

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

Scott Pakin  
scott+hyxmp@pakin.org

June 22, 2014

## 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>
```

---

<sup>\*</sup>This document corresponds to `hyperxmp` v2.5, dated 2014/06/19.

```

    </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`)
- contact address (`lptc4xmpCore:CreatorContactInfo/CiAdrExtadr`,  
`lptc4xmpCore:CreatorContactInfo/CiAdrCity`,  
`lptc4xmpCore:CreatorContactInfo/CiAdrRegion`,  
`lptc4xmpCore:CreatorContactInfo/CiAdrPcode`, and  
`lptc4xmpCore:CreatorContactInfo/CiAdrCtry`)
- contact email address(es) (`lptc4xmpCore:CreatorContactInfo/CiEmailWork`)
- contact telephone number(s) (`lptc4xmpCore:CreatorContactInfo/CiTelWork`)
- contact URL(s) (`lptc4xmpCore:CreatorContactInfo/CiUrlWork`)
- 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 version (`pdf:PDFVersion`)
- PDF-generating tool (`pdf:Producer` and `xmp:CreatorTool`)
- PDF/A compliance level and version (`pdfaid:part` and `pdfaid:conformance`)
- 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 `pdfLATEX` 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`
- `pdfcontactaddress`
- `pdfcontactcity`
- `pdfcontactcountry`
- `pdfcontactemail`
- `pdfcontactphone`
- `pdfcontactpostcode`
- `pdfcontactregion`
- `pdfcontacturl`
- `pdfcopyright`
- `pdflicenseurl`
- `pdfmetalang`

`pdfauthortitle` indicates the primary author's position or title. `pdfcaptionwriter` specifies the name of the person who added the metadata to the document. The next eight items describe how to contact the person or institution responsible for the document (the “contact”). `pdfcontactaddress` is the contact's street address and can include the institution name if the contact is an institution; `pdfcontactcity` is the contact's city. `pdfcontactcountry` is the contact's country; `pdfcontactemail` is the contact's email address (or multiple, comma-separated email addresses); `pdfcontactphone` is the contact's telephone number (or multiple, comma-separated telephone numbers); `pdfcontactpostcode` is the contact's postal code; `pdfcontactregion` is the contact's state or province; and `pdfcontacturl` is the contact's URL (or multiple, comma-separated URLs).

`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 [7], 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 `LATEX` 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},
  pdfcontactaddress={Kramgasse 49},
  pdfcontactcity={Bern},
  pdfcontactpostcode={3011},
  pdfcontactcountry={Switzerland},
  pdfcontactphone={031 312 00 91},
  pdfcontactemail={aeinstein@ipi.ch},
  pdfcontacturl={%
    http://einstein.biz/,
    https://www.facebook.com/AlbertEinstein
  },
  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:

- `pdfLATEX`

- 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 that shown in Appendix A. Figure 1 is a screenshot of the XMP metadata as it appears in Adobe Acrobat's "Advanced" metadata dialog box. Further clicking on the "Advanced" item within that dialog box displays all of the document's metadata sorted by schema as shown in Figure 2.

**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: Acrobat multiline-field bug** The IPTC Photo Metadata schema states that "the [contact] address is a multiline field" [6]. `hyperxmp` converts commas in `pdfcontactaddress`'s argument to line breaks in the generated XML. Unfortunately, A bug in Adobe Acrobat—at least in versions 10.0.1 and earlier—causes that PDF reader to discard line breaks in the contact address. Interestingly,

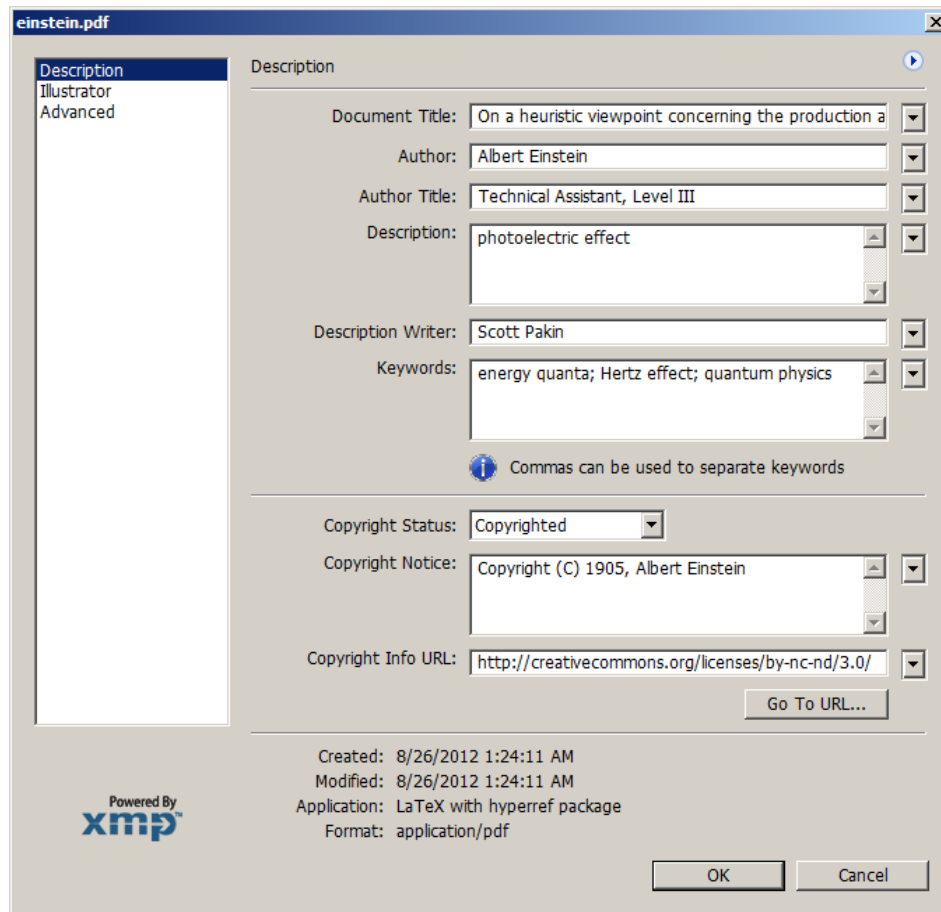


Figure 1: XMP metadata as it appears in Adobe Acrobat

`\xmplinesep` Adobe Illustrator CS5 correctly displays the contact address. If you find Adobe Acrobat's behavior bothersome, you can redefine the `\xmplinesep` macro as a string to use as an address-line separator. For example, the following replaces all commas appearing in `pdfcontactaddress`'s argument with semicolons:

```
\renewcommand*{\xmplinesep}{;}
```

**Note 3: X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X object compression** X<sub>q</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>q</sub>L<sup>A</sup>T<sub>E</sub>X to instruct `xdvipdfmx` to

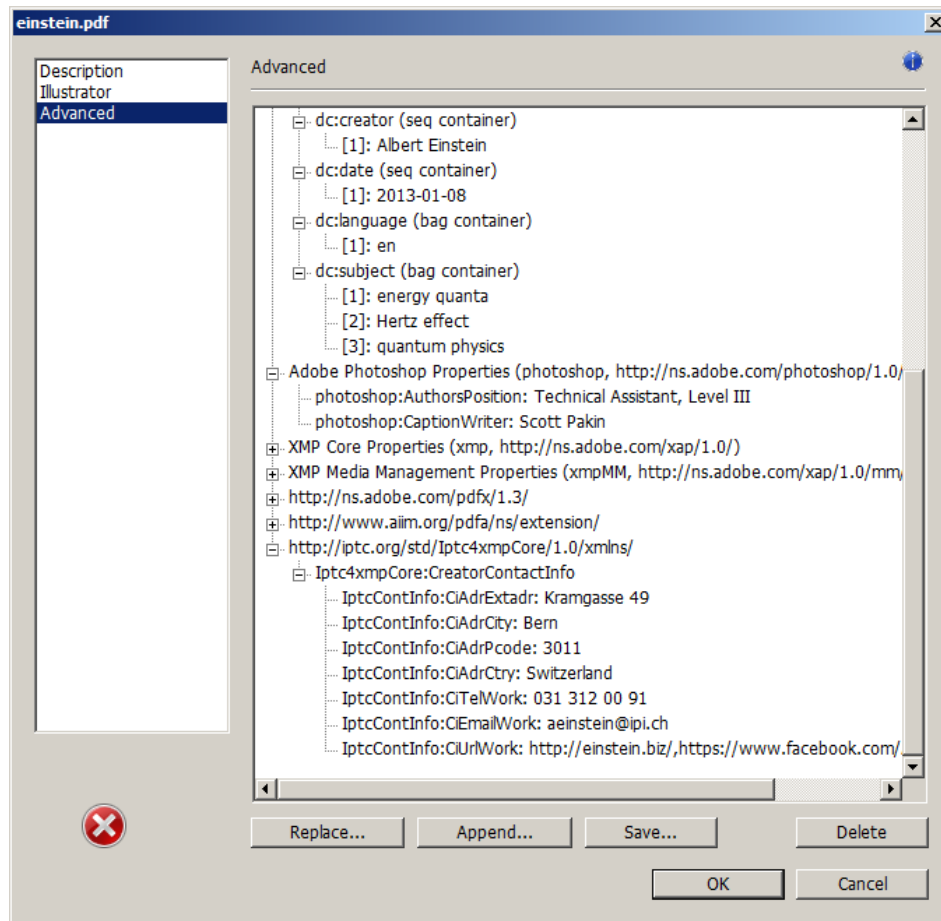


Figure 2: Additional XMP metadata as it appears in Adobe Acrobat

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.

**Note 4: 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 literal comma within an author or keyword list (where  
`\xmpquote` commas normally separate list items) or a street address (where commas normally  
separate lines), 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.}}

pdfcontactaddress={Office of the President,
                   \xmpquote{Wayne Enterprises\xmpcomma\ Inc.},
                   One Wayne Blvd}
```

As of version 2.2 of `hyperxmp`, it is acceptable to use `\xmpcomma` and `\xmpquote` within any `hyperxmp` option, not just in those in which a comma normally serves as a separator (i.e., lists and multiline fields). Outside of cases in which a comma serves as a separator, `\xmpcomma` is treated as an ordinary comma, and `\xmpquote` returns its argument unmodified. Hence, it is legitimate to use `\xmpcomma` and `\xmpquote` in cases like the following

```
pdfauthortitle={\xmpquote{Psychiatrist\xmpcomma\ Arkham Asylum}}
```

(Like most `hyperxmp` options, `pdfauthortitle` inserts its argument unmodified in an XMP tag.) When in doubt, use `\xmpcomma` and `\xmpquote`; it should always be safe to do so.

`\xmptilde` Version 2.4 of `hyperxmp` introduces a convenience macro called `\xmptilde`. `\xmptilde` expands to a literal tilde character instead of the nonbreaking space that “~” normally represents. Use it to represent URLs such as `http://www.pakin.org/~scott/` (“`http://www.pakin.org/\xmptilde scott/`”) in options such as `baseurl`, `pdfcontacturl` and `pdflicenseurl`.

**Note 5: Unicode support** Unicode support is provided via the `hyperref` package. If you specify `unicode=true` either as a `hyperref` option or as an argument to the `\hypersetup` command, the document can include Unicode characters in its XMP fields.

## 3 Implementation

This section presents the commented  $\text{\LaTeX}$  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 pdfTeX, `\hyxmp@driver` 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.

```
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 `baseurl`, `pdfauthor`, `pdfkeywords`, `pdflang`, `pdfproducer`, `pdfsubject`, and `pdftitle` options. It also introduces a number of new options, which are listed on page 4. 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 X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X.

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

`\hyxmp@pdfstringdef` Because `hyperxmp` uses underscores to represent hard spaces, we need “\\_” to map `\hyxmp@textunderscore` initially to something other than an underscore, in particular the ASCII NAK (<sup>^</sup>U) character. To accomplish this, we wrap `hyperref`’s `\pdfstringdef` macro with our own version that temporarily does the proper substitution. Later in the execution, after underscores have been replaced with spaces, we replace NAK characters with underscores.

```
15 \newcommand{\hyxmp@pdfstringdef}[2]{%
```

```

16 \let\hyxmp@textunderscore=\textunderscore
17 \let\textunderscore=\hyxmp@uscore
18 \pdfstringdef{#1}{#2}%
19 \let\textunderscore=\hyxmp@textunderscore
20 }

```

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

```

21 \def\@pdfcopyright{}
22 \define@key{Hyp}{pdfcopyright}{\hyxmp@pdfstringdef\@pdfcopyright{#1}}

```

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

```

23 \def\@pdflicenseurl{}
24 \define@key{Hyp}{pdflicenseurl}{\hyxmp@pdfstringdef\@pdflicenseurl{#1}}

```

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

```

25 \def\@pdfauthortitle{}
26 \define@key{Hyp}{pdfauthortitle}{\hyxmp@pdfstringdef\@pdfauthortitle{#1}}

```

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

```

27 \def\@pdfcaptionwriter{}
28 \define@key{Hyp}{pdfcaptionwriter}{\hyxmp@pdfstringdef\@pdfcaptionwriter{#1}}

```

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

```

29 \def\@pdfmetalang{}
30 \define@key{Hyp}{pdfmetalang}{\hyxmp@pdfstringdef\@pdfmetalang{#1}}

```

The following eight macros—`\@pdfcontactaddress`, `\@pdfcontactcity`, `\@pdfcontactregion`, `\@pdfcontactpostcode`, `\@pdfcontactcountry`, `\@pdfcontactphone`, `\@pdfcontactemail`, and `\@pdfcontacturl`—together specify how to contact the person or institution responsible for the document.

`\@pdfcontactaddress` Prepare to store a street address for the document’s contact person/institution. The IPTC standard defines this as follows:

The contact information address part. Comprises an optional company name and all required information to locate the building or postbox to which mail should be sent. To that end, the address is a multiline field.

For consistency with the rest of hyperxmp, we use commas to separate terms, in this case, lines of the address. The author can use `\xmpquote` and `\xmpcomma` to include literal commas.

```

31 \def\@pdfcontactaddress{}
32 \define@key{Hyp}{pdfcontactaddress}{%
33 \let\xmpcomma=\hyxmp@comma
34 \def\xmpquote##1{##1}%
35 \hyxmp@pdfstringdef\@pdfcontactaddress{#1}%
36 \def\xmpcomma{,}%
37 \let\xmpquote=\relax
38 }

```

<code>\@pdfcontactcity</code>	Prepare to store the city of the document's contact person/institution. 39 <code>\def\@pdfcontactcity{}</code> 40 <code>\define@key{Hyp}{pdfcontactcity}{\hyxmp@pdfstringdef\@pdfcontactcity{#1}}</code>
<code>\@pdfcontactregion</code>	Prepare to store the state or province of the document's contact person/institution. 41 <code>\def\@pdfcontactregion{}</code> 42 <code>\define@key{Hyp}{pdfcontactregion}{\hyxmp@pdfstringdef\@pdfcontactregion{#1}}</code>
<code>\@pdfcontactpostcode</code>	Prepare to store the postal code of the document's contact person/institution. 43 <code>\def\@pdfcontactpostcode{}</code> 44 <code>\define@key{Hyp}{pdfcontactpostcode}{\hyxmp@pdfstringdef\@pdfcontactpostcode{#1}}</code>
<code>\@pdfcontactcountry</code>	Prepare to store the country of the document's contact person/institution. 45 <code>\def\@pdfcontactcountry{}</code> 46 <code>\define@key{Hyp}{pdfcontactcountry}{\hyxmp@pdfstringdef\@pdfcontactcountry{#1}}</code>
<code>\@pdfcontactphone</code>	Prepare to store the telephone number of the document's contact person/institution. 47 <code>\def\@pdfcontactphone{}</code> 48 <code>\define@key{Hyp}{pdfcontactphone}{\hyxmp@pdfstringdef\@pdfcontactphone{#1}}</code>
<code>\@pdfcontactemail</code>	Prepare to store the email address of the document's contact person/institution. 49 <code>\def\@pdfcontactemail{}</code> 50 <code>\define@key{Hyp}{pdfcontactemail}{\hyxmp@pdfstringdef\@pdfcontactemail{#1}}</code>
<code>\@pdfcontacturl</code>	Prepare to store the URL of the document's contact person/institution. 51 <code>\def\@pdfcontacturl{}</code> 52 <code>\define@key{Hyp}{pdfcontacturl}{\hyxmp@pdfstringdef\@pdfcontacturl{#1}}</code>

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`.

<code>\hyxmp@pdfauthor</code>	Prepare to store the name of the author and a list of keywords.
<code>\hyxmp@pdfkeywords</code>	53 <code>\def\hyxmp@pdfauthor{}</code> 54 <code>\def\hyxmp@pdfkeywords{}</code>

\hyxmp@redefine@Hyp	<p>If not already redefined, redefine <code>hyperref's pdfauthor</code> and <code>pdfkeywords</code> options to properly handle <code>\xmpcomma</code> and <code>\xmpquote</code>.</p> <pre>55 \newcommand*{\hyxmp@redefine@Hyp}{%</pre>
\hyxmp@Hyp@pdfauthor	<p>Store the old definition of <code>\KV@Hyp@pdfauthor</code> in <code>\hyxmp@Hyp@pdfauthor</code>, but only if we see that <code>\KV@Hyp@pdfauthor</code> is defined and <code>\hyxmp@Hyp@pdfauthor</code> isn't. Otherwise, we'd be defining <code>\hyxmp@Hyp@pdfauthor</code> in terms of itself and creating an infinite loop.</p> <pre>56 \ifundefined{KV@Hyp@pdfauthor}{}% 57 \ifundefined{hyxmp@Hyp@pdfauthor}{% 58 \expandafter\let\expandafter\hyxmp@Hyp@pdfauthor 59 \csname KV@Hyp@pdfauthor\endcsname 60 }{}% 61 }%</pre>
\KV@Hyp@pdfauthor	<p>Redefine <code>\KV@Hyp@pdfauthor</code> to process its argument twice. The first time, <code>\xmpcomma</code> is defined as a placeholder character (<code>\hyxmp@comma</code>) and <code>\xmpquote</code> as the identity function. The result is stored in <code>\hyxmp@pdfauthor</code> for use in structured lists (those surrounding each entry with <code>&lt;rdf:li&gt;</code>). The second time, <code>\xmpcomma</code> is defined as an ordinary comma, and <code>\xmpquote</code> is defined as a macro that puts its argument within double quotes. The result is stored in <code>\@pdfauthor</code> for use in unstructured lists (those in which the entire list appears within a single pair of tags).</p> <pre>62 \define@key{Hyp}{pdfauthor}{% 63 \let\xmpcomma=\hyxmp@comma 64 \def\xmpquote####1{####1}% 65 \hyxmp@Hyp@pdfauthor{##1}% 66 \global\let\hyxmp@pdfauthor=\@pdfauthor 67 \def\xmpcomma{,%} 68 \def\xmpquote####1{"####1"%} 69 \hyxmp@Hyp@pdfauthor{##1}% 70 \def\xmpcomma{,%} 71 \let\xmpquote=\relax 72 }%</pre>
\hyxmp@Hyp@pdfkeywords	<p>The previous block of code now repeats for the keyword list, starting by storing the old definition of <code>\KV@Hyp@pdfkeywords</code> in <code>\hyxmp@Hyp@pdfkeywords</code>.</p> <pre>73 \ifundefined{KV@Hyp@pdfkeywords}{}% 74 \ifundefined{hyxmp@Hyp@pdfkeywords}{% 75 \expandafter\let\expandafter\hyxmp@Hyp@pdfkeywords 76 \csname KV@Hyp@pdfkeywords\endcsname 77 }{}% 78 }%</pre>
\KV@Hyp@pdfkeywords	<p>Redefine <code>\KV@Hyp@pdfkeywords</code> to process its argument twice. The first time, <code>\xmpcomma</code> is defined as a placeholder character (<code>\hyxmp@comma</code>) and <code>\xmpquote</code> as the identity function. The result is stored in <code>\hyxmp@pdfkeywords</code> for use in structured lists (those surrounding each entry with <code>&lt;rdf:li&gt;</code>). The second</p> <pre>\xmpcomma \xmpquote \hyxmp@pdfkeywords \@pdfkeywords</pre>

time, `\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).

```

79 \define@key{Hyp}{pdfkeywords}{%
80   \let\xmpcomma=\hyxmp@comma
81   \def\xmpquote####1{####1}%
82   \hyxmp@Hyp@pdfkeywords{##1}%
83   \global\let\hyxmp@pdfkeywords=\@pdfkeywords
84   \def\xmpcomma{,%}
85   \def\xmpquote####1{"####1"%}
86   \hyxmp@Hyp@pdfkeywords{##1}%
87   \def\xmpcomma{,%}
88   \let\xmpquote=\relax
89 }%
90 }

```

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

```

91 \let\hyxmp@ProcessKeyvalOptions=\ProcessKeyvalOptions
92 \renewcommand*{\ProcessKeyvalOptions}{%
93   \hyxmp@redefine@Hyp
94   \hyxmp@ProcessKeyvalOptions
95 }

```

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

```

96 \let\hyxmp@hypersetup=\hypersetup
97 \def\hypersetup{%
98   \hyxmp@redefine@Hyp
99   \hyxmp@hypersetup
100 }

```

`\hyxmp@find@metadata` `\hyxmp@concat@metadata` Issue a warning message if the author failed to specify any metadata at all. This excludes metadata that is included automatically such as the current timestamp. Note that we don't consider `\@pdfmetlang` as metadata as that value is meaningful only when used in conjunction with other information.

```

101 \newcommand*{\hyxmp@find@metadata}{%
102   \edef\hyxmp@concat@metadata{%
103     \@baseurl
104     \@pdfauthor
105     \@pdfauthortitle
106     \@pdfcaptionwriter
107     \@pdfcontactaddress
108     \@pdfcontactcity
109     \@pdfcontactcountry
110     \@pdfcontactemail
111     \@pdfcontactphone

```

```

112 \pdfcontactpostcode
113 \pdfcontactregion
114 \pdfcontacturl
115 \pdfcopyright
116 \pdfkeywords
117 \pdflang
118 \pdflicenseurl
119 \pdfsubject
120 \pdftitle
121 }%
122 \ifx\hyxmp@concat@metadata\empty
123 \PackageWarningNoLine{hyperxmp}{%
124 \jobname.tex did not specify any metadata to\MessageBreak
125 include in the XMP packet.\space\space Please see the\MessageBreak
126 hyperxmp documentation for instructions on how to\MessageBreak
127 provide metadata values to hyperxmp}%
128 \fi
129 }

```

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.

```

130 \AtBeginDocument{%
131 \ifpackageloaded{hyperref}{%

```

In older versions of `hyperref`, `\pdflang` is set to `\empty` if `pdflang` is not specified. In newer versions of `hyperref`, `\pdflang` is set to `\relax` if `pdflang` is not specified. The latter is a bit problematic for `hyperxmp` because it makes `\pdflang` non-expandable, which causes a literal “`\pdflang`” to be written as XMP metadata. To avoid that situation we redefine `\pdflang` as `\empty` if we see it set to `\relax`.

```

132 \ifx\pdflang\relax
133 \let\pdflang=\empty
134 \fi

```

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.

```

135 \ifx\pdflang\empty
136 \let\pdfmetalang=\hyxmp@x@default
137 \else
138 \edef\pdfmetalang{\pdflang}%
139 \fi
140 \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`.

```

141 \hyxmp@at@end{%

```

```

142         \hyxmp@find@metadata
143         \hyxmp@embed@packet
144     }%
145 }%
146 {\PackageWarningNoLine{hyperxmp}{%
147 \jobname.tex failed to include a\MessageBreak
148 \string\usepackage\string{hyperref\string}
149 in the preamble.\MessageBreak
150 Consequently, all hyperxmp functionality will be\MessageBreak
151 disabled}%
152 }%
153 }

```

### 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.)
154 \newcommand*{\hyxmp@commas@to@list}[2]{%
155   \gdef#1{%
156     \expandafter\hyxmp@commas@to@list@i\expandafter#1#2,,%
157 }

\hyxmp@commas@to@list@i Recursively construct macro #1 from comma-separated list #2. Stop if #2 is empty.
\next
158 \def\hyxmp@commas@to@list@i#1#2,{%
159   \gdef\hyxmp@sublist{#2}%
160   \ifx\hyxmp@sublist\@empty
161     \let\next=\relax
162   \else
163     \hyxmp@trimspaces\hyxmp@sublist
164     \@cons{#1}{\hyxmp@sublist}%
165     \def\next{\hyxmp@commas@to@list@i{#1}}%
166   \fi
167   \next
168 }

```



`\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 a comma so it can be used in *any* `hyperxmp` option, not just those that treat commas specially.

```
169 \def\xmpcomma{,}%
```

`\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.

```
170 \bgroup
171 \catcode'\^^C=11
172 \gdef\hyxmp@comma{^^C}
173 \egroup
```

`\hyxmp@uscore` This is what `\_` temporarily maps to during packet construction. Because underscores are replaced by spaces, we need a mechanism to preserve user-specified underscores (e.g., in email addresses). We assume that documents will never otherwise use an NAK (`^^U`) character in their XMP metadata.

```
174 \bgroup
175 \catcode'\^^U=11
176 \gdef\hyxmp@uscore{^^U}
177 \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.

```
178 \let\xmpquote=\relax
```

`\xmptilde` As a convenience for the user, we define `\xmptilde` as a category 12 (other) “~” character.

```
179 \bgroup
180 \catcode'\~=12%
181 \gdef\xmptilde{~}%
182 \egroup
```

`\XMPTruncateList` As a workaround for Adobe Acrobat's inability to display author lists correctly (cf. “Acrobat Author bug” on page 6) 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.

```
183 \newcommand{\XMPTruncateList}[1]{%
184 \edef\hyxmp@temp@str{\csname hyxmp@#1\endcsname}%
```

```

185 \hyxmp@commas@to@list{\hyxmp@temp@list}{\hyxmp@temp@str}%
186 \def\@elt##1{%
187   \expandafter\gdef\csname @#1\endcsname{##1}%
188   \let\@elt=\@gobble
189 }
190 \hyxmp@temp@list
191 }}

```

### 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.

```

192 \catcode'\Q=3
\hyxmp@trimspaces\x redefines \x to have the same replacement text sans leading
and trailing space tokens.
193 \newcommand{\hyxmp@trimspaces}[1]{%
  Use grouping to emulate a multi-token afterassignment queue.
194 \begingroup
  Put “\toks 0 {” into the afterassignment queue.
195 \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.
196 \expandafter\hyxmp@trimb\expandafter\noexpand#1Q Q}%
  Transfer the trimmed text back into #1.
197 \edef#1{\the\toks0}%
198 }

```

`\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.

```

199 \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.

```

200 \def\hyxmp@trimc#1Q#2{\afterassignment\endgroup \vfuzz\the\vfuzz#1}
201 \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.

```

\ifhymp@unicodetex XeΛTeX and LuaTeX natively support Unicode. We define the conditional
\hymp@unicodetexttrue \ifhymp@unicodetex to check for these so we can properly handle encoding
\hymp@unicodetextfalse 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.

202 \newif\ifhymp@unicodetex
203 \ifnum64='\^^^0040\relax
204   \hymp@unicodetexttrue
205 \else
206   \hymp@unicodetextfalse
207 \fi

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

208 \newcommand*{\hymp@reencode}[1]{%

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

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

\SE->pdfdoc@15 Preserve NAK (^U), which is normally an invalid character in PDFDocEncoding.
We use it in hyperxmp (and specifically in \hymp@xmlify below) as a placeholder
for an underscore character.

210 \expandafter\def\csname SE->pdfdoc@15\endcsname{0015}

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

211 \newcommand*{\hymp@xmlify}[1]{%
212   \gdef\hymp@xmlified{%
Escaped PDF string → PDFDocEncoding/Unicode
213   \EdefUnescapeString\hymp@text{#1}%
214   \ifhymp@unicodetex
PDFDocEncoding/Unicode → UTF-32BE
215     \hymp@is@unicode\hymp@text{%
216       \StringEncodingConvert
217       \hymp@text\hymp@text{utf16be}{utf32be}%
218     }{%

```

```

219     \ifxetex
220     \hyxmp@xetex@crap
221     \else
222     \StringEncodingConvert
223     \hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
224     \fi
225     }%

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

UTF-32BE → XML in ASCII
227     \edef\hyxmp@text{%
228     \expandafter
229     }\expandafter\hyxmp@toxml@unicodetex\hyxmp@text
230     \relax\relax\relax\relax\relax\relax\relax\relax
231     \else

PDFDocEncoding/Unicode → UTF-8
232     \hyxmp@is@unicode\hyxmp@text{%
233     \StringEncodingConvert
234     \hyxmp@text\hyxmp@text{utf16be}{utf8}%
235     }{%
236     \StringEncodingConvert
237     \hyxmp@text\hyxmp@text{pdfdoc}{utf8}%
238     }%

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

UTF-8 as hex string → XML in UTF-8 as hex string
240     \edef\hyxmp@text{%
241     \expandafter\hyxmp@toxml\hyxmp@text\@empty\@empty
242     }%

XML in UTF-8 as hex string → XML in UTF-8
243     \EdefUnescapeHex\hyxmp@text{\hyxmp@text}%
244     \fi
245     \global\let\hyxmp@xmlified\hyxmp@text
246 }

```

`\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.

```

247 \begingroup
248 \lccode'\<=254 %
249 \lccode'\>=255 %
250 \catcode254=12 %
251 \catcode255=12 %
252 \lowercase{\endgroup
253 \def\hyxmp@is@unicode#1{%
254     \expandafter\hyxmp@@is@unicode#1<>\@nil
255     }%

```

```

256 \def\hyxmp@is@unicode#1<>#2\@nil{%
257   \ifx\#1\%
258     \expandafter\@firstoftwo
259   \else
260     \expandafter\@secondoftwo
261   \fi
262 }%
263 }

```

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

```

264 \def\hyxmp@toxml#1#2{%
265   \ifx#1\empty
266   \else
267     \ifnum"#1#2='\& %
268       26616D703B% &amp;
269     \else\ifnum"#1#2='\< %
270       266C743B% &lt;
271     \else\ifnum"#1#2='\> %
272       2667743B% &gt;
273   \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.

```

274   \@ifundefined{pdfmark}{%
275     #1#2%
276   }{%
277     \ifnum"#1#2='\( %
278       5C28% \(
279     \else\ifnum"#1#2='\) %
280       5C29% \)
281     \else
282       #1#2%
283     \fi\fi
284   }%
285   \fi\fi\fi
286   \expandafter\hyxmp@toxml
287 \fi
288 }

```

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

```

289 \def\hyxmp@toxml@unicodetex#1#2#3#4#5#6#7#8{%
290   \ifx#1\relax
291   \else
292     \ifnum"#1#2#3#4#5#6#7#8>127 %
293       \uccode'\*="#1#2#3#4#5#6#7#8\relax
294       \uppercase{%
295         \edef\hyxmp@text{\hyxmp@text *}%
296       }%
297     \else\ifnum"#7#8='< %
298       \edef\hyxmp@text{\hyxmp@text &lt;}%
299     \else\ifnum"#7#8='& %
300       \edef\hyxmp@text{\hyxmp@text &amp;}%
301     \else\ifnum"#7#8='> %
302       \edef\hyxmp@text{\hyxmp@text &gt;}%
303     \else\ifnum"#7#8='\ %
304       \edef\hyxmp@text{\hyxmp@text\space}%
305     \else
306       \uccode'\*="#7#8\relax
307       \uppercase{%
308         \edef\hyxmp@text{\hyxmp@text *}%
309       }%
310     \fi\fi\fi\fi\fi
311     \expandafter\hyxmp@toxml@unicodetex
312   \fi
313 }
```

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

```

314 \def\hyxmp@skipzeros#1{%
315   \ifx#10%
316     \expandafter\hyxmp@skipzeros
317   \fi
318 }
```

`\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 319 \begingroup
\hyxmp@crap@result 320 \def\x#1{\endgroup
\hyxmp@text 321   \def\hyxmp@xetex@crap{%
322     \edef\hyxmp@try{%
323       \expandafter\hyxmp@SpaceOther\hyxmp@text#1@nil
324     }%
325     \let\hyxmp@crap@result=N%
326     \expandafter\hyxmp@crap@test\hyxmp@try\relax
327     \ifx\hyxmp@crap@result Y%
328       \let\hyxmp@text\@empty
329       \expandafter\hyxmp@crap@convert\hyxmp@try\relax
330     \else
```

```

331      \StringEncodingConvert\hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
332      \fi
333    }%
334  }
335  \x{ }

```

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

```

336 \begingroup
337   \catcode'\~=12 %
338   \lccode'\~=\' %
339 \lowercase{\endgroup
340   \def\hyxmp@SpaceOther#1 #2\@nil{%
341     #1%
342     \ifx\relax#2\relax
343       \expandafter\@gobble
344     \else
345       ~%
346       \expandafter\@firstofone
347     \fi
348     {\hyxmp@SpaceOther#2\@nil}%
349   }%
350 }

```

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

```

351 \def\hyxmp@crap@test#1{%
352   \ifx#1\relax
353   \else
354     \ifnum'#1>127 %
355       \let\hyxmp@crap@result=Y%
356       \expandafter\expandafter\expandafter\hyxmp@skiptorelax
357     \else
358       \expandafter\expandafter\expandafter\hyxmp@crap@test
359     \fi
360   \fi
361 }

```

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

```

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

```

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

```

\hyxmp@num 363 \def\hyxmp@crap@convert#1{%
\hyxmp@text 364   \ifx#1\relax
365   \else
366     \edef\hyxmp@num{\number'#1}%
367     \ifnum\hyxmp@num>"FFFFFF %
368       \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"1000000}\relax
369       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
370     \edef\hyxmp@num{\intcalcdMod{\hyxmp@num}{\number"1000000}}%
371   \else

```

```

372     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
373   \fi
374   \ifnum\hyxmp@num>"FFFF %
375     \lccode'\!=\intcalDiv{\hyxmp@num}{\number"10000}\relax
376     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
377     \edef\hyxmp@num{\intcalMod{\hyxmp@num}{\number"10000}}%
378   \else
379     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
380   \fi
381   \ifnum\hyxmp@num>"FF %
382     \lccode'\!=\intcalDiv{\hyxmp@num}{\number"100}\relax
383     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
384     \edef\hyxmp@num{\intcalMod{\hyxmp@num}{\number"100}}%
385   \else
386     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
387   \fi
388   \ifnum\hyxmp@num>0 %
389     \lccode'\!=\hyxmp@num\relax
390     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
391   \else
392     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
393   \fi
394   \expandafter\hyxmp@crap@convert
395 \fi
396 }

```

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

```

397 \begingroup
398   \catcode0=12 %
399   \gdef\hyxmp@zero{^^00}%
400 \endgroup

```

### 3.4 UUID generation

We use a linear congruential generator to produce pseudorandom version 4 UUIDs [8]. True, this method has its flaws but it’s simple to implement in  $\text{\TeX}$  and is good enough for producing the `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.

```

401 \def\hyxmp@modulo@a#1{%
402   \@tempcntb=\@tempcnta
403   \divide\@tempcntb by #1
404   \multiply\@tempcntb by #1
405   \advance\@tempcnta by -\@tempcntb
406 }

```

`\hyxmp@big@prime` Define a couple of large prime numbers that can still be stored in a  $\text{\TeX}$  counter.  
`\hyxmp@big@prime@ii` 407 \def\hyxmp@big@prime{536870923}



```

408 \def\hyxmp@big@prime@ii{536870027}

\hyxmp@seed@rng Seed hyperxmp's random-number generator from a given piece of text.
\hyxmp@one@token 409 \def\hyxmp@seed@rng#1{%
410   \@tempcnta=\hyxmp@big@prime
411   \futurelet\hyxmp@one@token\hyxmp@seed@rng@i#1\@empty
412 }

\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  $\text{\@tempcnta} \leftarrow 3 \cdot \text{\@tempcnta} + c \pmod{\text{\hyxmp@big@prime}}$ .
\next 413 \def\hyxmp@seed@rng@i{%
414   \ifx\hyxmp@one@token\@empty
415     \let\next=\relax
416   \else
417     \def\next##1{%
418       \multiply\@tempcnta by 3
419       \advance\@tempcnta by '##1
420       \hyxmp@modulo@a{\hyxmp@big@prime}%
421       \futurelet\hyxmp@one@token\hyxmp@seed@rng@i
422     }%
423   \fi
424   \next
425 }

\hyxmp@set@rand@num Advance \hyxmp@rand@num to the next pseudorandom number in the se-
\hyxmp@rand@num quence. 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  $\text{\@tempcnta}$ 
and  $\text{\@tempcntb}$  are overwritten in the process.
426 \def\hyxmp@set@rand@num{%
427   \@tempcnta=\hyxmp@rand@num
428   \multiply\@tempcnta by 3
429   \advance\@tempcnta by \hyxmp@big@prime@ii
430   \hyxmp@modulo@a{\hyxmp@big@prime}%
431   \xdef\hyxmp@rand@num{\the\@tempcnta}%
432 }

\hyxmp@append@hex Append a randomly selected hexadecimal digit to macro #1. Note that both
\@tempcnta and \@tempcntb are overwritten in the process.
433 \def\hyxmp@append@hex#1{%
434   \hyxmp@set@rand@num
435   \@tempcnta=\hyxmp@rand@num
436   \hyxmp@modulo@a{16}%
437   \ifnum\@tempcnta<10
438     \xdef#1{#1\the\@tempcnta}%
439   \else
440     \advance\@tempcnta by -10
441     \ifcase\@tempcnta

```

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

```

442     \xdef#1{#1a}%
443     \or\xdef#1{#1b}%
444     \or\xdef#1{#1c}%
445     \or\xdef#1{#1d}%
446     \or\xdef#1{#1e}%
447     \or\xdef#1{#1f}%
448   \fi
449 \fi
450 }

```

`\hyxmp@append@hex@iii` Invoke `\hyxmp@append@hex` three times.

```

451 \def\hyxmp@append@hex@iii#1{%
452   \hyxmp@append@hex#1%
453   \hyxmp@append@hex#1%
454   \hyxmp@append@hex#1%
455 }

```

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

```

456 \def\hyxmp@append@hex@iv#1{%
457   \hyxmp@append@hex@iii#1%
458   \hyxmp@append@hex#1%
459 }

```

`\hyxmp@create@uuid` As per the definition of a version 4 UUID [8], define macro `#1` as a UUID of the form “`uuid:xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx`” in which each “`x`” is a lowercase hexadecimal digit and “`y`” is one of “8”, “9”, “a”, or “b”. We assume that the random-number generator is already seeded. Note that `\hyxmp@create@uuid` overwrites both `\@tempcnta` and `\@tempcntb`.

```

460 \def\hyxmp@create@uuid#1{%
461   \def#1{uuid:}%
462   \hyxmp@append@hex@iv#1%
463   \hyxmp@append@hex@iv#1%
464   \g@addto@macro#1{-}%
465   \hyxmp@append@hex@iv#1%
466   \g@addto@macro#1{-4}%
467   \hyxmp@append@hex@iii#1%
468   \g@addto@macro#1{-}%

```

Randomly select one of “8”, “9”, “a”, or “b”.

```

469   \hyxmp@set@rand@num
470   \@tempcnta=\hyxmp@rand@num
471   \hyxmp@modulo@a{4}%
472   \ifcase\@tempcnta
473     \g@addto@macro#1{8}%
474     \or\g@addto@macro#1{9}%
475     \or\g@addto@macro#1{a}%
476     \or\g@addto@macro#1{b}%
477   \fi
478   \hyxmp@append@hex@iii#1%

```

```

479 \g@addto@macro#1{-}%
480 \hyxmp@append@hex@iv#1%
481 \hyxmp@append@hex@iv#1%
482 \hyxmp@append@hex@iv#1%
483 }

```

`\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.

```

484 \newcommand*{\hyxmp@def@DocumentID}{%
485 \edef\hyxmp@seed@string{\jobname:\@pdftitle:\@pdfauthor}%
486 \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
487 \edef\hyxmp@rand@num{\the\@tempcnta}%
488 \hyxmp@create@uuid\hyxmp@DocumentID
489 }

```

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

```

490 \newcommand*{\hyxmp@def@InstanceID}{%
491 \edef\hyxmp@seed@string{\jobname:\@pdftitle:\@pdfauthor:\hyxmp@today}%
492 \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
493 \edef\hyxmp@rand@num{\the\@tempcnta}%
494 \hyxmp@create@uuid\hyxmp@InstanceID
495 }

```

### 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), XMP Media Management (Section 3.5.5), XMP Basic (Section 3.5.6), Photoshop (Section 3.5.7), IPTC Photo Metadata (Section 3.5.8), and PDF/A Identification (Section 3.5.9). The `\hyxmp@construct@packet` macro (Section 3.5.10) 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.

```

496 \newcommand*{\hyxmp@add@to+xml}[1]{%
497 \bgroup
498 \@tempcnta=0

```

```

499     \loop
500     \lccode\@tempcnta=\@tempcnta
501     \advance\@tempcnta by 1
502     \ifnum\@tempcnta<256
503     \repeat
504     \lccode'\_='\ \relax
505     \lccode'\^^C='\ \relax
506     \lccode'\^^U='\_ \relax
507     \lowercase{\xdef\hyxmp@new@xml{#1}}%
508     \xdef\hyxmp@xml{\hyxmp@xml\hyxmp@new@xml}%
509 \egroup
510 }

```

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

```

511 \bgroup
512 \catcode'\#=11
513 \gdef\hyxmp@hash{#}
514 \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.

```

515 \bgroup
516 \xdef\hyxmp@xml{}%
517 \hyxmp@add@to@xml{%
518 -----^^J%
519 }
520 \xdef\hyxmp@padding{\hyxmp@xml}%
521 \egroup
522 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
523 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
524 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
525 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
526 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}

```

`\hyxmp@pdf@to@xmp@date` Convert a timestamp from PDF’s D:YYYYMMDDhhmmss-TT’tt’ format (e.g., D:20140622230121-06’00’) to XMP’s YYYY-MM-DDThh:mm:ss+TT:tt format (e.g., 2014-06-22T23:01:21-06:00) [4]. This macro is fully expandable.

```

527 \def\hyxmp@pdf@to@xmp@date#1:#2#3#4#5#6#7#8#9{%
528   #2#3#4#5-#6#7-#8#9%
529   \hyxmp@parse@time
530 }

```

`\hyxmp@parse@time` This is a helper function for `\hyxmp@pdf@to@xmp@date`. `\hyxmp@pdf@to@xmp@date` properly parses only the year, month, and day then calls `\hyxmp@parse@time`. `\hyxmp@parse@time` parses the hours, minutes, and seconds then calls `\hyxmp@parse@tz@char`.

```

531 \def\hyxmp@parse@time#1#2#3#4#5#6{%
532   T#1#2:#3#4:#5#6%
533   \hyxmp@parse@tz@char
534 }

```

`\hyxmp@parse@tz@char` This is another helper function for `\hyxmp@pdf@to@xmp@date`. So far, the date and time have been parsed. `\hyxmp@parse@tz@char` parses the first character of the timezone descriptor. This can be one of “+” for eastern timezones (UTC+ $x$ , including Asia, Oceania, and most of Europe), “-” for western timezones (UTC- $x$ , primarily the Americas), or “Z” for Zulu time (UTC+0). Timezones beginning with “+” or “-” are followed by an offset in hours and minutes (parsed by `\hyxmp@parse@tz`; timezones beginning with “Z” are not.

```

535 \def\hyxmp@parse@tz@char#1{%
536   #1%
537   \ifx#1-%
538     \expandafter\hyxmp@parse@tz
539   \else
540     \ifx#1+%
541       \expandafter\hyxmp@parse@tz
542     \fi
543   \fi
544 }

```

`\hyxmp@parse@tz` This is the final helper function for `\hyxmp@pdf@to@xmp@date`. It parses the piece of the timezone comprising the offset from Coordinated Universal Time, measured in hours and minutes.

```

545 \def\hyxmp@parse@tz#1'#2'{%
546   #1:#2%
547 }

```

`\hyxmp@today@define` Use TeX’s `\year`, `\month`, and `\day` primitives to define `\hyxmp@today` as today’s date in YYYY-MM-DD format.

```

548 \def\hyxmp@today@define{%
549   \xdef\hyxmp@today{\the\year}%
550   \ifnum\month<10
551     \xdef\hyxmp@today{\hyxmp@today-0\the\month}%
552   \else
553     \xdef\hyxmp@today{\hyxmp@today-\the\month}%
554   \fi
555   \ifnum\day<10
556     \xdef\hyxmp@today{\hyxmp@today-0\the\day}%
557   \else
558     \xdef\hyxmp@today{\hyxmp@today-\the\day}%
559   \fi
560 }

```

`\hyxmp@today` Define `\hyxmp@today` as the current date and (if available) time and timezone in XMP Date format [4].

```

561 \expandafter\ifx\csname pdfcreationdate\endcsname\relax
562   \hyxmp@today@define
563 \else
564   \edef\hyxmp@today{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
565 \fi

```

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

```

566 \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.

```

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

```

Add a block of XML to `\hyxmp@xml` that lists the document’s keywords (the `pdf:Keywords` property), the tools used to produce the PDF file (the `pdf:Producer` property), and the version of the PDF standard adhered to (the `pdf:PDFVersion` property). Unlike most of the other schemata that `hyperxmp` supports, the Adobe PDF schema is *always* included in the document, even if all of its keys are empty. This is because PDF/A-1b requires the keywords and producer to be the same in the XMP metadata and the PDF metadata. Because `hyperref` always specifies the `Keywords` and `Producer` fields, even when they’re empty, `hyperxmp` has to follow suit and define `pdf:Keywords` and `pdf:Producer` in the XMP packet.

```

568   \hyxmp@add@to@xml{%
569   -----<rdf:Description rdf:about=""^^J%
570   -----xmlns:pdf="http://ns.adobe.com/pdf/1.3/">^^J%
571   }%
572   \hyxmp@add@simple@var{pdf:Keywords}{@pdfkeywords}%
573   \hyxmp@add@simple@var{pdf:Producer}{@pdfproducer}%
574   \ifundefined{pdfminorversion}{-}%
575   \hyxmp@add@simple{pdf:PDFVersion}{1.\the\pdfminorversion}%
576   }%
577   \hyxmp@add@to@xml{%
578   -----</rdf:Description>^^J%
579   }%
580 }

```

`\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`

```

581 \newcommand*{\hyxmp@add@simple}[2]{%
582   \edef\hyxmp@string{#2}%
583   \ifx\hyxmp@string\empty
584   \else
585     \hyxmp@xmlify{\hyxmp@string}%
586     \hyxmp@add@to@xml{%
587     -----<#1>\hyxmp@xmlified</#1>^^J%
588     }%
589   \fi

```

590 }

`\hyxmp@add@simple@var` Given an XMP tag (#1) and a variable name (#2), if the string is defined, 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@add@simple@var` differs from `\hyxmp@add@simple` in that the former includes defined but empty values in the XMP packet while the latter excludes both undefined and defined but empty values.

```
591 \newcommand*{\hyxmp@add@simple@var}[2]{%
592   \expandafter\ifx\csname#2\endcsname\relax
593   \else
594     \hyxmp@xmllify{\csname#2\endcsname}%
595     \hyxmp@add@to@xml{%
596     -----<#1>\hyxmp@xmllified</#1>^^J%
597     }%
598   \fi
599 }
```

### 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.

```
600 \newcommand*{\hyxmp@rdf@dc}[2]{%
601   \ifx#2@empty
602   \else
603     \hyxmp@xmllify{#2}%
604     \hyxmp@add@to@xml{%
605     -----<dc:#1>^^J%
606     -----<rdf:Alt>^^J%
607     }%
608     \ifx\@pdfmetalang\hyxmp@x@default
609     \else
610       \hyxmp@add@to@xml{%
611       -----<rdf:li xml:lang="\@pdfmetalang">\hyxmp@xmllified</rdf:li>^^J%
612       }%
613       \fi
614       \hyxmp@add@to@xml{%
615       -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@xmllified</rdf:li>^^J%
616       -----</rdf:Alt>^^J%
617       -----</dc:#1>^^J%
618       }%
619     \fi%
620 }
```

`\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.

```

621 \newcommand*{\hyxmp@list@to@xml}[3]{%
622   \ifx#3\@empty
623   \else
624     \hyxmp@add@to@xml{%
625       -----<dc:#1>^^J%
626       -----<rdf:#2>^^J%
627     }%
628   \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`.

```

629     \hyxmp@xmlify{#3}%
630     \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
631     \def\@elt##1{%
632       \hyxmp@add@to@xml{%
633         -----<rdf:li>##1</rdf:li>^^J%
634       }%
635     }%
636     \hyxmp@list
637   \egroup
638   \hyxmp@add@to@xml{%
639     -----</rdf:#2>^^J%
640     -----</dc:#1>^^J%
641   }%
642 \fi
643 }

```

`\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.

```

644 \newcommand*{\hyxmp@dc@schema}{%
645   \hyxmp@add@to@xml{%
646     -----<rdf:Description rdf:about=""^^J%
647     -----xmlns:dc="http://purl.org/dc/elements/1.1/"^^J%
648     -----<dc:format>application/pdf</dc:format>^^J%
649   }%
650   \hyxmp@rdf@dc{title}{\@pdftitle}%
651   \hyxmp@rdf@dc{description}{\@pdfsubject}%
652   \hyxmp@rdf@dc{rights}{\@pdfcopyright}%
653   \hyxmp@list@to@xml{creator}{Seq}{\hyxmp@pdfauthor}%
654   \hyxmp@list@to@xml{subject}{Bag}{\hyxmp@pdfkeywords}%
655   \hyxmp@list@to@xml{date}{Seq}{\hyxmp@today}%
656   \hyxmp@list@to@xml{language}{Bag}{\@pdflang}%

```



```

657 \hyxmp@add@simple{dc:source}{\jobname.tex}%
658 \hyxmp@add@to@xml{%
659 -----</rdf:Description>^^J%
660 }%
661 }

```

### 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.

```

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

```

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

```

663 \let\hyxmp@rights=\@empty
664 \ifx\@pdflicenseurl\@empty
665 \else
666 \def\hyxmp@rights{YES}%
667 \fi
668 \ifx\@pdfcopyright\@empty
669 \else
670 \def\hyxmp@rights{YES}%
671 \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.

```

672 \ifx\hyxmp@rights\@empty
673 \else

```

Header

```

674 \hyxmp@add@to@xml{%
675 -----<rdf:Description rdf:about=""^^J%
676 -----xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/">^^J%
677 }%

```

Copyright indication

```

678 \ifx\@pdfcopyright\@empty
679 \else
680 \hyxmp@add@to@xml{%
681 -----<xmpRights:Marked>True</xmpRights:Marked>^^J%
682 }%
683 \fi

```

License URL

```

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

```

Trailer

```

685 \hyxmp@add@to@xml{%
686 -----</rdf:Description>^^J%

```

```

687     }%
688     \fi
689 }

```

### 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.

```

690 \gdef\hyxmp@mm@schema{%
691   \hyxmp@def@DocumentID
692   \hyxmp@def@InstanceID
693   \hyxmp@add@to+xml{%
694     <rdf:Description rdf:about=""^^J%
695     _____xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/">^^J%
696     <xmpMM:DocumentID>\hyxmp@DocumentID</xmpMM:DocumentID>^^J%
697     <xmpMM:InstanceID>\hyxmp@InstanceID</xmpMM:InstanceID>^^J%
698     </rdf:Description>^^J%
699   }%
700 }

```

### 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`.

```

701 \newcommand*{\hyxmp@xmp@basic@schema}{%
702   \hyxmp@add@to+xml{%
703     <rdf:Description rdf:about=""^^J%
704     _____xmlns:xmp="http://ns.adobe.com/xap/1.0/">^^J%
705   }%
706   \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@today}%
707   \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@today}%
708   \hyxmp@add@simple{xmp:MetadataDate}{\hyxmp@today}%
709   \hyxmp@add@simple{xmp:CreatorTool}{\@pdfcreator}%
710   \hyxmp@add@simple{xmp:BaseURL}{\@baseurl}%
711   \hyxmp@add@to+xml{%
712     </rdf:Description>^^J%
713   }%
714 }

```

### 3.5.7 The Photoshop schema

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

```

715 \gdef\hyxmp@photoshop@schema{%
716   \edef\hyxmp@photoshop@data{\@pdfauthor\title\@pdfcaptionwriter}%
717   \ifx\hyxmp@photoshop@data\@empty
718     \else
719       \hyxmp@add@to@xml{%
720         <rdf:Description rdf:about="^^J%
721         -----xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/">^^J%
722       }%
723     \fi
724     \hyxmp@add@simple{photoshop:AuthorsPosition}{\@pdfauthor\title}%
725     \hyxmp@add@simple{photoshop:CaptionWriter}{\@pdfcaptionwriter}%
726     \ifx\hyxmp@photoshop@data\@empty
727       \else
728         \hyxmp@add@to@xml{%
729           </rdf:Description>^^J%
730         }%
731       \fi
732 }

```

### 3.5.8 The IPTC Photo Metadata schema

`\xmplinesep` Lines in multiline fields are separated by `\xmplinesep` in the generated XML. This defaults to an LF (`^^J`) character but written as an XML character entity for consistency across operating systems.

```

733 \begingroup
734   \catcode'\&=12
735   \catcode'\#=12
736   \gdef\xmplinesep{&#xA;}
737 \endgroup

```

`\hyxmp@list@to@lines` Given a property (#1) and a macro containing a comma-separated list (#2), replace commas with `\xmplinesep`. Do nothing if the list is empty.

```

738 \newcommand*{\hyxmp@list@to@lines}[2]{%
739   \ifx#2\@empty
740     \else
741       \bgroup
742         \hyxmp@add@to@xml{%
743           -----<#1>%
744         }%

```

`\@elt@first` The first element of the list is output as is.

```

745   \def\@elt@first##1{%
746     \hyxmp@add@to@xml{##1}%
747     \let\@elt=\@elt@rest
748   }%

```

`\@elt@rest` The remaining elements of the list are output with a preceding line separator (`\xmplinesep`).

```

749   \def\@elt@rest##1{%

```

```

750      \hyxmp@add@to@xml{\xmplinesep##1}%
751      }%

```

\@elt Re-encode the text from Unicode if necessary. Then redefine \@elt to insert a line separator between terms.

```

752      \let\@elt=\@elt@first
753      \hyxmp@xmlify{#2}%
754      \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
755      \hyxmp@list
756      \hyxmp@add@to@xml{</#1>^^J}%
757      \egroup
758      \fi
759  }

```

\hyxmp@photometa@schema Add properties defined by the IPTC Photo Metadata schema [6] to the \hyxmp@xml macro. We currently support only the contact-information details structure, viz. the Iptc4xmpCore:CreatorContactInfo/CiAdrExtadr, Iptc4xmpCore:CreatorContactInfo/CiAdrCity, Iptc4xmpCore:CreatorContactInfo/CiAdrRegion, Iptc4xmpCore:CreatorContactInfo/CiAdrPcode, Iptc4xmpCore:CreatorContactInfo/CiAdrCtry, Iptc4xmpCore:CreatorContactInfo/CiTelWork, Iptc4xmpCore:CreatorContactInfo/CiEmailWork, and Iptc4xmpCore:CreatorContactInfo/CiUrlWork properties.

```

760 \gdef\hyxmp@photometa@schema{%
761   \edef\hyxmp@photometa@data{%
762     \@pdfcontactaddress
763     \@pdfcontactcity
764     \@pdfcontactregion
765     \@pdfcontactpostcode
766     \@pdfcontactcountry
767     \@pdfcontactphone
768     \@pdfcontactemail
769     \@pdfcontacturl
770   }%
771   \ifx\hyxmp@photometa@data\@empty
772   \else
773     \hyxmp@iptc@extensions
774     \hyxmp@add@to@xml{%
775       <rdf:Description rdf:about=""^^J%
776       <-----xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"^^J%
777       <-----xmlns:IptcContInfo="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/contactinfo/">^^J%
778       <-----<Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">^^J%
779       }%
780     \fi
781     \hyxmp@list@to@lines{IptcContInfo:CiAdrExtadr}{\@pdfcontactaddress}%
782     \hyxmp@add@simple{IptcContInfo:CiAdrCity}{\@pdfcontactcity}%
783     \hyxmp@add@simple{IptcContInfo:CiAdrRegion}{\@pdfcontactregion}%
784     \hyxmp@add@simple{IptcContInfo:CiAdrPcode}{\@pdfcontactpostcode}%
785     \hyxmp@add@simple{IptcContInfo:CiAdrCtry}{\@pdfcontactcountry}%

```

`\xmplinesep` The IPTC standard states that sets of telephone numbers, email addresses, and URLs for the contact person or institution, “[m]ay have to be separated by a comma in the user interface” [6]. This is rather ambiguous: Does the comma appear *only* in the user interface or also in the generated XML? Here we assume the latter interpretation and temporarily redefine `\xmplinesep` as a comma and use `\hyxmp@list@to@lines` to insert the data. Unlike `\hyxmp@add@simple`, this approach trims all spaces surrounding commas.

```

786 \bgroup
787 \def\xmplinesep{,}%
788 \hyxmp@list@to@lines{IptcContInfo: CiTelWork}{\@pdfcontactphone}%
789 \hyxmp@list@to@lines{IptcContInfo: CiEmailWork}{\@pdfcontactemail}%
790 \hyxmp@list@to@lines{IptcContInfo: CiUrlWork}{\@pdfcontacturl}%
791 \egroup
792 \ifx\hyxmp@photometa@data\empty
793 \else
794 \hyxmp@add@to@xml{%
795 -----</Iptc4xmpCore:CreatorContactInfo>^^J%
796 -----</rdf:Description>^^J%
797 }%
798 \fi
799 }

```

`\hyxmp@iptc@extensions` Because IPTC metadata are not recognized by the PDF/A standard, PDF/A conversion would normally fail for documents that utilize `\pdfcontactaddress`, `\pdfcontactcity`, etc. However, there exists a technique, described in a PDF Association technical note [10], for describing nonstandard XMP metadata within the XMP packet itself. We use that technique here to describe all of the metadata that `\hyxmp@photometa@schema` can produce. Doing so enables the document to be converted to PDF/A format.

```

800 \newcommand*{\hyxmp@iptc@extensions}{%
801 \hyxmp@add@to@xml{%
802 -----<rdf:Description rdf:about=""^^J%
803 -----xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"^^J%
804 -----xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema\hyxmp@hash"^^J%
805 -----xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property\hyxmp@hash"^^J%
806 -----xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type\hyxmp@hash"^^J%
807 -----xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field\hyxmp@hash">^^J%
808 -----<pdfaExtension:schemas>^^J%
809 -----<rdf:Bag>^^J%
810 -----<rdf:li rdf:parseType="Resource">^^J%
811 -----<pdfaSchema:schema>IPTC Core Schema</pdfaSchema:schema>^^J%
812 -----<pdfaSchema:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaSchema:
813 -----<pdfaSchema:prefix>Iptc4xmpCore</pdfaSchema:prefix>^^J%
814 -----<pdfaSchema:property>^^J%
815 -----<rdf:Seq>^^J%
816 -----<rdf:li rdf:parseType="Resource">^^J%
817 -----<pdfaProperty:name>CreatorContactInfo</pdfaProperty:name>^^J%
818 -----<pdfaProperty:valueType>contactinfo</pdfaProperty:valueType>^^J%

```

```

819 -----<pdfaProperty:category>external</pdfaProperty:category>^^J%
820 -----<pdfaProperty:description>contact information for the document's creator</p
821 -----</rdf:li>^^J%
822 -----</rdf:Seq>^^J%
823 -----</pdfaSchema:property>^^J%
824 -----<pdfaSchema:valueType>^^J%
825 -----<rdf:Seq>^^J%
826 -----<rdf:li rdf:parseType="Resource">^^J%
827 -----<pdfaType:type>contactinfo</pdfaType:type>^^J%
828 -----<pdfaType:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/contactin
829 -----<pdfaType:prefix>IptcContInfo</pdfaType:prefix>^^J%
830 -----<pdfaType:description>contact information</pdfaType:description>^^J%
831 -----<pdfaType:field>^^J%
832 -----<rdf:Seq>^^J%
833 }%

834 \hyxmp@text@resource{CiAdrExtadr}{contact address}%
835 \hyxmp@text@resource{CiAdrCity}{contact city}%
836 \hyxmp@text@resource{CiAdrRegion}{contact region}%
837 \hyxmp@text@resource{CiAdrPcode}{contact postal code}%
838 \hyxmp@text@resource{CiAdrCtry}{contact country}%
839 \hyxmp@text@resource{CiTelWork}{contact telephone number}%
840 \hyxmp@text@resource{CiEmailWork}{contact email address}%
841 \hyxmp@text@resource{CiUrlWork}{contact url}%

842 \hyxmp@add@to+xml{%
843 -----</rdf:Seq>^^J%
844 -----</pdfaType:field>^^J%
845 -----</rdf:li>^^J%
846 -----</rdf:Seq>^^J%
847 -----</pdfaSchema:valueType>^^J%
848 -----</rdf:li>^^J%
849 -----</rdf:Bag>^^J%
850 -----</pdfaExtension:schemas>^^J%
851 -----</rdf:Description>^^J%
852 }%
853 }

```

\hyxmp@text@resource    Output a single Text resource given its name and description.

```

854 \newcommand*{\hyxmp@text@resource}[2]{%
855   \hyxmp@add@to+xml{%
856 -----<rdf:li rdf:parseType="Resource">^^J%
857 -----<pdfaField:name>#1</pdfaField:name>^^J%
858 -----<pdfaField:valueType>Text</pdfaField:valueType>^^J%
859 -----<pdfaField:description>#2</pdfaField:description>^^J%
860 -----</rdf:li>^^J%
861   }
862 }

```

### 3.5.9 The PDF/A Identification schema

`\hyxmp@pdfa@id@schema` Add properties defined by the PDF/A Identification schema [9] to the `\hyxmp+xml` macro. These properties identify a document as conforming to a particular PDF/A standard. Currently, we assume PDF/A-1b if any PDF/A compliance is detected.

```

863 \newcommand*{\hyxmp@pdfa@id@schema}{%
864   \ifHy@pdfa
865     \hyxmp@add@to+xml{%
866       <rdf:Description rdf:about=""^^J%
867       -----xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/">^^J%
868     }%
869     \hyxmp@add@simple{pdfaid:part}{1}%
870     \hyxmp@add@simple{pdfaid:conformance}{B}%
871     \hyxmp@add@to+xml{%
872       </rdf:Description>^^J%
873     }%
874   \fi
875 }
```

#### 3.5.10 Combining schemata into an XMP packet

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

```

876 \begingroup
877   \ifhyxmp@unicodetex
878     \lccode'\!="FEFF %
879     \lowercase{%
880       \gdef\hyxmp@bom{!}
881     }%
882   \else
883     \catcode'\^^ef=12
884     \catcode'\^^bb=12
885     \catcode'\^^bf=12
886     \gdef\hyxmp@bom{^^ef^^bb^^bf}%
887   \fi
888 \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.

```

889 \def\hyxmp@construct@packet{%
890   \gdef\hyxmp+xml{%
891     \hyxmp@add@to+xml{<?xpacket begin="\hyxmp@bom" %
892 id="W5M0MpCehiHzreSzNTczkc9d"?>^^J%
893 <x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="3.1-702">^^J%
894 ___<rdf:RDF
895   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\hyxmp@hash">^^J%
896   }%
897   \hyxmp@pdf@schema
898   \hyxmp@xmpRights@schema
899   \hyxmp@dc@schema
```

```

900 \hyxmp@photoshop@schema
901 \hyxmp@photometa@schema
902 \hyxmp@xmp@basic@schema
903 \hyxmp@pdfa@id@schema
904 \hyxmp@mm@schema
905 \hyxmp@add@to+xml{%
906 ___</rdf:RDF>^^J%
907 </x:xmpmeta>^^J%
908 \hyxmp@padding
909 <?xpacket end="w"?>^^J%
910 }%
911 }

```

### 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 function.  
`\hyxmp@driver`

```

912 \newcommand*{\hyxmp@embed@packet}{%
913   \hyxmp@construct@packet
914   \def\hyxmp@driver{hpdftex}%
915   \ifx\hyxmp@driver\Hy@driver
916     \hyxmp@embed@packet@pdftex
917   \else
918     \def\hyxmp@driver{hdvipdfm}%
919     \ifx\hyxmp@driver\Hy@driver
920       \hyxmp@embed@packet@dvipdfm
921     \else
922       \def\hyxmp@driver{hxdetex}%
923       \ifx\hyxmp@driver\Hy@driver
924         \hyxmp@embed@packet@xdetex
925       \else
926         \@ifundefined{pdfmark}{%
927           \PackageWarningNoLine{hyperxmp}{%
928             Unrecognized hyperref driver ‘\Hy@driver’.\MessageBreak
929             \jobname.tex’s XMP metadata will *not* be\MessageBreak
930             embedded in the resulting file}%
931         }{%
932           \hyxmp@embed@packet@pdfmark
933         }%
934       \fi
935     \fi
936   \fi
937 }

```



### 3.6.1 Embedding using pdfTeX

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

```
938 \newcommand*{\hyxmp@embed@packet@pdftex}{%
939   \bgroup
940     \pdfcompresslevel=0
941     \immediate\pdfobj stream attr {%
942       /Type /Metadata
943       /Subtype /XML
944     }\hyxmp@xml}%
945   \pdfcatalog {/Metadata \the\pdflastobj\space 0 R}%
946   \egroup
947 }
```

### 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.

```
948 \newcommand*{\hyxmp@embed@packet@pdfmark}{%
949   \pdfmark{%
950     pdfmark=/NamespacePush
951   }%
952   \pdfmark{%
953     pdfmark=/OBJ,
954     Raw={/_objdef \string{hyxmp@Metadata\string} /type /stream}%
955   }%
956   \pdfmark{%
957     pdfmark=/PUT,
958     Raw={\string{hyxmp@Metadata\string}
959       2 dict begin
960         /Type /Metadata def
961         /Subtype /XML def
962         currentdict
963       end
964     }%
965   }%
966   \pdfmark{%
967     pdfmark=/PUT,
968     Raw={\string{hyxmp@Metadata\string} (\hyxmp@xml)}%
969   }%
970   \pdfmark{%
971     pdfmark=/Metadata,
972     Raw={\string{Catalog\string} \string{hyxmp@Metadata\string}}%
973   }%
974   \pdfmark{%
975     pdfmark=/NamespacePop
976   }%
977 }
```

### 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.

```

978 \newcommand*{\hyxmp@embed@packet@dvipdfm}{%
979   \hyxmp@string@len{\hyxmp@xml}%
980   \special{pdf: object @hyxmp@Metadata
981     <<
982       /Type /Metadata
983       /Subtype /XML
984       /Length \the\@tempcnta
985     >>
986     stream^^J\hyxmp@xml endstream%
987   }%
988   \special{pdf: docview
989     <<
990       /Metadata @hyxmp@Metadata
991     >>
992   }%
993 }
```

`\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.

```

994 \newcommand*{\hyxmp@string@len}[1]{%
995   \@tempcnta=0
996   \expandafter\hyxmp@count@spaces#1 {} %
997   \expandafter\hyxmp@count@non@spaces#1{}%
998 }
```

`\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.

```

999 \def\hyxmp@count@spaces#1 {%
1000   \def\hyxmp@one@token{#1}%
1001   \ifx\hyxmp@one@token\@empty
1002     \advance\@tempcnta by -1
1003   \else
1004     \advance\@tempcnta by 1
1005     \expandafter\hyxmp@count@spaces
1006   \fi
1007 }
```

`\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<sub>E</sub>X 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.

```

1008 \newcommand*{\hyxmp@count@non@spaces}[1]{%
1009   \def\hyxmp@one@token{#1}%
1010   \ifx\hyxmp@one@token\@empty
1011     \else
1012       \advance\@tempcnta by 1
1013       \expandafter\hyxmp@count@non@spaces
1014     \fi
1015 }

```

### 3.6.4 Embedding using X<sub>Y</sub>TeX

`\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.

```

1016 \newcommand*{\hyxmp@embed@packet@xetex}{%
1017   \special{pdf:stream @hyxmp@Metadata (\hyxmp+xml)
1018     <<
1019       /Type /Metadata
1020       /Subtype /XML
1021     >>
1022   }%
1023   \special{pdf:put @catalog
1024     <<
1025       /Metadata @hyxmp@Metadata
1026     >>
1027   }%
1028 }

```

## 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.

```

1029 \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, X<sub>Y</sub>TeX, etc.), please send me a code patch.

## A Sample XMP packet

The following is an example of a complete XMP packet as may be produced by hyperxmp. This packet corresponds to the metadata included in the sample L<sup>A</sup>TeX

document presented on page 5. For clarity, metadata values, either specified explicitly by the document or introduced automatically by hyperxmp, are colored blue.

```
<?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>
      <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>2014-06-22T23:01:21-06:00</rdf:li>
    </rdf:Seq>
</dc:date>
<dc:language>
    <rdf:Bag>
        <rdf:li>en</rdf:li>
    </rdf:Bag>
</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:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"
    xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema#"
    xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property#"
    xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type#"
    xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field#">
    <pdfaExtension:schemas>
        <rdf:Bag>
            <rdf:li rdf:parseType="Resource">

```

```

<pdfaSchema:schema>IPTC Core Schema</pdfaSchema:schema>
<pdfaSchema:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaSchema:namespaceURI>
<pdfaSchema:prefix>Iptc4xmpCore</pdfaSchema:prefix>
<pdfaSchema:property>
  <rdf:Seq>
    <rdf:li rdf:parseType="Resource">
      <pdfaProperty:name>CreatorContactInfo</pdfaProperty:name>
      <pdfaProperty:valueType>contactinfo</pdfaProperty:valueType>
      <pdfaProperty:category>external</pdfaProperty:category>
      <pdfaProperty:description>contact information for the document's creator</pdfaProperty:description>
    </rdf:li>
  </rdf:Seq>
</pdfaSchema:property>
<pdfaSchema:valueType>
  <rdf:Seq>
    <rdf:li rdf:parseType="Resource">
      <pdfaType:type>contactinfo</pdfaType:type>
      <pdfaType:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaType:namespaceURI>
      <pdfaType:prefix>IptcContInfo</pdfaType:prefix>
      <pdfaType:description>contact information</pdfaType:description>
      <pdfaType:field>
        <rdf:Seq>
          <rdf:li rdf:parseType="Resource">
            <pdfaField:name>CiAdrExtadr</pdfaField:name>
            <pdfaField:valueType>Text</pdfaField:valueType>
            <pdfaField:description>contact address</pdfaField:description>
          </rdf:li>
          <rdf:li rdf:parseType="Resource">
            <pdfaField:name>CiAdrCity</pdfaField:name>
            <pdfaField:valueType>Text</pdfaField:valueType>
            <pdfaField:description>contact city</pdfaField:description>
          </rdf:li>
          <rdf:li rdf:parseType="Resource">
            <pdfaField:name>CiAdrRegion</pdfaField:name>
            <pdfaField:valueType>Text</pdfaField:valueType>
            <pdfaField:description>contact region</pdfaField:description>
          </rdf:li>
          <rdf:li rdf:parseType="Resource">
            <pdfaField:name>CiAdrPcode</pdfaField:name>
            <pdfaField:valueType>Text</pdfaField:valueType>
            <pdfaField:description>contact postal code</pdfaField:description>
          </rdf:li>
          <rdf:li rdf:parseType="Resource">
            <pdfaField:name>CiAdrCtry</pdfaField:name>
            <pdfaField:valueType>Text</pdfaField:valueType>
            <pdfaField:description>contact country</pdfaField:description>
          </rdf:li>
        </rdf:Seq>
      </pdfaType:field>
    </rdf:li>
  </rdf:Seq>
</pdfaSchema:valueType>

```

```

        </rdf:li>
        <rdf:li rdf:parseType="Resource">
            <pdfaField:name>CiTelWork</pdfaField:name>
            <pdfaField:valueType>Text</pdfaField:valueType>
            <pdfaField:description>contact telephone number</pdfaField:de
        </rdf:li>
        <rdf:li rdf:parseType="Resource">
            <pdfaField:name>CiEmailWork</pdfaField:name>
            <pdfaField:valueType>Text</pdfaField:valueType>
            <pdfaField:description>contact email address</pdfaField:de
        </rdf:li>
        <rdf:li rdf:parseType="Resource">
            <pdfaField:name>CiUrlWork</pdfaField:name>
            <pdfaField:valueType>Text</pdfaField:valueType>
            <pdfaField:description>contact url</pdfaField:description>
        </rdf:li>
    </rdf:Seq>
</pdfaType:field>
</rdf:li>
</rdf:Seq>
</pdfaSchema:valueType>
</rdf:li>
</rdf:Bag>
</pdfaExtension:schemas>
</rdf:Description>
<rdf:Description rdf:about=""
    xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"
    xmlns:IptcContInfo="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/contactinf
<Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">
    <IptcContInfo:CiAdrExtadr>Kramgasse 49</IptcContInfo:CiAdrExtadr>
    <IptcContInfo:CiAdrCity>Bern</IptcContInfo:CiAdrCity>
    <IptcContInfo:CiAdrPcode>3011</IptcContInfo:CiAdrPcode>
    <IptcContInfo:CiAdrCtry>Switzerland</IptcContInfo:CiAdrCtry>
    <IptcContInfo:CiTelWork>031 312 00 91</IptcContInfo:CiTelWork>
    <IptcContInfo:CiEmailWork>aeinstein@ipi.ch</IptcContInfo:CiEmailWork>
    <IptcContInfo:CiUrlWork>
        http://einstein.biz/,
        https://www.facebook.com/AlbertEinstein
    </IptcContInfo:CiUrlWork>
</Iptc4xmpCore:CreatorContactInfo>
</rdf:Description>
<rdf:Description rdf:about=""
    xmlns:xmp="http://ns.adobe.com/xap/1.0/">
    <xmp:CreateDate>2014-06-22T23:01:21-06:00</xmp:CreateDate>
    <xmp:ModifyDate>2014-06-22T23:01:21-06:00</xmp:ModifyDate>
    <xmp:MetadataDate>2014-06-22T23:01:21-06:00</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"?>

```

## 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*, April 2012. Available from <http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/xmp/pdfs/cc-201306/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] International Press Telecommunications Council. *IPTC Photo Metadata: Core 1.1/Extension 1.1*, July 2010. Revision 1. Available from [http://www.iptc.org/std/photometadata/specification/IPTC-PhotoMetadata-201007\\_1.pdf](http://www.iptc.org/std/photometadata/specification/IPTC-PhotoMetadata-201007_1.pdf).



- [7] Internet Assigned Numbers Authority. Language subtag registry, January 11, 2011. Available from <http://www.iana.org/assignments/language-subtag-registry>.
- [8] Paul J. Leach, Michael Mealling, and Rich Salz. A Universally Unique Identifier (UUID) URN namespace. Request for Comments 4122, Internet Engineering Task Force, Network Working Group, July 2005. Category: Standards Track. Available from <http://www.ietf.org/rfc/rfc4122.txt>.
- [9] PDF/A Competence Center, Berlin, Germany. *TechNote 0008: Pre-defined XMP Properties in PDF/A-1*, March 20, 2008. Available from [http://www.pdfa.org/wp-content/uploads/2011/08/tn0008\\_predefined\\_xmp\\_properties\\_in\\_pdfa-1\\_2008-03-20.pdf](http://www.pdfa.org/wp-content/uploads/2011/08/tn0008_predefined_xmp_properties_in_pdfa-1_2008-03-20.pdf).
- [10] PDF/A Competence Center, Berlin, Germany. *TechNote 0009: XMP Extension Schemas in PDF/A-1*, March 20, 2008. Available from [http://www.pdfa.org/wp-content/uploads/2011/08/tn0009\\_xmp\\_extension\\_schemas\\_in\\_pdfa-1\\_2008-03-20.pdf](http://www.pdfa.org/wp-content/uploads/2011/08/tn0009_xmp_extension_schemas_in_pdfa-1_2008-03-20.pdf).

## Change History

v1.0			strings as 8-bit strings before manipulating them into XMP schema. This change addresses a bug reported by Martin Münch	19
General: Initial version	.....	1		
v1.1				
\hyxmp@construct@packet:	Explicitly set the category codes of characters $\langle EF \rangle$ , $\langle BB \rangle$ , and $\langle BF \rangle$ to “letter”. Thanks to Daniel Schömer for the bug report	.....	39	
v1.2				
General:	Added support for the X <sub>Y</sub> TEX backend (x <sub>d</sub> v <sub>i</sub> p <sub>d</sub> f <sub>m</sub> x)	..	1	
	Added support for the Photoshop schema	.....	1	
	Made the package compatible with ngerman. Thanks to Tobias Mueller for the bug report.	...	9	
v1.3				
General:	Introduced the pdfmetalang package option, which enables an author to specify the language in which he wrote the document’s metadata	.....	15	
\hyxmp@reencode:	Introduced this macro to re-encode Unicode			
v1.4				
\hyxmp@mm@schema:	Renamed the xapMM namespace prefix to xmpMM	.....	34	
\hyxmp@rdf@dc:	Included metadata in the x-default language regardless of the specified metadata language	.....	31	
\hyxmp@xmpRights@schema:	Renamed the xapRights namespace prefix to xmpRights	....	33	
v1.5				
General:	Made the XMP inclusion more robust. Thanks to Heiko Oberdiek for the bug report and suggested modifications.	....	10	
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 . . . . .	15	\hyxmp@xmlify: Completely rewritten by Heiko Oberdiek to better support Unicode-enabled T <sub>E</sub> X programs . . . . .	19
\hyxmp@add@to@xml: Updated also to replace commas . . . . .	27	\hyxmp@xmp@basic@schema: Added this macro . . . . .	34
\hyxmp@bom: Added by Heiko Oberdiek . . . . .	39	\hyxmp@xmpRights@schema: Modified to include xmpRights:Marked only when pdfcopyright is specified and xmpRights:WebStatement only when pdflicenseurl is specified . . . . .	33
\hyxmp@comma: Added this macro	17	\hyxmp@zero: Added by Heiko Oberdiek . . . . .	24
\hyxmp@construct@packet: Modified by Heiko Oberdiek to use an appropriate BOM representation via \hyxmp@bom . . . . .	39	\ifhyxmp@unicodetex: Added by Heiko Oberdiek . . . . .	19
\hyxmp@crap@convert: Added by Heiko Oberdiek . . . . .	23	\ProcessKeyvalOptions: Added this macro . . . . .	14
\hyxmp@crap@test: Added by Heiko Oberdiek . . . . .	23	\xmpcomma: Added this macro . . .	17
\hyxmp@dc@schema: Added support for dc:language and dc:source .	32	\xmpquote: Added this macro . . .	17
\hyxmp@is@unicode: Added by Heiko Oberdiek . . . . .	20	\XMPTruncateList: Added this macro . . . . .	17
\hyxmp@list@to@xml: Modified by Heiko Oberdiek to use the new Unicode-processing macros . .	31		
\hyxmp@photoshop@schema: Simplified using \hyxmp@add@simple	34	v2.1	
\hyxmp@ProcessKeyvalOptions: Added this macro . . . . .	14	General: Enabled hyperxmp and hyperref to be loaded in either order. This addresses a bug report by Yury Donskoy . . . . .	12
\hyxmp@reencode: Replaced with an empty macro by Heiko Oberdiek . . . . .	19	\hypersetup: Added this macro .	14
\hyxmp@skiptorelax: Added by Heiko Oberdiek . . . . .	23	\hyxmp@hypersetup: Added this macro . . . . .	14
\hyxmp@skipzeros: Added by Heiko Oberdiek . . . . .	22	\hyxmp@redefine@Hyp: Added this macro . . . . .	13
\hyxmp@SpaceOther: Added by Heiko Oberdiek . . . . .	23		
\hyxmp@string: Added this macro	30	v2.2	
\hyxmp@toxml: Added by Heiko Oberdiek . . . . .	21	General: Added support for the IPTC Photo Metadata schema . . . . .	1
Escaped parentheses written with pdfmarks to prevent dvips from line-wrapping the XMP packet	21	\hyxmp@iptc@extensions: Added this macro to support PDF/A generation . . . . .	37
\hyxmp@toxml@unicodetex: Added by Heiko Oberdiek . . . . .	21	\hyxmp@list@to@lines: Added this macro . . . . .	35
\hyxmp@xetex@crap: Added by Heiko Oberdiek . . . . .	22	\hyxmp@photometa@schema: Added this macro . . . . .	36
		\hyxmp@text@resource: Added this macro . . . . .	38
		\xmpcomma: Changed the default from \relax to an ordinary comma . . . . .	17
		\xmplinesep: Added this macro .	35

v2.3	\hyxmp@iptc@extensions: Gave the lptc4xmpCore:CreatorContactInfo fields a unique pdfaType:prefix to better support conversion of the document to PDF/A . . . . .	37	\hyxmp@parse@tz: Added this macro . . . . .	29
v2.3a	General: Bug fix: Redefine \@pdflang as \@empty when hyperref has set it to \relax .	15	\hyxmp@parse@tz@char: Added this macro . . . . .	29
v2.3b	\XMPTtruncateList: Made all definitions local to avoid spurious Too many unprocessed floats errors when running with memoir	17	\hyxmp@pdf@schema: Made \hyxmp@pdf@schema <i>always</i> include the Adobe PDF schema, even when empty. Florian Breitwieser noted that this is necessary for PDF/A-1b compliance	30
v2.4	General: Added support for the PDF/A Identification schema, as requested by Florian Breitwieser . . . . .	1	\hyxmp@pdf@to@xmp@date: Added this macro . . . . .	28
	\hyxmp@add@simple@var: Added this macro . . . . .	31	\hyxmp@pdfa@id@schema: Added this macro . . . . .	39
	\hyxmp@create@uuid: Modified this macro to produce a proper version 4 (random or pseudorandom) UUID . . . . .	26	\hyxmp@today: Modified the code to parse the time and timezone from \pdfcreationdate, as proposed by Florian Breitwieser .	29
	\hyxmp@dc@schema: Made dc:language a Bag instead of an individual item so as to conform to the latest XMP specifications, a detail identified by Florian Breitwieser . . . . .	32	\hyxmp@today@define: Added this macro . . . . .	29
	\hyxmp@parse@time: Added this macro . . . . .	28	\xmptilde: Added this macro . . .	17
			v2.5	
			General: Enabled “\_” to work within email addresses, as requested by Leonid Sinev . . . . .	1
			\hyxmp@add@to+xml: Updated also to replace underscores and to modify only the text being added, not the already-modified text . . . . .	27
			\hyxmp@textunderscore: Added this macro . . . . .	10
			\hyxmp@uscore: Added this macro	17

## 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			
\# . . . . .	512, 735	\@elt@rest . . . .	747, <u>749</u>
\& . . . . .	267, 299, 734	\@firstofone . . . . .	346
\@baseurl . . . . .	103, 710	\@firstoftwo . . . . .	258
\@elt . . . . .	<u>183</u> , <u>629</u> , <u>747</u> , <u>752</u>	\@gobble . . . . .	188, 343
\@elt@first . . . . .	<u>745</u>	\@pdfauthor . . . . .	
		\@pdfauthorhortitle . . .	
			25, 105, 716, 724
		\@pdfcaptionwriter .	
			27, 106, 716, 725
		\@pdfcontactaddress	
			31, 107, 762, 781

\@pdfcontactcity ...	CiAdrCtry ..... 2, 36	\hyxmp@add@simple ..
. <u>39</u> , 108, 763, 782	CiAdrExtadr ..... 2, 36	..... 575, <u>581</u> ,
\@pdfcontactcountry	CiAdrPcode ..... 2, 36	657, 684, 706–
. <u>45</u> , 109, 766, 785	CiAdrRegion ..... 2, 36	710, 724, 725,
\@pdfcontactemail ..	CiEmailWork ..... 2, 36	782–785, 869, 870
. <u>49</u> , 110, 768, 789	CiTelWork ..... 2, 36	\hyxmp@add@simple@var
\@pdfcontactphone ..	CiUrlWork ..... 2, 36	.... 572, 573, <u>591</u>
. <u>47</u> , 111, 767, 788		\hyxmp@add@to@xml ..
\@pdfcontactpostcode	<b>D</b>	..... <u>496</u> ,
. <u>43</u> , 112, 765, 784	Date ..... 29	517, 568, 577,
\@pdfcontactregion .	\day ..... 555, 556, 558	586, 595, 604,
. <u>41</u> , 113, 764, 783	dc:creator ..... 2, 6, 32	610, 614, 624,
\@pdfcontacturl ...	dc:date ..... 2, 32	632, 638, 645,
. <u>51</u> , 114, 769, 790	dc:description ..... 3, 32	658, 674, 680,
\@pdfcopyright . <u>21</u> ,	dc:format ..... 2	685, 693, 702,
115, 652, 668, 678	dc:language ..... 2, 32	711, 719, 728,
\@pdfcreator ..... 709	dc:rights ..... 2, 32	742, 746, 750,
\@pdfkeywords .. <u>79</u> , 116	dc:source ..... 2, 32	756, 774, 794,
\@pdflang .. 117, 132,	dc:subject ..... 2, 32	801, 842, 855,
133, 135, 138, 656	dc:title ..... 3, 32	865, 871, 891, 905
\@pdflicenseurl ...	\define@key .....	\hyxmp@append@hex ..
. <u>23</u> , 118, 664, 684	. 22, 24, 26, 28,	. <u>433</u> , 452–454, 458
\@pdfmetalang <u>29</u> , 136,	30, 32, 40, 42, 44,	\hyxmp@append@hex@iii
138, 140, 608, 611	46, 48, 50, 52, 62, 79	. <u>451</u> , 457, 467, 478
\@pdfsubject .. 119, 651	dvipdf (option) ..... 41	\hyxmp@append@hex@iv
\@pdftitle .....	dvipdfm ..... 42	..... <u>456</u> , 462,
. 120, 485, 491, 650	dvips (option) ..... 41	463, 465, 480–482
\@secondoftwo ..... 260	dvips ..... 6, 21	\hyxmp@at@end ... <u>3</u> , 141
\^ ..... 171, 175, 203,	dvipsone (option) ..... 41	\hyxmp@big@prime ...
505, 506, 883–885	dviwindo (option) .... 41	. <u>407</u> , 410, 420, 430
\_ ..... 504, 506		\hyxmp@big@prime@ii
\~ ..... 180, 337, 338	<b>E</b>	..... <u>407</u> , 429
	\EdefEscapeHex 226, 239	\hyxmp@bom .... <u>876</u> , 891
\_ ..... 303, 338, 504	\EdefUnescapeHex ... 243	\hyxmp@comma .....
	\EdefUnescapeString 213	... 33, 63, 80, <u>170</u>
<b>A</b>	ETX ..... 17, 19	\hyxmp@commas@to@list
ASCII ..... 10, 20	<b>G</b>	. <u>154</u> , 185, 630, 754
\AtBeginDocument ... 130	Ghostsript ..... 6	\hyxmp@commas@to@list@i
\AtEndDocument ..... 5	<b>H</b>	..... 156, <u>158</u>
\AtEndDvi ..... 8	\Hy@driver ..... 4,	\hyxmp@concat@metadata
atenddvi ..... 10	915, 919, 923, 928	..... <u>101</u>
Author ..... 6, 17	hyperref ... 1, 3–6, 9,	\hyxmp@construct@packet
<b>B</b>	10, 12–16, 30, 40, 41	..... <u>889</u> , 913
baseurl (option) 3, 9, 10, 34	\hypersetup ..... <u>96</u>	\hyxmp@count@non@spaces
BOM ..... 39	hyperxmp 1–6, 8–12, 15–	..... 997, <u>1008</u>
<b>C</b>	17, 19, 25, 30, 43, 44	\hyxmp@count@spaces
CiAdrCity ..... 2, 36	\hyxmp@@is@unicode . <u>247</u>	..... 996, <u>999</u>
		\hyxmp@crap@convert
		..... 329, <u>363</u>

\hyxmp@crap@result .	\hyxmp@one@token . . .	\hyxmp@set@rand@num
..... 319, 355	409, 413, 1000,	.... 426, 434, 469
\hyxmp@crap@test 326, 351	1001, 1009, 1010	\hyxmp@skiptorelax .
\hyxmp@create@uuid .	\hyxmp@padding 515, 908	..... 356, 362
.... 460, 488, 494	\hyxmp@parse@time . .	\hyxmp@skipzeros . . . 314
\hyxmp@dc@schema 644, 899	..... 529, 531	\hyxmp@SpaceOther . .
\hyxmp@def@DocumentID	\hyxmp@parse@tz . . .	..... 323, 336
..... 484, 691	.... 538, 541, 545	\hyxmp@string . . . . 581
\hyxmp@def@InstanceID	\hyxmp@parse@tz@char	\hyxmp@string@len . .
..... 490, 692	..... 533, 535	..... 979, 994
\hyxmp@DocumentID . .	\hyxmp@pdf@schema . .	\hyxmp@sublist . . . .
..... 484, 696	..... 567, 897	. 159, 160, 163, 164
\hyxmp@dq@code . 1, 1029	\hyxmp@pdf@to@xmp@date	\hyxmp@temp@list . . . 183
\hyxmp@driver . . . 3, 912	..... 527, 564	\hyxmp@temp@str . . . 183
\hyxmp@embed@packet	\hyxmp@pdfa@id@schema	\hyxmp@text . . . . .
..... 143, 912	..... 863, 903	. 211, 289, 319, 363
\hyxmp@embed@packet@dvi@pdfm	\hyxmp@pdfauthor . . .	\hyxmp@text@resource
..... 920, 978	..... 53, 62, 653	.... 834–841, 854
\hyxmp@embed@packet@pdfmark	\hyxmp@pdfkeywords .	\hyxmp@textunderscore
..... 932, 948	..... 53, 79, 654	..... 15
\hyxmp@embed@packet@pdf@tex	\hyxmp@pdfstringdef	\hyxmp@today . . . . .
..... 916, 938	.... 15, 22, 24,	. 491, 549, 551,
\hyxmp@embed@packet@xetex	26, 28, 30, 35, 40,	553, 556, 558,
..... 924, 1016	42, 44, 46, 48, 50, 52	561, 655, 706–708
\hyxmp@find@metadata	\hyxmp@photometa@data	\hyxmp@today@define
..... 101, 142	..... 760	..... 548, 562
\hyxmp@hash . . . . .	\hyxmp@photometa@schema	\hyxmp@toxml . . 241, 264
. 511, 804–807, 895	..... 760, 901	\hyxmp@toxml@unicodetex
\hyxmp@Hyp@pdfauthor 56	\hyxmp@photoshop@data	..... 229, 289
\hyxmp@Hyp@pdfkeywords	..... 715	\hyxmp@trimb . . 196, 199
..... 73	\hyxmp@photoshop@schema	\hyxmp@trimc . . 199, 200
\hyxmp@hypersetup . . 96	..... 715, 900	\hyxmp@trimspaces . .
\hyxmp@InstanceID . .	\hyxmp@ProcessKeyvalOptions	..... 163, 192
..... 490, 697	..... 91	\hyxmp@try . . . . . 319
\hyxmp@iptc@extensions	\hyxmp@rand@num 426,	\hyxmp@unicodetexfalse
..... 773, 800	435, 470, 487, 493	..... 202
\hyxmp@is@unicode . .	\hyxmp@rdf@dc . . . .	\hyxmp@unicodetextrue
.... 215, 232, 247	.... 600, 650–652	..... 202
\hyxmp@legal . . . . . 663	\hyxmp@redefine@Hyp	\hyxmp@uscore . . 17, 174
\hyxmp@list . . . . .	..... 55, 93, 98	\hyxmp@x@default . . .
. 630, 636, 754, 755	\hyxmp@reencode . . . 208	. 136, 566, 608, 615
\hyxmp@list@to@lines	\hyxmp@rights . . . .	\hyxmp@xetex@crap . .
. 738, 781, 788–790	. 663, 666, 670, 672	..... 220, 319
\hyxmp@list@to@xml .	\hyxmp@seed@rng . . .	\hyxmp@xml . . . . . 508,
.... 621, 653–656	.... 409, 486, 492	515, 889, 944,
\hyxmp@mm@schema 690, 904	\hyxmp@seed@rng@i . .	968, 979, 986, 1017
\hyxmp@modulo@a 401,	..... 411, 413	\hyxmp@xmlified . . .
420, 430, 436, 471	\hyxmp@seed@string .	. 211, 587, 596,
\hyxmp@new@xml 507, 508	. 485, 486, 491, 492	611, 615, 630, 754
\hyxmp@num . . . . . 363		

<code>\hyxmp@xmlify</code> . . . . .	<code>ngerman</code> . . . . . 9	<code>pdfmark</code> (option) . . . . . 41
. 140, <u>211</u> , 585,	<code>\number</code> 366, 368, 370,	<code>\pdfmark</code> . . 949, 952,
594, 603, 629, 753	375, 377, 382, 384	956, 966, 970, 974
<code>\hyxmp@xmp@basic@schema</code>	<b>P</b>	<code>pdfmetalang</code> (option) . . 4
. . . . . 701, 902	<code>\PackageWarningNoLine</code>	<code>\pdfminorversion</code> . . 575
<code>\hyxmp@xmpRights@schema</code>	. . . . 123, 146, 927	<code>\pdfobj</code> . . . . . 941
. . . . . 662, 898	PDF 1–3, 5–8, 15, 16, 21,	<code>pdfproducer</code> (option) 3, 10
<code>\hyxmp@zero</code> . . . . 372,	27, 28, 30, 39, 40, 43	<code>\pdfstringdef</code> . . . . 18
379, 386, 392, <u>397</u>	PDF/A . . 3, 27, 30, 37, 39	<code>pdfsubject</code> (option) . .
<b>I</b>	<code>pdf:Keywords</code> . . . . . 2, 30	. . . . . 3, 10, 32
IETF . . . . . 4	<code>pdf:PDFVersion</code> . . . . 3, 30	<code>pdfTeX</code> . . 10, 21, 41, 43
<code>\ifHy@pdfa</code> . . . . . 864	<code>pdf:Producer</code> . . . . . 3, 30	<code>pdftitle</code> (option) 4, 10, 32
<code>\ifhyxmp@unicodetex</code>	<code>pdfaid:conformance</code> . . . 3	<code>photoshop:AuthorsPosition</code>
. . . . <u>202</u> , 214, 877	<code>pdfaid:part</code> . . . . . 3	. . . . . 3, 34
<code>ifxetex</code> . . . . . 10	<code>pdfauthor</code> (option) . . .	<code>photoshop:CaptionWriter</code>
<code>\ifxetex</code> . . . . . 219	3, 8, 10, 12, 13, 32	. . . . . 2, 34
<code>Info</code> . . . . . 6	<code>pdfauthor:tile</code> (option) 4, 9	<code>PI</code> . . . . . 27
<code>intcalc</code> . . . . . 10	<code>pdfcaptionwriter</code> (op-	<code>\ProcessKeyvalOptions</code>
<code>\intcalcDiv</code> 368, 375, 382	tion) . . . . . 4	. . . . . <u>91</u>
<code>\intcalcMod</code> 370, 377, 384	<code>\pdfcatalog</code> . . . . . 945	<code>Producer</code> . . . . . 30
<code>IPTC</code> . . . . 6, 11, 27, 36, 37	<code>\pdfcompresslevel</code> . . 940	<code>ps2pdf</code> (option) . . . . . 41
<code>lptc4xmpCore:CreatorContactInfo</code>	<code>pdfcontactaddress</code> (op-	<b>Q</b>
. . . . . 2, 36	tion) . . . . 4, 6, 7	<code>\Q</code> . . . . . 192, 201
<code>ISO</code> . . . . . 11	<code>pdfcontactcity</code> (option) . 4	<b>R</b>
<b>J</b>	<code>pdfcontactcountry</code> (op-	<code>rdf:li</code> . . . . . 2
<code>\jobname</code> . . 124, 147,	tion) . . . . . 4	<code>rdf:Seq</code> . . . . . 2
485, 491, 657, 929	<code>pdfcontactemail</code> (op-	<code>\renewcommand</code> . . . . 92
<b>K</b>	tion) . . . . . 4	<code>\RequirePackage</code> 7, 10–14
<code>Keywords</code> . . . . . 6, 30	<code>pdfcontactphone</code> (op-	<b>S</b>
<code>\KV@Hyp@pdfauthor</code> . . <u>62</u>	tion) . . . . . 4	<code>\SE-&gt;pdfdoc@03</code> . . . . <u>209</u>
<code>\KV@Hyp@pdfkeywords</code> <u>79</u>	<code>pdfcontactpostcode</code> (op-	<code>\SE-&gt;pdfdoc@15</code> . . . . <u>210</u>
<code>kvoptions</code> . . . . . 10, 14	tion) . . . . . 4	<code>\special</code> . . . . .
<b>L</b>	<code>pdfcontactregion</code> (op-	980, 988, 1017, 1023
<code>LF</code> . . . . . 35	tion) . . . . . 4, 9	<code>stringenc</code> . . . . . 10
<code>LuaL<sup>A</sup>TeX</code> . . . . . 6, 7	<code>pdfcopyright</code> (option) .	<code>\StringEncodingConvert</code>
<code>LuaTeX</code> . . . . . 19, 22, 43	. . . . . 4, 32, 33	. . . . . 216,
<b>M</b>	<code>\pdfcreationdate</code> . . 564	222, 233, 236, 331
<code>Metadata</code> . . . . . 6, 40, 43	<code>PDFDocEncoding</code> . . .	<code>Subject</code> . . . . . 6
<code>\month</code> . . . . 550, 551, 553	. . . . . 12, 19, 20	<b>T</b>
<b>N</b>	<code>pdfescape</code> . . . . . 10	<code>T<sub>E</sub>X</code> . . . . . 19,
<code>NAK</code> . . . . . 10, 17, 19	<code>pdfkeywords</code> (option) .	21, 22, 24, 34, 42, 43
<code>nativepdf</code> (option) . . . . 41	3, 8, 10, 12, 13, 32	<code>Text</code> . . . . . 38
<code>\newif</code> . . . . . 202	<code>pdflang</code> (option) . . . .	<code>\textunderscore</code> . . .
<code>\next</code> . . . . . <u>158</u> , <u>413</u>	. . . 3, 4, 10, 15, 32	. . . . . 16, 17, 19
	<code>\pdflastobj</code> . . . . . 945	<code>textures</code> (option) . . . . 41
	<code>pdfL<sup>A</sup>TeX</code> . . . . . 3, 5	<code>Title</code> . . . . . 6
	<code>pdflicenseurl</code> (option) .	
	. . . . . 4, 9, 33	

<b>U</b>		
Unicode	9, 10, 19–23, 32, 36, 39, 43	
unicode (option)	9	
URL	2, 4, 9, 11, 12, 33, 34, 37	
UTF-16BE	20	
UTF-32BE	19, 20	
UTF-8	20	
UUID	24, 26, 27	
<b>V</b>		
\vfuzz	200	
vtexpdfmark (option)	41	
<b>X</b>		
\x	319	
<b>Y</b>		
xdvipdfmx	7, 43	
X <sub>g</sub> LaTeX	6, 7	
X <sub>g</sub> TeX	10, 19, 22, 43	
XML	1, 2, 6, 16, 19–22, 27, 30–32, 35, 37, 39	
XMP	1–3, 6–10, 15, 17, 18, 21, 24, 27–31, 33, 34, 37, 41–43	
xmp:BaseURL	2	
xmp:CreateDate	2	
xmp:CreatorTool	3	
xmp:MetadataDate	2	
xmp:ModifyDate	2	
\xmpcomma		
	33, 36, 62, 79, 169	
xmpincl	3	
\xmplinesep	733, 750, 786	
xmpMM:DocumentID		
	2, 24, 34	
xmpMM:InstanceID		
	2, 24, 34	
\xmpquote		
	34, 37, 62, 79, 178	
xmpRights:Marked	2, 33	
xmpRights:WebStatement		
	2, 33	
\xmptilde	179	
\XMPTruncateList	183	
\year	549	