@text{%
170 \expandafter
171 }\expandafter\hyxmp@toxml@unicodetex\hyxmp@text
172 \relax\relax\relax\relax\relax\relax\relax
173 \else

```

PDFDocEncoding/Unicode → UTF-8

```

174 \hyxmp@is@unicode\hyxmp@text{%
175 \StringEncodingConvert
176 \hyxmp@text\hyxmp@text{utf16be}{utf8}%
177 }{%
178 \StringEncodingConvert
179 \hyxmp@text\hyxmp@text{pdfdoc}{utf8}%
180 }%

```

UTF-8 → UTF-8 as hex string

```

181 \EdefEscapeHex\hyxmp@text{\hyxmp@text}%

```

UTF-8 as hex string → XML in UTF-8 as hex string

```

182 \edef\hyxmp@text{%
183 \expandafter\hyxmp@toxml\hyxmp@text\@empty\@empty
184 }%

```

XML in UTF-8 as hex string → XML in UTF-8

```

185 \EdefUnescapeHex\hyxmp@text{\hyxmp@text}%
186 \fi
187 \global\let\hyxmp@xmlified\hyxmp@text
188 }

```

\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.

```

189 \begingroup
190 \lccode'\<=254 %
191 \lccode'\>=255 %
192 \catcode254=12 %
193 \catcode255=12 %
194 \lowercase{\endgroup
195 \def\hyxmp@is@unicode#1{%
196 \expandafter\hyxmp@@is@unicode#1<>\@nil
197 }%
198 \def\hyxmp@@is@unicode#1<>#2\@nil{%
199 \ifx\#1\%
200 \expandafter\@firstoftwo
201 \else
202 \expandafter\@secondoftwo
203 \fi
204 }%
205 }

```

`\hyxmp@toxml` Replace the characters “<”, “&”, and “>” with XML entities when using a non-native-Unicode T<sub>E</sub>X (T<sub>E</sub>X or pdfT<sub>E</sub>X).

```

206 \def\hyxmp@toxml#1#2{%
207   \ifx#1\@empty
208   \else
209     \ifnum"#1#2='\& %
210       26616D703B% &amp;
211     \else\ifnum"#1#2='\< %
212       266C743B% &lt;
213     \else\ifnum"#1#2='\> %
214       2667743B% &gt;
215     \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.

```

216   \@ifundefined{pdfmark}{%
217     #1#2%
218   }{%
219     \ifnum"#1#2='\( %
220       5C28% \(
221     \else\ifnum"#1#2='\) %
222       5C29% \)
223     \else
224       #1#2%
225     \fi\fi
226   }%
227   \fi\fi\fi
228   \expandafter\hyxmp@toxml
229 \fi
230 }

```

`\hyxmp@toxml@unicodetex` Replace the characters “<”, “&”, and “>” with XML entities when using a native-Unicode T<sub>E</sub>X (X<sub>L</sub>T<sub>E</sub>X or LuaT<sub>E</sub>X).

`\hyxmp@text`

```

231 \def\hyxmp@toxml@unicodetex#1#2#3#4#5#6#7#8{%
232   \ifx#1\relax
233   \else
234     \ifnum"#1#2#3#4#5#6#7#8>127 %
235       \uccode'\*"#1#2#3#4#5#6#7#8\relax
236     \uppercase{%
237       \edef\hyxmp@text{\hyxmp@text *}%

```

```

238     }%
239     \else\ifnum"#7#8='< %
240         \edef\hyxmp@text{\hyxmp@text &lt;}%
241     \else\ifnum"#7#8='& %
242         \edef\hyxmp@text{\hyxmp@text &amp;}%
243     \else\ifnum"#7#8='> %
244         \edef\hyxmp@text{\hyxmp@text &gt;}%
245     \else\ifnum"#7#8='\ %
246         \edef\hyxmp@text{\hyxmp@text\space}%
247     \else
248         \uccode'\*"#7#8\relax
249         \uppercase{%
250             \edef\hyxmp@text{\hyxmp@text *}%
251         }%
252     \fi\fi\fi\fi\fi
253     \expandafter\hyxmp@toxml@unicodetex
254 \fi
255 }

```

`\hyxmp@skipzeros` Skip over leading zeroes in the input argument.

```

256 \def\hyxmp@skipzeros#1{%
257     \ifx#10%
258         \expandafter\hyxmp@skipzeros
259     \fi
260 }

```

`\x` In the case of X<sub>Y</sub>TEX, the strings defined by `\pdfstringdef` can contain big characters. In this case, the string is treated as Unicode.

```

\hyxmp@xetex@crap \hyxmp@try
\hyxmp@crap@result \def\x#1{\endgroup
\hyxmp@text \def\hyxmp@xetex@crap{%
264     \edef\hyxmp@try{%
265         \expandafter\hyxmp@SpaceOther\hyxmp@text#1\@nil
266     }%
267     \let\hyxmp@crap@result=N%
268     \expandafter\hyxmp@crap@test\hyxmp@try\relax
269     \ifx\hyxmp@crap@result Y%
270         \let\hyxmp@text\@empty
271         \expandafter\hyxmp@crap@convert\hyxmp@try\relax
272     \else
273         \StringEncodingConvert\hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
274     \fi
275 }%
276 }
277 \x{ }

```

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

```

278 \begingroup
279 \catcode'\~ =12 %

```

```

280 \lccode'\~='\' %
281 \lowercase{\endgroup
282 \def\hyxmp@SpaceOther#1 #2\@nil{%
283   #1%
284   \ifx\relax#2\relax
285     \expandafter\@gobble
286   \else
287     ~%
288     \expandafter\@firstofone
289   \fi
290   {\hyxmp@SpaceOther#2\@nil}%
291 }%
292 }

```

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

```

293 \def\hyxmp@crap@test#1{%
294   \ifx#1\relax
295   \else
296     \ifnum'#1>127 %
297       \let\hyxmp@crap@result=Y%
298       \expandafter\expandafter\expandafter\hyxmp@skiptorelax
299     \else
300       \expandafter\expandafter\expandafter\hyxmp@crap@test
301     \fi
302   \fi
303 }

```

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

```

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

```

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

```

\hyxmp@num 305 \def\hyxmp@crap@convert#1{%
\hyxmp@text 306   \ifx#1\relax
307   \else
308     \edef\hyxmp@num{\number'#1}%
309     \ifnum\hyxmp@num>"FFFFFF %
310       \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"1000000}\relax
311       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
312       \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"1000000}}%
313     \else
314       \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
315     \fi
316     \ifnum\hyxmp@num>"FFFF %
317       \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"10000}\relax
318       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
319       \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"10000}}%
320     \else
321       \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
322     \fi

```

```

323 \ifnum\hyxmp@num>"FF %
324 \lccode'\!=\intcalDiv{\hyxmp@num}{\number"100}\relax
325 \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
326 \edef\hyxmp@num{\intcalMod{\hyxmp@num}{\number"100}}%
327 \else
328 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
329 \fi
330 \ifnum\hyxmp@num>0 %
331 \lccode'\!=\hyxmp@num\relax
332 \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
333 \else
334 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
335 \fi
336 \expandafter\hyxmp@crap@convert
337 \fi
338 }

```

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

```

339 \begingroup
340 \catcode0=12 %
341 \gdef\hyxmp@zero{^^00}%
342 \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<sub>E</sub>X 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.

```

343 \def\hyxmp@modulo@a#1{%
344 \@tempcntb=\@tempcnta
345 \divide\@tempcntb by #1
346 \multiply\@tempcntb by #1
347 \advance\@tempcnta by -\@tempcntb
348 }

```

`\hyxmp@big@prime` Define a couple of large prime numbers that can still be stored in a T<sub>E</sub>X counter.

```

\hyxmp@big@prime@ii 349 \def\hyxmp@big@prime{536870923}
350 \def\hyxmp@big@prime@ii{536870027}

```

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

```

\hyxmp@one@token 351 \def\hyxmp@seed@rng#1{%
352 \@tempcnta=\hyxmp@big@prime
353 \futurelet\hyxmp@one@token\hyxmp@seed@rng@i#1\@empty
354 }

```

`\hyxmp@seed@rng@i` Do all of the work for `\hyxmp@seed@rng`. For each character code  $c$  of the input

`\hyxmp@one@token` text, assign  $\backslash@tempcnta \leftarrow 3 \cdot \backslash@tempcnta + c \pmod{\backslashhyxmp@big@prime}$ .

```

\next 355 \def\hyxmp@seed@rng@i{%
356   \ifx\hyxmp@one@token\@empty
357     \let\next=\relax
358   \else
359     \def\next##1{%
360       \multiply\@tempcnta by 3
361       \advance\@tempcnta by '##1
362       \hyxmp@modulo@a{\hyxmp@big@prime}%
363       \futurelet\hyxmp@one@token\hyxmp@seed@rng@i
364     }%
365   \fi
366 \next
367 }

```

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

```

368 \def\hyxmp@set@rand@num{%
369   \@tempcnta=\hyxmp@rand@num
370   \multiply\@tempcnta by 3
371   \advance\@tempcnta by \hyxmp@big@prime@ii
372   \hyxmp@modulo@a{\hyxmp@big@prime}%
373   \xdef\hyxmp@rand@num{\the\@tempcnta}%
374 }

```

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

```

375 \def\hyxmp@append@hex#1{%
376   \hyxmp@set@rand@num
377   \@tempcnta=\hyxmp@rand@num
378   \hyxmp@modulo@a{16}%
379   \ifnum\@tempcnta<10
380     \xdef#1{#1\the\@tempcnta}%
381   \else

```

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

```

382     \advance\@tempcnta by -10
383     \ifcase\@tempcnta
384       \xdef#1{#1a}%
385     \or\xdef#1{#1b}%
386     \or\xdef#1{#1c}%
387     \or\xdef#1{#1d}%
388     \or\xdef#1{#1e}%
389     \or\xdef#1{#1f}%
390   \fi
391 \fi
392 }

```

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

```
393 \def\hyxmp@append@hex@iv#1{%
394   \hyxmp@append@hex#1%
395   \hyxmp@append@hex#1%
396   \hyxmp@append@hex#1%
397   \hyxmp@append@hex#1%
398 }
```

`\hyxmp@create@uuid` Define macro `#1` as a UUID of the form “`uuid:xxxxxx-xxx-xxx-xxxxxxxxxx`” 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`.

```
399 \def\hyxmp@create@uuid#1{%
400   \def#1{uuid:}%
401   \hyxmp@append@hex@iv#1%
402   \hyxmp@append@hex@iv#1%
403   \g@addto@macro#1{-}%
404   \hyxmp@append@hex@iv#1%
405   \g@addto@macro#1{-}%
406   \hyxmp@append@hex@iv#1%
407   \g@addto@macro#1{-}%
408   \hyxmp@append@hex@iv#1%
409   \hyxmp@append@hex@iv#1%
410   \hyxmp@append@hex@iv#1%
411 }
```

`\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.

```
412 \newcommand*{\hyxmp@def@DocumentID}{%
413   \edef\hyxmp@seed@string{\jobname:\@pdftitle:\@pdfauthor}%
414   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
415   \edef\hyxmp@rand@num{\the\@tempcnta}%
416   \hyxmp@create@uuid\hyxmp@DocumentID
417 }
```

`\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.

```
418 \newcommand*{\hyxmp@def@InstanceID}{%
419   \edef\hyxmp@seed@string{%
420     \jobname:\@pdftitle:\@pdfauthor:%
421     \the\year/\the\month/\the\day:%
422     \the\time
423   }%
424   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
425   \edef\hyxmp@rand@num{\the\@tempcnta}%

```

```

426 \hyxmp@create@uuid\hyxmp@InstanceID
427 }

```

### 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.

```

428 \newcommand*{\hyxmp@add@to+xml}[1]{%
429   \bgroup
430     \@tempcnta=0
431     \loop
432       \lccode\@tempcnta=\@tempcnta
433       \advance\@tempcnta by 1
434       \ifnum\@tempcnta<256
435         \repeat
436         \lccode'\_='\ \relax
437         \lccode'\^C='\,\relax
438         \lowercase{\xdef\hyxmp+xml{\hyxmp+xml#1}}%
439   \egroup
440 }

```

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

```

441 \bgroup
442 \catcode'\#=11
443 \gdef\hyxmp@hash{#}
444 \egroup

```

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

```

445 \bgroup
446 \xdef\hyxmp+xml{%
447   \hyxmp@add@to+xml{%
448     -----^^J%

```



```

449 }
450 \xdef\hyxmp@padding{\hyxmp+xml}%
451 \egroup
452 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
453 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
454 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
455 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
456 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}

```

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

```

457 \xdef\hyxmp@today{\the\year}%
458 \ifnum\month<10
459 \xdef\hyxmp@today{\hyxmp@today-0\the\month}%
460 \else
461 \xdef\hyxmp@today{\hyxmp@today-\the\month}%
462 \fi
463 \ifnum\day<10
464 \xdef\hyxmp@today{\hyxmp@today-0\the\day}%
465 \else
466 \xdef\hyxmp@today{\hyxmp@today-\the\day}%
467 \fi

```

`\hyxmp@x@default` Define an x-default string that we can use in comparisons with `\@pdfmetalang`.

```

468 \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.

```

469 \newcommand*{\hyxmp@pdf@schema}{%

```

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

```

470 \let\hyxmp@have@any=!%
471 \ifx\@pdfkeywords\@empty
472 \ifx\@pdfproducer\@empty
473 \let\hyxmp@have@any=\@empty
474 \fi
475 \fi
476 \ifx\hyxmp@have@any\@empty
477 \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).

```

478 \hyxmp@add@to+xml{%
479 -----<rdf:Description rdf:about=""^^J%
480 -----xmlns:pdf="http://ns.adobe.com/pdf/1.3/">^^J%
481 }%
482 \hyxmp@add@simple{pdf:Keywords}{\@pdfkeywords}%

```

```

483 \hyxmp@add@simple{pdf:Producer}{\@pdfproducer}%
484 \ifundefined{pdfminorversion}{\@pdfminorversion}%
485 \hyxmp@add@simple{pdf:PDFVersion}{1.\the\pdfminorversion}%
486 }%
487 \hyxmp@add@to@xml{%
488 -----</rdf:Description>^^J%
489 }%
490 \fi
491 }

```

`\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.

```

492 \newcommand*{\hyxmp@add@simple}[2]{%
493 \edef\hyxmp@string{#2}%
494 \ifx\hyxmp@string\@empty
495 \else
496 \hyxmp@xmlify{\hyxmp@string}%
497 \hyxmp@add@to@xml{%
498 -----<#1>\hyxmp@xmlified</#1>^^J%
499 }%
500 \fi
501 }

```

### 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.

```

502 \newcommand*{\hyxmp@rdf@dc}[2]{%
503 \ifx#2\@empty
504 \else
505 \hyxmp@xmlify{#2}%
506 \hyxmp@add@to@xml{%
507 -----<dc:#1>^^J%
508 -----<rdf:Alt>^^J%
509 }%
510 \ifx\@pdfmetalang\hyxmp@x@default
511 \else
512 \hyxmp@add@to@xml{%
513 -----<rdf:li xml:lang="\@pdfmetalang">\hyxmp@xmlified</rdf:li>^^J%
514 }%
515 \fi
516 \hyxmp@add@to@xml{%
517 -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@xmlified</rdf:li>^^J%
518 -----</rdf:Alt>^^J%
519 -----</dc:#1>^^J%
520 }%
521 \fi%

```

522 }%

`\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.

```
523 \newcommand*{\hyxmp@list@to@xml}[3]{%
524   \ifx#3\@empty
525   \else
526     \hyxmp@add@to@xml{%
527       -----<dc:#1>^^J%
528       -----<rdf:#2>^^J%
529     }%
530   \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`.

```
531   \hyxmp@xmlify{#3}%
532   \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
533   \def\@elt##1{%
534     \hyxmp@add@to@xml{%
535       -----<rdf:li>##1</rdf:li>^^J%
536     }%
537   }%
538   \hyxmp@list
539   \egroup
540   \hyxmp@add@to@xml{%
541     -----</rdf:#2>^^J%
542     -----</dc:#1>^^J%
543   }%
544   \fi
545 }
```

`\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<sup>A</sup>T<sub>E</sub>X and the `dc:source` property using the base name of the source file with `.tex` appended.

```
546 \newcommand*{\hyxmp@dc@schema}{%
547   \hyxmp@add@to@xml{%
548     -----<rdf:Description rdf:about=""^^J%
549     -----_xmlns:dc="http://purl.org/dc/elements/1.1/">^^J%
550     -----<dc:format>application/pdf</dc:format>^^J%
551   }%
552   \hyxmp@rdf@dc{title}{\@pdftitle}%
```

```

553 \hyxmp@rdf@dc{description}{\@pdfsubject}%
554 \hyxmp@rdf@dc{rights}{\@pdfcopyright}%
555 \hyxmp@list@to@xml{creator}{Seq}{\hyxmp@pdfauthor}%
556 \hyxmp@list@to@xml{subject}{Bag}{\hyxmp@pdfkeywords}%
557 \hyxmp@list@to@xml{date}{Seq}{\hyxmp@today}%
558 \hyxmp@add@simple{dc:language}{\@pdflang}%
559 \hyxmp@add@simple{dc:source}{\jobname.tex}%
560 \hyxmp@add@to@xml{%
561 -----</rdf:Description>^^J%
562 }%
563 }

```

### 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.

```

564 \newcommand*{\hyxmp@xmpRights@schema}{%

```

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

```

565 \let\hyxmp@rights=\@empty
566 \ifx\@pdflicenseurl\@empty
567 \else
568 \def\hyxmp@rights{YES}%
569 \fi
570 \ifx\@pdfcopyright\@empty
571 \else
572 \def\hyxmp@rights{YES}%
573 \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.

```

574 \ifx\hyxmp@rights\@empty
575 \else

```

Header

```

576 \hyxmp@add@to@xml{%
577 -----<rdf:Description rdf:about=""^^J%
578 -----xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/">^^J%
579 }%

```

Copyright indication

```

580 \ifx\@pdfcopyright\@empty
581 \else
582 \hyxmp@add@to@xml{%
583 -----<xmpRights:Marked>True</xmpRights:Marked>^^J%
584 }%
585 \fi

```

License URL

```
586 \hyxmp@add@simple{xmpRights:WebStatement}{\@pdflicenseurl}%  
Trailer  
587 \hyxmp@add@to@xml{%  
588 -----</rdf:Description>^^J%  
589 }%  
590 \fi  
591 }
```

### 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  $\text{\TeX}$ -based workflow.

```
592 \gdef\hyxmp@mm@schema{%  
593 \hyxmp@def@DocumentID  
594 \hyxmp@def@InstanceID  
595 \hyxmp@add@to@xml{%  
596 -----<rdf:Description rdf:about=""^^J%  
597 -----_xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"^^J%  
598 -----<xmpMM:DocumentID>\hyxmp@DocumentID</xmpMM:DocumentID>^^J%  
599 -----<xmpMM:InstanceID>\hyxmp@InstanceID</xmpMM:InstanceID>^^J%  
600 -----</rdf:Description>^^J%  
601 }%  
602 }
```

### 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`.

```
603 \newcommand*{\hyxmp@xmp@basic@schema}{%  
604 \hyxmp@add@to@xml{%  
605 -----<rdf:Description rdf:about=""^^J%  
606 -----_xmlns:xmp="http://ns.adobe.com/xap/1.0/"^^J%  
607 }%  
608 \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@today}%  
609 \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@today}%  
610 \hyxmp@add@simple{xmp:MetadataDate}{\hyxmp@today}%  
611 \hyxmp@add@simple{xmp:CreatorTool}{\@pdfcreator}%  
612 \hyxmp@add@simple{xmp:BaseURL}{\@baseurl}%  
613 \hyxmp@add@to@xml{%  
614 -----</rdf:Description>^^J%  
615 }%  
616 }
```

### 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.

```

617 \gdef\hyxmp@photoshop@schema{%
618   \edef\hyxmp@photoshop@data{\@pdfauthortitle\@pdfcaptionwriter}%
619   \ifx\hyxmp@photoshop@data\@empty
620     \else
621       \hyxmp@add@to@xml{%
622         <rdf:Description rdf:about=""^^J%
623         -----_xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/">^^J%
624       }%
625     \fi
626     \hyxmp@add@simple{photoshop:AuthorsPosition}{\@pdfauthortitle}%
627     \hyxmp@add@simple{photoshop:CaptionWriter}{\@pdfcaptionwriter}%
628     \ifx\hyxmp@photoshop@data\@empty
629       \else
630         \hyxmp@add@to@xml{%
631         -----</rdf:Description>^^J%
632       }%
633     \fi
634 }
```

### 3.5.8 Constructing the XMP packet

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

```

635 \begingroup
636   \ifhyxmp@unicodetex
637     \lccode'\!="FEFF %
638     \lowercase{%
639       \gdef\hyxmp@bom{!}
640     }%
641   \else
642     \catcode'\^^ef=12
643     \catcode'\^^bb=12
644     \catcode'\^^bf=12
645     \gdef\hyxmp@bom{^^ef^^bb^^bf}%
646   \fi
647 \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.

```

648 \def\hyxmp@construct@packet{%
649   \gdef\hyxmp@xml{%
650     \hyxmp@add@to@xml{<?xpacket begin="\hyxmp@bom" %
651     id="W5M0MpCehiHzreSzNTczkc9d"?>^^J%
652     <x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="3.1-702">^^J%
653     ___<rdf:RDF
```

```

654 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\hyxmp@hash">^^J%
655 }%
656 \hyxmp@pdf@schema
657 \hyxmp@xmpRights@schema
658 \hyxmp@dc@schema
659 \hyxmp@photoshop@schema
660 \hyxmp@xmp@basic@schema
661 \hyxmp@mm@schema
662 \hyxmp@add@to@xml{%
663 ___</rdf:RDF>^^J%
664 </x:xmpmeta>^^J%
665 \hyxmp@padding
666 <?xpacket end="w"?>^^J%
667 }%
668 }

```

### 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.
669 \newcommand*{\hyxmp@embed@packet}{%
670 \hyxmp@construct@packet
671 \def\hyxmp@driver{hpdftex}%
672 \ifx\hyxmp@driver\Hy@driver
673 \hyxmp@embed@packet@pdftex
674 \else
675 \def\hyxmp@driver{hdvipdfm}%
676 \ifx\hyxmp@driver\Hy@driver
677 \hyxmp@embed@packet@dvipdfm
678 \else
679 \def\hyxmp@driver{hxdetex}%
680 \ifx\hyxmp@driver\Hy@driver
681 \hyxmp@embed@packet@xdetex
682 \else
683 \@ifundefined{pdfmark}{%
684 \PackageWarningNoLine{hyperxmp}{%
685 Unrecognized hyperref driver ‘\Hy@driver’. \MessageBreak
686 \jobname.tex’s XMP metadata will *not* be \MessageBreak
687 embedded in the resulting file}%
688 }{%
689 \hyxmp@embed@packet@pdfmark
690 }%
691 \fi
692 \fi
693 \fi

```

694 }

### 3.6.1 Embedding using pdfTeX

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

```
695 \newcommand*{\hyxmp@embed@packet@pdftex}{%
696   \bgroup
697     \pdfcompresslevel=0
698     \immediate\pdfobj stream attr {%
699       /Type /Metadata
700       /Subtype /XML
701     }\hyxmp@xml}%
702   \pdfcatalog {\Metadata \the\pdflastobj\space 0 R}%
703   \egroup
704 }
```

### 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.

```
705 \newcommand*{\hyxmp@embed@packet@pdfmark}{%
706   \pdfmark{%
707     pdfmark=/NamespacePush
708   }%
709   \pdfmark{%
710     pdfmark=/OBJ,
711     Raw={/_objdef \string{hyxmp@Metadata\string} /type /stream}%
712   }%
713   \pdfmark{%
714     pdfmark=/PUT,
715     Raw={\string{hyxmp@Metadata\string}
716       2 dict begin
717         /Type /Metadata def
718         /Subtype /XML def
719         currentdict
720       end
721     }%
722   }%
723   \pdfmark{%
724     pdfmark=/PUT,
725     Raw={\string{hyxmp@Metadata\string} (\hyxmp@xml)}%
726   }%
727   \pdfmark{%
728     pdfmark=/Metadata,
729     Raw={\string{Catalog\string} \string{hyxmp@Metadata\string}}%
730   }%
731   \pdfmark{%
732     pdfmark=/NamespacePop
```



```

733 }%
734 }

```

### 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.

```

735 \newcommand*{\hyxmp@embed@packet@dvipdfm}{%
736   \hyxmp@string@len{\hyxmp@xml}%
737   \special{pdf: object @hyxmp@Metadata
738     <<
739       /Type /Metadata
740       /Subtype /XML
741       /Length \the\@tempcnta
742     >>
743   stream^^J\hyxmp@xml endstream%
744 }%
745 \special{pdf: docview
746   <<
747     /Metadata @hyxmp@Metadata
748   >>
749 }%
750 }

```

`\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.

```

751 \newcommand*{\hyxmp@string@len}[1]{%
752   \@tempcnta=0
753   \expandafter\hyxmp@count@spaces#1 {} %
754   \expandafter\hyxmp@count@non@spaces#1{}%
755 }

```

`\hyxmp@count@spaces` Count the number of spaces in a given string. We rely on the built-in pattern matching of T<sub>E</sub>X's `\def` primitive to pry one word at a time off the head of the input string.

```

756 \def\hyxmp@count@spaces#1 {%
757   \def\hyxmp@one@token{#1}%
758   \ifx\hyxmp@one@token\@empty
759     \advance\@tempcnta by -1
760   \else
761     \advance\@tempcnta by 1
762     \expandafter\hyxmp@count@spaces
763   \fi
764 }

```

`\hyxmp@count@non@spaces` Count the number of non-spaces in a given string. Ideally, we'd count both spaces and non-spaces but  $\TeX$  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.

```

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

### 3.6.4 Embedding using $\text{\XeTeX}$

`\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.

```

773 \newcommand*{\hyxmp@embed@packet@xetex}{%
774   \special{pdf:stream @hyxmp@Metadata (\hyxmp+xml)
775     <<
776       /Type /Metadata
777       /Subtype /XML
778     >>
779   }%
780   \special{pdf:put @catalog
781     <<
782       /Metadata @hyxmp@Metadata
783     >>
784   }%
785 }
```

## 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.

```

786 \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 (`pdf $\TeX$` , `Lua $\TeX$` ,  `$\text{\XeTeX}$` , 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](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	backend ( <code>xdvipdfmx</code> ) . . . . .	34
General: Initial version . . . . .	<code>\hyxmp@photoshop@data</code> : Added support for the Photoshop schema . . . . .	30
v1.1	<code>\hyxmp@construct@packet</code> : Explicitly set the category codes of characters <code>&lt;EF&gt;</code> , <code>&lt;BB&gt;</code> , and <code>&lt;BF&gt;</code> to “letter”. Thanks to Daniel Schömer for the bug report . . . . .	30
v1.2	General: Made the package compatible with <code>ngerman</code> . Thanks to Tobias Mueller for the bug report. . . . .	9
	<code>\hyxmp@embed@packet@xetex</code> : Added support for the <code>X<sub>Y</sub>TEX</code>	
	v1.3	
	General: Introduced the <code>pdfmetalang</code> package option, which enables an author to specify the language in which he wrote the document’s metadata	12
	<code>\hyxmp@reencode</code> : 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	16

v1.4			
	\hyxmp@mm@schema: Renamed the xapMM namespace prefix to xmpMM . . . . .	29	
	\hyxmp@rdf@dc: Included metadata in the x-default language regardless of the specified metadata language . . . . .	26	
	\hyxmp@xmpRights@schema: Renamed the xapRights namespace prefix to xmpRights . . . .	28	
v1.5	General: Made the XMP inclusion more robust. Thanks to Heiko Oberdiek for the bug report and suggested modifications. . . . .	9	
v2.0	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 T <sub>E</sub> X implementations (X <sub>Y</sub> T <sub>E</sub> X and LuaT <sub>E</sub> X) . . . .	1	
	New \AtBeginDocument code from Heiko Oberdiek to properly encode \@pdfmetalang . . . . .	12	
	\hyxmp@add@to@xml: Updated also to replace commas . . . . .	24	
	\hyxmp@bom: Added by Heiko Oberdiek . . . . .	30	
	\hyxmp@comma: Added this macro . . . . .	14	
	\hyxmp@construct@packet: Modified by Heiko Oberdiek to use an appropriate BOM representation via \hyxmp@bom . . . . .	30	
	\hyxmp@crap@convert: Added by Heiko Oberdiek . . . . .	20	
	\hyxmp@crap@test: Added by Heiko Oberdiek . . . . .	20	
	\hyxmp@dc@schema: Added support for dc:language and dc:source . . . . .	27	
	\hyxmp@is@unicode: Added by Heiko Oberdiek . . . . .	17	
	\hyxmp@list@to@xml: Modified by Heiko Oberdiek to use the new Unicode-processing macros . . . .	27	
	\hyxmp@photoshop@data: Simplified using \hyxmp@add@simple . . . .	30	
	\hyxmp@ProcessKeyvalOptions: Added this macro . . . . .	10	
	\hyxmp@reencode: Replaced with an empty macro by Heiko Oberdiek . . . . .	16	
	\hyxmp@skiptorelax: Added by Heiko Oberdiek . . . . .	20	
	\hyxmp@skipzeros: Added by Heiko Oberdiek . . . . .	19	
	\hyxmp@SpaceOther: Added by Heiko Oberdiek . . . . .	19	
	\hyxmp@string: Added this macro . . . . .	26	
	\hyxmp@toxml: Added by Heiko Oberdiek . . . . .	18	
	Escaped parentheses written with pdfmarks to prevent dvips from line-wrapping the XMP packet . . . .	18	
	\hyxmp@toxml@unicodetex: Added by Heiko Oberdiek . . . . .	18	
	\hyxmp@xetex@crap: Added by Heiko Oberdiek . . . . .	19	
	\hyxmp@xmlify: Completely rewritten by Heiko Oberdiek to better support Unicode-enabled T <sub>E</sub> X programs . . . . .	16	
	\hyxmp@xmp@basic@schema: Added this macro . . . . .	29	
	\hyxmp@xmpRights@schema: Modified to include xmpRights:Marked only when pdfcopyright is specified and xmpRights:WebStatement only when pdflicenseurl is specified . . . . .	28	
	\hyxmp@zero: Added by Heiko Oberdiek . . . . .	21	
	\ifhyxmp@unicodetex: Added by Heiko Oberdiek . . . . .	15	
	\ProcessKeyvalOptions: Added this macro . . . . .	10	
	\xmpcomma: Added this macro . . . . .	14	
	\xmpquote: Added this macro . . . . .	14	
	\XMPTruncateList: Added this macro . . . . .	14	

# Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in **roman** refer to the code lines where the entry is used.

Symbols	A	G
\! 310, 317, 324, 331, 637	ASCII ..... 17	\g@addto@macro .... 403, 405, 407
\" ..... 1, 2, 786	\AtBeginDocument ... 84	Ghostscript ..... 5
\# ..... 442	\AtEndDocument ..... 5	
\& ..... 209, 241	\AtEndDvi ..... 8	
\( ..... 219, 220	atenddvi ..... 9	
\) ..... 221, 222	Author ..... 5, 7, 14	
\* ..... 235, 248		
\, ..... 437		
\@baseurl ..... 65, 612	<b>B</b>	
\@elt ..... <u>126</u> , <u>531</u>	baseurl (option) .... 3, 29	hyperref ... 1, 3, 4, 7,
\@firstofone ..... 288	BOM ..... 30, 36	9, 10, 12, 13, 31, 32
\@firstoftwo ..... 200		hyperxmp ..... 1–4,
\@gobble ..... 131, 285	<b>C</b>	7–10, 12–14, 16, 21
\@nil ..... 196,	\csname ..... 30, 47, 127, 130, 152	\hyxmp@is@unicode . <u>189</u>
198, 265, 282, 290		\hyxmp@add@simple ..
\@pdfauthor ..... .. 33, 66, 413, 420	<b>D</b>	..... 482,
\@pdfauthortitle ... .. 19, 67, 618, 626	\day .. 421, 463, 464, 466	483, 485, <u>492</u> ,
\@pdfcaptionwriter . .. 21, 68, 618, 627	dc:creator ..... 2, 7, 27	558, 559, 586,
\@pdfcopyright .... .. 15, 69, 554, 570, 580	dc:date ..... 2, 27	608–612, 626, 627
\@pdfcreator ..... 611	dc:description .... 2, 27	\hyxmp@add@to+xml ..
\@pdfkeywords ..... .. 50, 70, 471, 482	dc:format ..... 2	..... <u>428</u> ,
\@pdflang . 71, 86, 89, 558	dc:language .... 2, 27, 36	447, 478, 487,
\@pdflicenseurl ... .. 17, 72, 566, 586	dc:rights ..... 2, 27	497, 506, 512,
\@pdfmetalang .. 23, 87, 89, 91, 510, 513	dc:source .... 2, 27, 36	516, 526, 534,
\@pdfproducer . 472, 483	dc:subject ..... 2, 27	540, 547, 560,
\@pdfsubject ... 73, 553	dc:title ..... 2, 27	576, 582, 587,
\@pdftitle ..... .. 74, 413, 420, 552	\define@key .... 16,	595, 604, 613,
\@secondoftwo ..... 202	18, 20, 22, 24, 33, 50	621, 630, 650, 662
\^ 122, 146, 437, 642–644	dvipdf (option) ..... 32	\hyxmp@append@hex ..
\_ ..... 436	dvipdfm ..... 33	.... <u>375</u> , 394–397
\~ ..... 279, 280	dvips (option) ..... 32	\hyxmp@append@hex@iv
	dvips ..... 5, 18, 36	. <u>393</u> , 401, 402,
	dvipsone (option) .... 32	404, 406, 408–410
	dviwindo (option) ... 32	\hyxmp@at@end .... <u>3</u> , 92
		\hyxmp@big@prime ...
		. <u>349</u> , 352, 362, 372
		\hyxmp@big@prime@ii
		..... <u>349</u> , 371
	<b>E</b>	\hyxmp@bom .... <u>635</u> , 650
	\EdefEscapeHex 168, 181	\hyxmp@comma 34, 51, <u>121</u>
	\EdefUnescapeHex ... 185	\hyxmp@commas@to@list
	\EdefUnescapeString 155	.... <u>105</u> , 128, 532
	\endcsname ..... 30, 47, 127, 130, 152	\hyxmp@commas@to@list@i
	ETX ..... 14, 16	..... 107, 109

\hyxmp@concat@metadata	\hyxmp@modulo@a . . .	\hyxmp@toxml@unicodetex
..... 63	. 343, 362, 372, 378	..... 171, 231
\hyxmp@construct@packet	\hyxmp@num . . . . .	\hyxmp@trimb . . 139, 142
..... 648, 670	\hyxmp@one@token . . .	\hyxmp@trimc . . 142, 143
\hyxmp@count@non@spaces	..... 351, 355,	\hyxmp@trimspaces . .
..... 754, 765	757, 758, 766, 767	..... 114, 135
\hyxmp@count@spaces	\hyxmp@padding 445, 665	\hyxmp@try . . . . .
..... 753, 756	\hyxmp@pdf@schema . .	\hyxmp@unicodetexfalse
\hyxmp@crap@convert	..... 469, 656	..... 145
..... 271, 305	\hyxmp@pdfauthor 33, 555	\hyxmp@unicodetextrue
\hyxmp@crap@result .	\hyxmp@pdfkeywords .	..... 145
..... 261, 297	..... 50, 556	\hyxmp@x@default . .
\hyxmp@crap@test 268, 293	\hyxmp@photoshop@data	. . 87, 468, 510, 517
\hyxmp@create@uuid .	..... 617	\hyxmp@xetex@crap . .
..... 399, 416, 426	\hyxmp@photoshop@schema	..... 162, 261
\hyxmp@dc@schema 546, 658	..... 617, 659	\hyxmp@xml . . . . .
\hyxmp@def@DocumentID	\hyxmp@ProcessKeyvalOptions	..... 438,
..... 412, 593	..... 25	725, 736, 743, 774
\hyxmp@def@InstanceID	\hyxmp@rand@num . . .	\hyxmp@xmlified 153,
..... 418, 594	. 368, 377, 415, 425	498, 513, 517, 532
\hyxmp@DocumentID . .	\hyxmp@rdf@dc . . . .	\hyxmp@xmlify . . 91,
..... 412, 598	..... 502, 552–554	153, 496, 505, 531
\hyxmp@dq@code . . 1, 786	\hyxmp@reencode . . . 151	\hyxmp@xmp@basic@schema
\hyxmp@driver . . . 3, 669	\hyxmp@rights . . . .	..... 603, 660
\hyxmp@embed@packet	. 565, 568, 572, 574	\hyxmp@xmpRights@schema
..... 94, 669	\hyxmp@seed@rng . . .	..... 564, 657
\hyxmp@embed@packet@dvi	..... 351, 414, 424	\hyxmp@zero . . . . 314,
..... 677, 735	\hyxmp@seed@rng@i . .	321, 328, 334, 339
\hyxmp@embed@packet@pdfmark	..... 353, 355	
..... 689, 705	\hyxmp@seed@string .	<b>I</b>
\hyxmp@embed@packet@pdf	. 413, 414, 419, 424	IETF . . . . . 4
..... 673, 695	\hyxmp@set@rand@num	\ifhyxmp@unicodetex
\hyxmp@embed@packet@xetex	..... 368, 376	..... 145, 156, 636
..... 681, 773	\hyxmp@skiptorelax .	ifxetex . . . . . 10
\hyxmp@find@metadata	..... 298, 304	\ifxetex . . . . . 161
..... 63, 93	\hyxmp@skipzeros . . . 256	Info . . . . . 5
\hyxmp@hash . . . 441, 654	\hyxmp@SpaceOther . .	intcalc . . . . . 10
\hyxmp@have@any . . . 470	..... 265, 278	\intcalcDiv 310, 317, 324
\hyxmp@Hyp@pdfauthor 27	\hyxmp@string . . . . 492	\intcalcMod 312, 319, 326
\hyxmp@Hyp@pdfkeywords	\hyxmp@string@len . .	ISO . . . . . 10
..... 44	..... 736, 751	
\hyxmp@InstanceID . .	\hyxmp@sublist . . . .	<b>J</b>
..... 418, 599	. 110, 111, 114, 115	\jobname . . . . 78, 98,
\hyxmp@is@unicode . .	\hyxmp@temp@list . . . 126	413, 420, 559, 686
..... 157, 174, 189	\hyxmp@temp@str . . . 126	
\hyxmp@legal . . . . . 565	\hyxmp@text . . . . .	<b>K</b>
\hyxmp@list . . . 532, 538	. 153, 231, 261, 305	Keywords . . . . . 5
\hyxmp@list@to@xml .	\hyxmp@today . . . . .	\KV@Hyp@pdfauthor . . 33
..... 523, 555–557	. 457, 557, 608–610	\KV@Hyp@pdfkeywords 50
\hyxmp@mm@schema 592, 661	\hyxmp@toxml . . 183, 206	kvoptions . . . . . 10

<b>L</b>		<code>\pdflastobj</code> . . . . . 702	<code>Title</code> . . . . . 5
<code>\lccode</code> . . . . . 190,		<code>pdfL<sup>A</sup>T<sub>E</sub>X</code> . . . . . 3, 5	
191, 280, 310,		<code>pdflicenseurl</code> (option) .	<b>U</b>
317, 324, 331,		. . . . 3, 4, 9, 28, 36	<code>\uccode</code> . . . . . 235, 248
432, 436, 437, 637	<code>pdfmark</code> (option) . . . . 32		<code>Unicode</code> . . . . . 10,
<code>\lowercase</code> . . . . . 194,	<code>\pdfmark</code> . . 706, 709,		15–20, 27, 30, 34, 36
281, 311, 318,	713, 723, 727, 731	<code>pdfmetalang</code> (option) .	<code>\uppercase</code> . . . . 236, 249
325, 332, 438, 638	. . . . . 3, 4, 9		<code>URL</code> . . . . . 2, 4, 10, 28, 29
<code>LuaL<sup>A</sup>T<sub>E</sub>X</code> . . . . . 5, 7	<code>\pdfminorversion</code> . . . 485	<code>\usepackage</code> . . . . . 99	<code>UTF-16BE</code> . . . . . 17
<code>LuaT<sub>E</sub>X</code> . . 15, 18, 34, 36	<code>\pdfobj</code> . . . . . 698	<code>UTF-16BE</code> . . . . . 16, 17	<code>UTF-8</code> . . . . . 17
<b>M</b>		<code>pdfproducer</code> (option) . . 3	<code>UUID</code> . . . . . 21, 23
<code>Metadata</code> . . . . . 5, 31, 34	<code>\pdfstringdef</code> . . . . .		
<code>\month</code> 421, 458, 459, 461	. . 16, 18, 20, 22, 24	<b>V</b>	
<b>N</b>		<code>pdfsubject</code> (option) 3, 9, 27	<code>\vfuzz</code> . . . . . 143
<code>nativepdf</code> (option) . . . 32	<code>pdfT<sub>E</sub>X</code> . . . 9, 18, 32, 34	<code>vtexpdfmark</code> (option) . 32	
<code>\newif</code> . . . . . 145	<code>pdftitle</code> (option) . 3, 9, 27		
<code>\next</code> . . . . . 109, 355	<code>photoshop:AuthorsPosition</code>	<b>X</b>	
<code>ngerman</code> . . . . . 9, 35	. . . . . 2, 30	<code>\x</code> . . . . . 261	
<code>\number</code> 308, 310, 312,	<code>photoshop:CaptionWriter</code>	<code>xdvipdfmx</code> . . . . . 7, 34	
317, 319, 324, 326	. . . . . 2, 30	<code>X<sub>l</sub>L<sup>A</sup>T<sub>E</sub>X</code> . . . . . 5, 7	
<b>P</b>		<code>X<sub>g</sub>L<sup>A</sup>T<sub>E</sub>X</code> 10, 15, 18, 19, 34–36	
<code>\PackageWarningNoLine</code>	<code>PI</code> . . . . . 24	<code>XML</code> . . . . . 1, 2, 13,	
. . . . . 77, 97, 684	<code>\ProcessKeyvalOptions</code>	15, 17, 18, 24–27, 30	
<code>PDF</code> . . . . . 1–5, 7, 13,	. . . . . 25	<code>XMP</code> . . . . . 1–3,	
18, 23–25, 30, 31, 34	<code>ps2pdf</code> (option) . . . . 32	5, 7–9, 13–15, 18,	
<code>pdf:Keywords</code> . . . . . 2, 25	<b>Q</b>	21, 24, 26, 29, 32–36	
<code>pdf:PDFVersion</code> . . . . . 2	<code>\Q</code> . . . . . 135, 144	<code>XMP</code> . . . . . 36	
<code>pdf:Producer</code> . . . . . 2, 25	<b>R</b>	<code>xmp:BaseURL</code> . . . . . 2	
<code>pdfauthor</code> (option) . . .	<code>rdf:li</code> . . . . . 2	<code>xmp:CreateDate</code> . . . . . 2	
. . . . . 3, 8–10, 27	<code>rdf:Seq</code> . . . . . 2	<code>xmp:CreatorTool</code> . . . . . 2	
<code>pdfauthor:Title</code> (option)	<code>\renewcommand</code> . . . . 26	<code>xmp:MetadataDate</code> . . . . 2	
. . . . . 3, 4, 9	<code>\RequirePackage</code> 7, 10–14	<code>xmp:ModifyDate</code> . . . . . 2	
<code>pdfcaptionwriter</code> (op-	<b>S</b>	<code>\xmpcomma</code> . . . 33, 50, 120	
tion) . . . . . 3, 4, 9	<code>\SE-&gt;pdfdoc@03</code> . . . . 152	<code>xmpincl</code> . . . . . 3	
<code>\pdfcatalog</code> . . . . . 702	<code>\special</code> 737, 745, 774, 780	<code>xmpMM:DocumentID</code> .	
<code>\pdfcompresslevel</code> . . 697	<code>stringenc</code> . . . . . 10	. . . . . 2, 21, 29	
<code>pdfcopyright</code> (option) .	<code>\StringEncodingConvert</code>	<code>xmpMM:InstanceID</code> . .	
. . 3, 4, 9, 27, 28, 36	. . . . . 158,	. . . . . 2, 21, 29	
<code>PDFDocEncoding</code> . . .	164, 175, 178, 273	<code>\xmpquote</code> . . . 33, 50, 125	
. . . . . 10, 16, 17	<code>Subject</code> . . . . . 5	<code>xmpRights:Marked</code> 2, 28, 36	
<code>pdfescape</code> . . . . . 10	<b>T</b>	<code>xmpRights:WebStatement</code>	
<code>pdfkeywords</code> (option) .	<code>T<sub>E</sub>X</code> . . . . . 16,	. . . . . 2, 28, 36	
. . . . . 3, 8–10, 27	18, 21, 29, 33, 34, 36	<code>\XMPTruncateList</code> . . 126	
<code>pdflang</code> (option) . . . .	<code>textures</code> (option) . . . . 32	<b>Y</b>	
. . . . 3, 4, 9, 12, 27	<code>\time</code> . . . . . 422	<code>\year</code> . . . . . 421, 457	