

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

Scott Pakin  
scott+hyxmp@pakin.org

January 2, 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` v2.4, dated 2014/01/02.

```

</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,

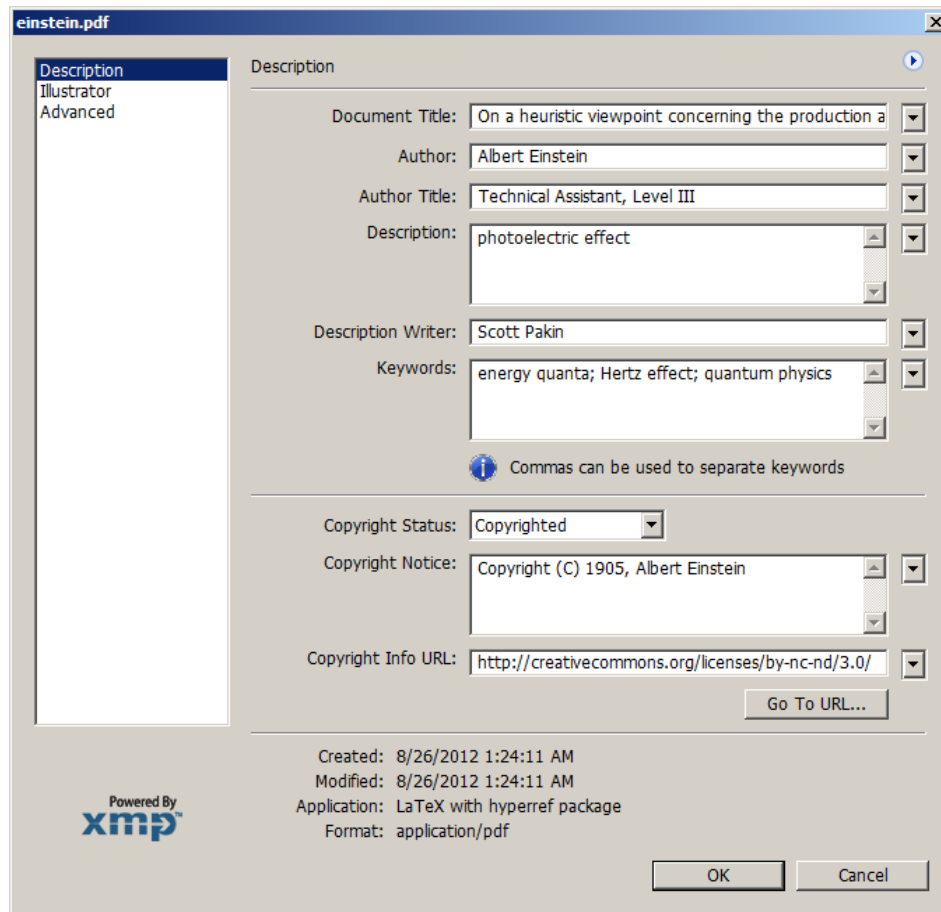


Figure 1: XMP metadata as it appears in Adobe Acrobat

`\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

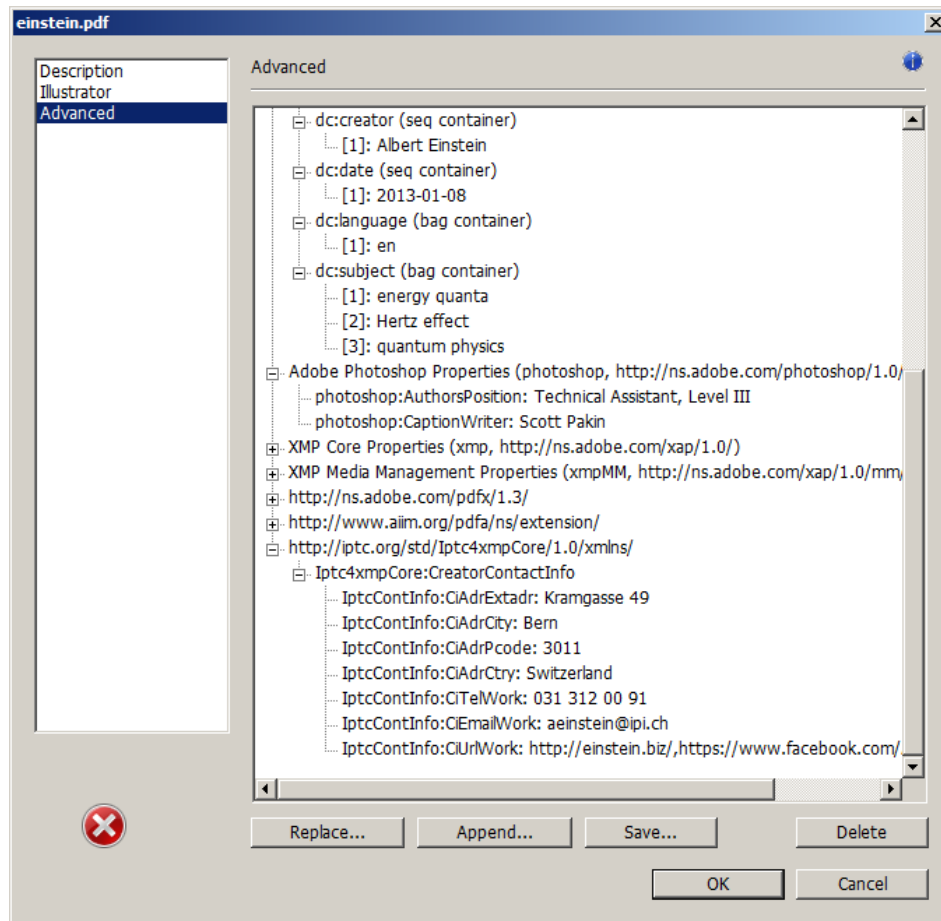


Figure 2: Additional XMP metadata as it appears in Adobe Acrobat

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 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`.

## 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, the standard `\AtEndDocument` works well enough. For all the other backends we use `\AtEndDvi` from the `atenddvi` package, which is more robust but requires an additional L<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}
```

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

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

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

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

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

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

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

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

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

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

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.

```
25 \def\@pdfcontactaddress{}
26 \define@key{Hyp}{pdfcontactaddress}{%
27   \let\xmpcomma=\hyxmpcomma
28   \def\xmpquote##1{##1}%
29   \pdfstringdef\@pdfcontactaddress{#1}%
30   \def\xmpcomma{,}%
31   \let\xmpquote=\relax
32 }
```

`\@pdfcontactcity` Prepare to store the city of the document’s contact person/institution.

```
33 \def\@pdfcontactcity{}
34 \define@key{Hyp}{pdfcontactcity}{\pdfstringdef\@pdfcontactcity{#1}}
```

`\@pdfcontactregion` Prepare to store the state or province of the document’s contact person/institution.

```
35 \def\@pdfcontactregion{}
36 \define@key{Hyp}{pdfcontactregion}{\pdfstringdef\@pdfcontactregion{#1}}
```

`\@pdfcontactpostcode` Prepare to store the postal code of the document’s contact person/institution.

```
37 \def\@pdfcontactpostcode{}
38 \define@key{Hyp}{pdfcontactpostcode}{\pdfstringdef\@pdfcontactpostcode{#1}}
```

`\@pdfcontactcountry` Prepare to store the country of the document’s contact person/institution.

```
39 \def\@pdfcontactcountry{}
40 \define@key{Hyp}{pdfcontactcountry}{\pdfstringdef\@pdfcontactcountry{#1}}
```

`\@pdfcontactphone` Prepare to store the telephone number of the document’s contact person/institution.

```
41 \def\@pdfcontactphone{}
42 \define@key{Hyp}{pdfcontactphone}{\pdfstringdef\@pdfcontactphone{#1}}
```

`\@pdfcontactemail` Prepare to store the email address of the document's contact person/institution.

```

43 \def\@pdfcontactemail{}
44 \define@key{Hyp}{pdfcontactemail}{\pdfstringdef\@pdfcontactemail{#1}}

```

`\@pdfcontacturl` Prepare to store the URL of the document's contact person/institution.

```

45 \def\@pdfcontacturl{}
46 \define@key{Hyp}{pdfcontacturl}{\pdfstringdef\@pdfcontacturl{#1}}

```

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

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

`\hyxmp@pdfauthor` Prepare to store the name of the author and a list of keywords.

`\hyxmp@pdfkeywords`

```

47 \def\hyxmp@pdfauthor{}
48 \def\hyxmp@pdfkeywords{}

```

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

```

49 \newcommand*{\hyxmp@redefine@Hyp}{%

```

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

```

50 \ifundefined{KV@Hyp@pdfauthor}{\}%
51 \ifundefined{hyxmp@Hyp@pdfauthor}{%
52 \expandafter\let\expandafter\hyxmp@Hyp@pdfauthor
53 \csname KV@Hyp@pdfauthor\endcsname
54 }{\}%
55 }%

```

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

for use in unstructured lists (those in which the entire list appears within a single pair of tags).

```

56 \define@key{Hyp}{pdfauthor}{%
57   \let\xmpcomma=\hyxmp@comma
58   \def\xmpquote####1{####1}%
59   \hyxmp@Hyp@pdfauthor{##1}%
60   \global\let\hyxmp@pdfauthor=\@pdfauthor
61   \def\xmpcomma{,%
62   \def\xmpquote####1{"####1"%
63   \hyxmp@Hyp@pdfauthor{##1}%
64   \def\xmpcomma{,%
65   \let\xmpquote=\relax
66 }%

```

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

```

67 \@ifundefined{KV@Hyp@pdfkeywords}{}%
68   \@ifundefined{hyxmp@Hyp@pdfkeywords}{%
69     \expandafter\let\expandafter\hyxmp@Hyp@pdfkeywords
70     \csname KV@Hyp@pdfkeywords\endcsname
71   }{}%
72 }%

```

`\KV@Hyp@pdfkeywords` Redefine `\KV@Hyp@pdfkeywords` to process its argument twice. The first time, `\xmpcomma` is defined as a placeholder character (`\hyxmp@comma`) and `\xmpquote` as the identity function. The result is stored in `\hyxmp@pdfkeywords` for use in structured lists (those surrounding each entry with `<rdf:li>`). The second time, `\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).

```

73 \define@key{Hyp}{pdfkeywords}{%
74   \let\xmpcomma=\hyxmp@comma
75   \def\xmpquote####1{####1}%
76   \hyxmp@Hyp@pdfkeywords{##1}%
77   \global\let\hyxmp@pdfkeywords=\@pdfkeywords
78   \def\xmpcomma{,%
79   \def\xmpquote####1{"####1"%
80   \hyxmp@Hyp@pdfkeywords{##1}%
81   \def\xmpcomma{,%
82   \let\xmpquote=\relax
83 }%
84 }

```

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

```

85 \let\hyxmp@ProcessKeyvalOptions=\ProcessKeyvalOptions
86 \renewcommand*{\ProcessKeyvalOptions}{%

```

```

87 \hyxmp@redefine@Hyp
88 \hyxmp@ProcessKeyvalOptions
89 }

\hyxmp@hypersetup  Redefine hyperref's \hypersetup command to invoke \hyxmp@redefine@Hyp before
\hypersetup        performing its normal option processing.
90 \let\hyxmp@hypersetup=\hypersetup
91 \def\hypersetup{%
92   \hyxmp@redefine@Hyp
93   \hyxmp@hypersetup
94 }

\hyxmp@find@metadata  Issue a warning message if the author failed to specify any metadata at all. This
\hyxmp@concat@metadata 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.
95 \newcommand*{\hyxmp@find@metadata}{%
96   \edef\hyxmp@concat@metadata{%
97     \@baseurl
98     \@pdfauthor
99     \@pdfauthortitle
100    \@pdfcaptionwriter
101    \@pdfcontactaddress
102    \@pdfcontactcity
103    \@pdfcontactcountry
104    \@pdfcontactemail
105    \@pdfcontactphone
106    \@pdfcontactpostcode
107    \@pdfcontactregion
108    \@pdfcontacturl
109    \@pdfcopyright
110    \@pdfkeywords
111    \@pdflang
112    \@pdflicenseurl
113    \@pdfsubject
114    \@pdftitle
115   }%
116   \ifx\hyxmp@concat@metadata\@empty
117     \PackageWarningNoLine{hyperxmp}{%
118 \jobname.tex did not specify any metadata to\MessageBreak
119 include in the XMP packet.\space\space Please see the\MessageBreak
120 hyperxmp documentation for instructions on how to\MessageBreak
121 provide metadata values to hyperxmp}%
122   \fi
123 }

```

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.

```
124 \AtBeginDocument{%
125   \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`.

```
126   \ifx\pdflang\relax
127     \let\pdflang=\empty
128   \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.

```
129   \ifx\pdflang\empty
130     \let\pdfmetlang=\hyxmp@xdefault
131   \else
132     \edef\pdfmetlang{\pdflang}%
133   \fi
134   \hyxmp@xmlify\pdfmetlang
```

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`.

```
135   \hyxmp@at@end{%
136     \hyxmp@find@metadata
137     \hyxmp@embed@packet
138   }%
139 }%
140 {\PackageWarningNoLine{hyperxmp}{%
141 \jobname.tex failed to include a\MessageBreak
142 \string\usepackage\string{hyperref}\string}
143 in the preamble.\MessageBreak
144 Consequently, all hyperxmp functionality will be\MessageBreak
145 disabled}%
146 }%
147 }
```

### 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.)
148 \newcommand*{\hyxmp@commas@to@list}[2]{%
149   \gdef#1{%
150     \expandafter\hyxmp@commas@to@list@i\expandafter#1#2,,%
151   }

\hyxmp@commas@to@list@i  Recursively construct macro #1 from comma-separated list #2. Stop if #2 is empty.
                        \next
152 \def\hyxmp@commas@to@list@i#1#2,{%
153   \gdef\hyxmp@sublist{#2}%
154   \ifx\hyxmp@sublist@empty
155     \let\next=\relax
156   \else
157     \hyxmp@trimspaces\hyxmp@sublist
158     \@cons{#1}{\hyxmp@sublist}%
159     \def\next{\hyxmp@commas@to@list@i{#1}}%
160   \fi
161   \next
162 }

\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.
163 \def\xmpcomma{,}%

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

164 \bgroup
165 \catcode'\^C=11
166 \gdef\hyxmp@comma{^C}
167 \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.

```
168 \let\xmpquote=\relax
```

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

```
169 \bgroup
170 \catcode'\~=12%
171 \gdef\xmptilde{~}%
172 \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.

```
173 \newcommand{\XMPTruncateList}[1]{%
174 \edef\hyxmp@temp@str{\csname hyxmp@#1\endcsname}%
175 \hyxmp@commas@to@list{\hyxmp@temp@list}{\hyxmp@temp@str}%
176 \def\@elt##1{%
177 \expandafter\gdef\csname @#1\endcsname{##1}%
178 \let\@elt=\@gobble
179 }
180 \hyxmp@temp@list
181 }}
```

### 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.

```
182 \catcode'\Q=3
\hyxmp@trimspaces\x redefines \x to have the same replacement text sans leading
and trailing space tokens.
183 \newcommand{\hyxmp@trimspaces}[1]{%
Use grouping to emulate a multi-token afterassignment queue.
184 \begingroup
Put “\toks 0 {” into the afterassignment queue.
185 \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.

```
186 \expandafter\hyxmp@trimb\expandafter\noexpand#1Q Q}%
```

Transfer the trimmed text back into #1.

```
187 \edef#1{\the\toks0}%
```

```
188 }
```

`\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.

```
189 \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.

```
190 \def\hyxmp@trimc#1Q#2{\afterassignment\endgroup \vfuzz\the\vfuzz#1}
```

```
191 \catcode'\Q=11
```

### 3.3.3 Converting text to XML

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

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

```
192 \newif\ifhyxmp@unicodetex
```

```
193 \ifnum64='^^^^0040\relax
```

```
194 \hyxmp@unicodetextrue
```

```
195 \else
```

```
196 \hyxmp@unicodetexfalse
```

```
197 \fi
```

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

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

`\SE->pdfdoc003` Preserve `ETX` (`^^C`), which is normally an invalid character in `PDFDocEncoding`. We use it in `hyperxmp` (and specifically in `\hyxmp@xmlify` below) as a list-element separator.

```
199 \expandafter\def\csname SE->pdfdoc003\endcsname{0003}
```

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

```

200 \newcommand*{\hyxmp@xmlify}[1]{%
201   \gdef\hyxmp@xmlified{%
    Escaped PDF string → PDFDocEncoding/Unicode
202   \EdefUnescapeString\hyxmp@text{#1}%
203   \ifhyxmp@unicodetex
    PDFDocEncoding/Unicode → UTF-32BE
204     \hyxmp@is@unicode\hyxmp@text{%
205       \StringEncodingConvert
206       \hyxmp@text\hyxmp@text{utf16be}{utf32be}%
207     }{%
208       \ifxetex
209         \hyxmp@xetex@crap
210       \else
211         \StringEncodingConvert
212         \hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
213       \fi
214     }%
    UTF-32BE → UTF-32BE as hex string
215     \EdefEscapeHex\hyxmp@text{\hyxmp@text}%
    UTF-32BE → XML in ASCII
216     \edef\hyxmp@text{%
217       \expandafter
218     }\expandafter\hyxmp@toxml@unicodetex\hyxmp@text
219     \relax\relax\relax\relax\relax\relax\relax\relax
220   \else
    PDFDocEncoding/Unicode → UTF-8
221     \hyxmp@is@unicode\hyxmp@text{%
222       \StringEncodingConvert
223       \hyxmp@text\hyxmp@text{utf16be}{utf8}%
224     }{%
225       \StringEncodingConvert
226       \hyxmp@text\hyxmp@text{pdfdoc}{utf8}%
227     }%
    UTF-8 → UTF-8 as hex string
228     \EdefEscapeHex\hyxmp@text{\hyxmp@text}%
    UTF-8 as hex string → XML in UTF-8 as hex string
229     \edef\hyxmp@text{%
230       \expandafter\hyxmp@toxml\hyxmp@text\@empty\@empty
231     }%
  
```

XML in UTF-8 as hex string  $\rightarrow$  XML in UTF-8

```

232 \EdefUnescapeHex\hyxmp@text{\hyxmp@text}%
233 \fi
234 \global\let\hyxmp@xmlified\hyxmp@text
235 }

```

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

```

236 \begingroup
237 \lccode'\<=254 %
238 \lccode'\>=255 %
239 \catcode254=12 %
240 \catcode255=12 %
241 \lowercase{\endgroup
242 \def\hyxmp@is@unicode#1{%
243 \expandafter\hyxmp@@is@unicode#1<>\@nil
244 }%
245 \def\hyxmp@@is@unicode#1<>#2\@nil{%
246 \ifx\#1\%
247 \expandafter\@firstoftwo
248 \else
249 \expandafter\@secondoftwo
250 \fi
251 }%
252 }

```

`\hyxmp@toxml` Replace the characters “<”, “&”, and “>” with XML entities when using a non-native-Unicode  $\text{\TeX}$  ( $\text{\TeX}$  or  $\text{pdf\TeX}$ ).

```

253 \def\hyxmp@toxml#1#2{%
254 \ifx#1\@empty
255 \else
256 \ifnum"#1#2='\& %
257 26616D703B% &#1;
258 \else\ifnum"#1#2='\< %
259 266C743B% <#1;
260 \else\ifnum"#1#2='\> %
261 2667743B% >#1;
262 \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.

```

263     \@ifundefined{pdfmark}{%
264         #1#2%
265     }{%
266         \ifnum"#1#2='\( %
267             5C28% \(\
268         \else\ifnum"#1#2='\) %
269             5C29% \)
270         \else
271             #1#2%
272         \fi\fi
273     }%
274     \fi\fi\fi
275     \expandafter\hyxmp@toxml
276 \fi
277 }

```

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

`\hyxmp@text`

```

278 \def\hyxmp@toxml@unicodetex#1#2#3#4#5#6#7#8{%
279     \ifx#1\relax
280     \else
281         \ifnum"#1#2#3#4#5#6#7#8>127 %
282             \uccode'\*="#1#2#3#4#5#6#7#8\relax
283             \uppercase{%
284                 \edef\hyxmp@text{\hyxmp@text *}%
285             }%
286         \else\ifnum"#7#8='\< %
287             \edef\hyxmp@text{\hyxmp@text &lt;}%
288         \else\ifnum"#7#8='\& %
289             \edef\hyxmp@text{\hyxmp@text &amp;}%
290         \else\ifnum"#7#8='\> %
291             \edef\hyxmp@text{\hyxmp@text &gt;}%
292         \else\ifnum"#7#8='\ %
293             \edef\hyxmp@text{\hyxmp@text\space}%
294         \else
295             \uccode'\*="#7#8\relax
296             \uppercase{%
297                 \edef\hyxmp@text{\hyxmp@text *}%
298             }%
299         \fi\fi\fi\fi\fi
300         \expandafter\hyxmp@toxml@unicodetex
301     \fi
302 }

```

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

```

303 \def\hyxmp@skipzeros#1{%
304     \ifx#10%

```

```

305     \expandafter\hyxmp@skipzeros
306   \fi
307 }

\hyxmp@xetex@crap  \x  In the case of XeTeX, the strings defined by \pdfstringdef can contain big
\hyxmp@xetex@crap  characters. In this case, the string is treated as Unicode.
\hyxmp@try 308 \begingroup
\hyxmp@crap@result 309 \def\x#1{\endgroup
\hyxmp@text 310   \def\hyxmp@xetex@crap{%
311     \edef\hyxmp@try{%
312       \expandafter\hyxmp@SpaceOther\hyxmp@text#1\@nil
313     }%
314     \let\hyxmp@crap@result=N%
315     \expandafter\hyxmp@crap@test\hyxmp@try\relax
316     \ifx\hyxmp@crap@result Y%
317       \let\hyxmp@text\@empty
318       \expandafter\hyxmp@crap@convert\hyxmp@try\relax
319     \else
320       \StringEncodingConvert\hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
321     \fi
322   }%
323 }
324 \x{ }

\hyxmp@SpaceOther  Re-encode all spaces in a string with category code 12 (“other”).
325 \begingroup
326   \catcode'\~=12 %
327   \lccode'\~=\' %
328   \lowercase{\endgroup
329   \def\hyxmp@SpaceOther#1 #2\@nil{%
330     #1%
331     \ifx\relax#2\relax
332       \expandafter\@gobble
333     \else
334       ~%
335       \expandafter\@firstofone
336     \fi
337     {\hyxmp@SpaceOther#2\@nil}%
338   }%
339 }

\hyxmp@crap@test  Determine if we need to treat a string as Unicode.
340 \def\hyxmp@crap@test#1{%
341   \ifx#1\relax
342   \else
343     \ifnum'#1>127 %
344       \let\hyxmp@crap@result=Y%
345       \expandafter\expandafter\expandafter\hyxmp@skiptorelax
346     \else

```

```

347     \expandafter\expandafter\expandafter\hyxmp@crap@test
348     \fi
349     \fi
350 }

```

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

```

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

```

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

```

\hyxmp@num 352 \def\hyxmp@crap@convert#1{%
\hyxmp@text 353   \ifx#1\relax
354   \else
355     \edef\hyxmp@num{\number'#1}%
356     \ifnum\hyxmp@num>"FFFFFF %
357       \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"1000000}\relax
358       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
359       \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"1000000}}%
360     \else
361       \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
362     \fi
363     \ifnum\hyxmp@num>"FFFF %
364       \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"10000}\relax
365       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
366       \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"10000}}%
367     \else
368       \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
369     \fi
370     \ifnum\hyxmp@num>"FF %
371       \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"100}\relax
372       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
373       \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"100}}%
374     \else
375       \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
376     \fi
377     \ifnum\hyxmp@num>0 %
378       \lccode'\!=\hyxmp@num\relax
379       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
380     \else
381       \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
382     \fi
383     \expandafter\hyxmp@crap@convert
384   \fi
385 }

```

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

```

386 \begingroup
387   \catcode0=12 %
388   \gdef\hyxmp@zero{^^00}%
389 \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 T<sub>E</sub>X and is good enough for producing the XMP xmpMM:DocumentID and xmpMM:InstanceID fields.

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

```

390 \def\hyxmp@modulo@a#1{%
391   \@tempcntb=\@tempcnta
392   \divide\@tempcntb by #1
393   \multiply\@tempcntb by #1
394   \advance\@tempcnta by -\@tempcntb
395 }

```

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

```

\hyxmp@big@prime@ii 396 \def\hyxmp@big@prime{536870923}
397 \def\hyxmp@big@prime@ii{536870027}

```

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

```

\hyxmp@one@token 398 \def\hyxmp@seed@rng#1{%
399   \@tempcnta=\hyxmp@big@prime
400   \futurelet\hyxmp@one@token\hyxmp@seed@rng@i#1\@empty
401 }

```

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

```

\hyxmp@one@token \next 402 \def\hyxmp@seed@rng@i{%
403   \ifx\hyxmp@one@token\@empty
404     \let\next=\relax
405   \else
406     \def\next##1{%
407       \multiply\@tempcnta by 3
408       \advance\@tempcnta by '##1
409       \hyxmp@modulo@a{\hyxmp@big@prime}%
410       \futurelet\hyxmp@one@token\hyxmp@seed@rng@i
411     }%
412   \fi
413   \next
414 }

```

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

```

\hyxmp@rand@num 415 \def\hyxmp@set@rand@num{%
416   \@tempcnta=\hyxmp@rand@num
417   \multiply\@tempcnta by 3
418   \advance\@tempcnta by \hyxmp@big@prime@ii

```



```

419 \hyxmp@modulo@a{\hyxmp@big@prime}%
420 \xdef\hyxmp@rand@num{\the\@tempcnta}%
421 }

```

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

```

422 \def\hyxmp@append@hex#1{%
423   \hyxmp@set@rand@num
424   \@tempcnta=\hyxmp@rand@num
425   \hyxmp@modulo@a{16}%
426   \ifnum\@tempcnta<10
427     \xdef#1{#1\the\@tempcnta}%
428   \else

```

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

```

429     \advance\@tempcnta by -10
430     \ifcase\@tempcnta
431       \xdef#1{#1a}%
432     \or\xdef#1{#1b}%
433     \or\xdef#1{#1c}%
434     \or\xdef#1{#1d}%
435     \or\xdef#1{#1e}%
436     \or\xdef#1{#1f}%
437   \fi
438 \fi
439 }

```

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

```

440 \def\hyxmp@append@hex@iii#1{%
441   \hyxmp@append@hex#1%
442   \hyxmp@append@hex#1%
443   \hyxmp@append@hex#1%
444 }

```

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

```

445 \def\hyxmp@append@hex@iv#1{%
446   \hyxmp@append@hex@iii#1%
447   \hyxmp@append@hex#1%
448 }

```

`\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`.

```

449 \def\hyxmp@create@uuid#1{%
450   \def#1{uuid:}%
451   \hyxmp@append@hex@iv#1%
452   \hyxmp@append@hex@iv#1%

```

```

453 \g@addto@macro#1{-}%
454 \hyxmp@append@hex@iv#1%
455 \g@addto@macro#1{-4}%
456 \hyxmp@append@hex@iii#1%
457 \g@addto@macro#1{-}%

```

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

```

458 \hyxmp@set@rand@num
459 \@tempcnta=\hyxmp@rand@num
460 \hyxmp@modulo@a{4}%
461 \ifcase\@tempcnta
462   \g@addto@macro#1{8}%
463   \or\g@addto@macro#1{9}%
464   \or\g@addto@macro#1{a}%
465   \or\g@addto@macro#1{b}%
466 \fi
467 \hyxmp@append@hex@iii#1%
468 \g@addto@macro#1{-}%
469 \hyxmp@append@hex@iv#1%
470 \hyxmp@append@hex@iv#1%
471 \hyxmp@append@hex@iv#1%
472 }

```

`\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.

```

473 \newcommand*{\hyxmp@def@DocumentID}{%
474   \edef\hyxmp@seed@string{\jobname:\@pdftitle:\@pdfauthor}%
475   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
476   \edef\hyxmp@rand@num{\the\@tempcnta}%
477   \hyxmp@create@uuid\hyxmp@DocumentID
478 }

```

`\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.

```

479 \newcommand*{\hyxmp@def@InstanceID}{%
480   \edef\hyxmp@seed@string{\jobname:\@pdftitle:\@pdfauthor:\hyxmp@today}%
481   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
482   \edef\hyxmp@rand@num{\the\@tempcnta}%
483   \hyxmp@create@uuid\hyxmp@InstanceID
484 }

```

### 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.

```

485 \newcommand*{\hyxmp@add@to+xml}[1]{%
486   \bgroup
487   \@tempcnta=0
488   \loop
489     \lccode\@tempcnta=\@tempcnta
490     \advance\@tempcnta by 1
491     \ifnum\@tempcnta<256
492       \repeat
493       \lccode'\_='\ \relax
494       \lccode'\^C='\,\relax
495       \lowercase{\xdef\hyxmp+xml{\hyxmp+xml#1}}%
496   \egroup
497 }
```

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

```

498 \bgroup
499 \catcode'\#=11
500 \gdef\hyxmp@hash{#}
501 \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.

```

502 \bgroup
503 \xdef\hyxmp+xml{%
504   \hyxmp@add@to+xml{%
505     ----- ^^J%
506   }
507   \xdef\hyxmp@padding{\hyxmp+xml}%
508 \egroup
509 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
510 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
511 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
512 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
513 \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:20140102225100-07'00') to XMP's YYYY-MM-DDThh:mm:ss+TT:tt format (e.g., 2014-01-02T22:51:00-07:00) [4]. This macro is fully expandable.

```

514 \def\hyxmp@pdf@to@xmp@date#1:#2#3#4#5#6#7#8#9{%
515   #2#3#4#5-#6#7-#8#9%
516   \hyxmp@parse@time
517 }

```

`\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`.

```

518 \def\hyxmp@parse@time#1#2#3#4#5#6{%
519   T#1#2:#3#4:#5#6%
520   \hyxmp@parse@tz@char
521 }

```

`\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).

```

522 \def\hyxmp@parse@tz@char#1{%
523   #1%
524   \ifx#1-%
525     \expandafter\hyxmp@parse@tz
526   \else
527     \ifx#1+%
528       \expandafter\hyxmp@parse@tz
529     \fi
530   \fi
531 }

```

`\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.

```

532 \def\hyxmp@parse@tz#1'#2' {%
533   #1:#2%
534 }

```

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

```

535 \def\hyxmp@today@define{%
536   \xdef\hyxmp@today{\the\year}%
537   \ifnum\month<10
538     \xdef\hyxmp@today{\hyxmp@today-0\the\month}%

```

```

539 \else
540   \xdef\hyxmp@today{\hyxmp@today-\the\month}%
541 \fi
542 \ifnum\day<10
543   \xdef\hyxmp@today{\hyxmp@today-0\the\day}%
544 \else
545   \xdef\hyxmp@today{\hyxmp@today-\the\day}%
546 \fi
547 }

```

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

```

548 \expandafter\ifx\csname pdfcreationdate\endcsname\relax
549 \hyxmp@today@define
550 \else
551   \edef\hyxmp@today{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
552 \fi

```

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

```

553 \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.

```

554 \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.

```

555   \hyxmp@add@to@xml{%
556   -----<rdf:Description rdf:about=""^^J%
557   -----xmlns:pdf="http://ns.adobe.com/pdf/1.3/">^^J%
558   }%
559   \hyxmp@add@simple@var{pdf:Keywords}{@pdfkeywords}%
560   \hyxmp@add@simple@var{pdf:Producer}{@pdfproducer}%
561   \@ifundefined{pdfminorversion}{}{%
562     \hyxmp@add@simple{pdf:PDFVersion}{1.\the\pdfminorversion}%
563   }%
564   \hyxmp@add@to@xml{%
565   -----</rdf:Description>^^J%
566   }%
567 }

```

`\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.

```

568 \newcommand*{\hyxmp@add@simple}[2]{%
569   \edef\hyxmp@string{#2}%
570   \ifx\hyxmp@string\@empty
571     \else
572       \hyxmp@xmlify{\hyxmp@string}%
573       \hyxmp@add@to@xml{%
574         <#1>\hyxmp@xmlified</#1>^^J%
575       }%
576   \fi
577 }
```

`\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.

```

578 \newcommand*{\hyxmp@add@simple@var}[2]{%
579   \expandafter\ifx\csname#2\endcsname\relax
580   \else
581     \hyxmp@xmlify{\csname#2\endcsname}%
582     \hyxmp@add@to@xml{%
583       <#1>\hyxmp@xmlified</#1>^^J%
584     }%
585   \fi
586 }
```

### 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.

```

587 \newcommand*{\hyxmp@rdf@dc}[2]{%
588   \ifx#2\@empty
589     \else
590       \hyxmp@xmlify{#2}%
591       \hyxmp@add@to@xml{%
592         <dc:#1>^^J%
593         <rdf:Alt>^^J%
594       }%
595       \ifx\@pdfmetalang\hyxmp@x@default
596       \else
597         \hyxmp@add@to@xml{%
598           <rdf:li xml:lang="\@pdfmetalang">\hyxmp@xmlified</rdf:li>^^J%
599         }%
600       \fi
601     \fi
602 }
```

```

600     \fi
601     \hyxmp@add@to@xml{%
602 -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@xmlified</rdf:li>^^J%
603 -----</rdf:Alt>^^J%
604 -----</dc:#1>^^J%
605     }%
606 \fi%
607 }%

```

`\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.

```

608 \newcommand*{\hyxmp@list@to@xml}[3]{%
609   \ifx#3\@empty
610   \else
611     \hyxmp@add@to@xml{%
612 -----<dc:#1>^^J%
613 -----<rdf:#2>^^J%
614     }%
615     \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`.

```

616     \hyxmp@xmlify{#3}%
617     \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
618     \def\@elt##1{%
619       \hyxmp@add@to@xml{%
620 -----<rdf:li>##1</rdf:li>^^J%
621       }%
622     }%
623     \hyxmp@list
624     \egroup
625     \hyxmp@add@to@xml{%
626 -----</rdf:#2>^^J%
627 -----</dc:#1>^^J%
628     }%
629 \fi
630 }

```

`\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.

```

631 \newcommand*{\hyxmp@dc@schema}{%
632   \hyxmp@add@to@xml{%
633     -----<rdf:Description rdf:about=""^^J%
634     -----xmlns:dc="http://purl.org/dc/elements/1.1/">^^J%
635     -----<dc:format>application/pdf</dc:format>^^J%
636   }%
637   \hyxmp@rdf@dc{title}{\@pdftitle}%
638   \hyxmp@rdf@dc{description}{\@pdfsubject}%
639   \hyxmp@rdf@dc{rights}{\@pdfcopyright}%
640   \hyxmp@list@to@xml{creator}{Seq}{\hyxmp@pdfauthor}%
641   \hyxmp@list@to@xml{subject}{Bag}{\hyxmp@pdfkeywords}%
642   \hyxmp@list@to@xml{date}{Seq}{\hyxmp@today}%
643   \hyxmp@list@to@xml{language}{Bag}{\@pdflang}%
644   \hyxmp@add@simple{dc:source}{\jobname.tex}%
645   \hyxmp@add@to@xml{%
646     -----</rdf:Description>^^J%
647   }%
648 }

```

### 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.

```

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

```

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

```

650   \let\hyxmp@rights=\@empty
651   \ifx\@pdflicenseurl\@empty
652   \else
653     \def\hyxmp@rights{YES}%
654   \fi
655   \ifx\@pdfcopyright\@empty
656   \else
657     \def\hyxmp@rights{YES}%
658   \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.

```

659   \ifx\hyxmp@rights\@empty
660   \else

```

Header

```

661     \hyxmp@add@to@xml{%
662       -----<rdf:Description rdf:about=""^^J%
663       -----xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/">^^J%
664     }%

```



Copyright indication

```

665 \ifx\@pdfcopyright\@empty
666 \else
667 \hyxmp@add@to@xml{%
668 -----<xmpRights:Marked>True</xmpRights:Marked>^^J%
669 }%
670 \fi

```

License URL

```

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

```

Trailer

```

672 \hyxmp@add@to@xml{%
673 -----</rdf:Description>^^J%
674 }%
675 \fi
676 }

```

### 3.5.5 The XMP Media Management schema

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

```

677 \gdef\hyxmp@mm@schema{%
678 \hyxmp@def@DocumentID
679 \hyxmp@def@InstanceID
680 \hyxmp@add@to@xml{%
681 -----<rdf:Description rdf:about=""^^J%
682 -----_xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"^^J%
683 -----<xmpMM:DocumentID>\hyxmp@DocumentID</xmpMM:DocumentID>^^J%
684 -----<xmpMM:InstanceID>\hyxmp@InstanceID</xmpMM:InstanceID>^^J%
685 -----</rdf:Description>^^J%
686 }%
687 }

```

### 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`.

```

688 \newcommand*{\hyxmp@xmp@basic@schema}{%
689 \hyxmp@add@to@xml{%
690 -----<rdf:Description rdf:about=""^^J%
691 -----_xmlns:xmp="http://ns.adobe.com/xap/1.0/"^^J%
692 }%
693 \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@today}%
694 \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@today}%

```

```

695 \hyxmp@add@simple{xmp:MetadataDate}{\hyxmp@today}%
696 \hyxmp@add@simple{xmp:CreatorTool}{\pdfcreator}%
697 \hyxmp@add@simple{xmp:BaseURL}{\@baseurl}%
698 \hyxmp@add@to@xml{%
699 -----</rdf:Description>^^J%
700 }%
701 }

```

### 3.5.7 The Photoshop schema

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

```

702 \gdef\hyxmp@photoshop@schema{%
703 \edef\hyxmp@photoshop@data{\@pdfauthor\title\pdfcaptionwriter}%
704 \ifx\hyxmp@photoshop@data\empty
705 \else
706 \hyxmp@add@to@xml{%
707 -----<rdf:Description rdf:about=""^^J%
708 -----_xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/">^^J%
709 }%
710 \fi
711 \hyxmp@add@simple{photoshop:AuthorsPosition}{\pdfauthor\title}%
712 \hyxmp@add@simple{photoshop:CaptionWriter}{\pdfcaptionwriter}%
713 \ifx\hyxmp@photoshop@data\empty
714 \else
715 \hyxmp@add@to@xml{%
716 -----</rdf:Description>^^J%
717 }%
718 \fi
719 }

```

### 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.

```

720 \begingroup
721 \catcode'\&=12
722 \catcode'\#=12
723 \gdef\xmplinesep{&#xA;}
724 \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.

```

725 \newcommand*{\hyxmp@list@to@lines}[2]{%
726 \ifx#2\empty
727 \else
728 \bgroup

```

```

729      \hyxmp@add@to@xml{%
730  -----<#1>%
731      }%

```

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

```

732      \def\@elt@first##1{%
733          \hyxmp@add@to@xml{##1}%
734          \let\@elt=\@elt@rest
735      }%

```

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

```

736      \def\@elt@rest##1{%
737          \hyxmp@add@to@xml{\xmplinesep##1}%
738      }%

```

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

```

739      \let\@elt=\@elt@first
740      \hyxmp@xmlify{#2}%
741      \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
742      \hyxmp@list
743      \hyxmp@add@to@xml{</#1>^~J}%
744      \egroup
745      \fi
746  }

```

`\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 `lptc4xmpCore:CreatorContactInfo/CiAdrExtadr`, `lptc4xmpCore:CreatorContactInfo/CiAdrCity`, `lptc4xmpCore:CreatorContactInfo/CiAdrRegion`, `lptc4xmpCore:CreatorContactInfo/CiAdrPcode`, `lptc4xmpCore:CreatorContactInfo/CiAdrCtry`, `lptc4xmpCore:CreatorContactInfo/CiTelWork`, `lptc4xmpCore:CreatorContactInfo/CiEmailWork`, and `lptc4xmpCore:CreatorContactInfo/CiUrlWork` properties.

```

747 \gdef\hyxmp@photometa@schema{%
748   \edef\hyxmp@photometa@data{%
749     \@pdfcontactaddress
750     \@pdfcontactcity
751     \@pdfcontactregion
752     \@pdfcontactpostcode
753     \@pdfcontactcountry
754     \@pdfcontactphone
755     \@pdfcontactemail
756     \@pdfcontacturl
757   }%
758   \ifx\hyxmp@photometa@data\@empty
759     \else

```

```

760 \hyxmp@iptc@extensions
761 \hyxmp@add@to+xml{%
762 -----<rdf:Description rdf:about=""^^J%
763 -----xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"^^J%
764 -----xmlns:IptcContInfo="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/contactinfo/">^^J%
765 -----<Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">^^J%
766 }%
767 \fi
768 \hyxmp@list@to@lines{IptcContInfo:CiAdrExtadr}{\@pdfcontactaddress}%
769 \hyxmp@add@simple{IptcContInfo:CiAdrCity}{\@pdfcontactcity}%
770 \hyxmp@add@simple{IptcContInfo:CiAdrRegion}{\@pdfcontactregion}%
771 \hyxmp@add@simple{IptcContInfo:CiAdrPcode}{\@pdfcontactpostcode}%
772 \hyxmp@add@simple{IptcContInfo:CiAdrCtry}{\@pdfcontactcountry}%

```

\xmplinesep The IPTC standard states that sets of telephone numbers, email address, 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.

```

773 \bgroup
774 \def\xmplinesep{,%
775 \hyxmp@list@to@lines{IptcContInfo:CiTelWork}{\@pdfcontactphone}%
776 \hyxmp@list@to@lines{IptcContInfo:CiEmailWork}{\@pdfcontactemail}%
777 \hyxmp@list@to@lines{IptcContInfo:CiUrlWork}{\@pdfcontacturl}%
778 \egroup
779 \ifx\hyxmp@photometa@data\@empty
780 \else
781 \hyxmp@add@to+xml{%
782 -----</Iptc4xmpCore:CreatorContactInfo>^^J%
783 -----</rdf:Description>^^J%
784 }%
785 \fi
786 }

```

\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.

```

787 \newcommand*{\hyxmp@iptc@extensions}{%
788 \hyxmp@add@to+xml{%
789 -----<rdf:Description rdf:about=""^^J%
790 -----xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"^^J%
791 -----xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema\hyxmp@hash"^^J%
792 -----xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property\hyxmp@hash"^^J%

```

```

793 _____xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type\hyxmp@hash"^^J%
794 _____xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field\hyxmp@hash"^^J%
795 _____<pdfaExtension:schemas>^^J%
796 _____<rdf:Bag>^^J%
797 _____<rdf:li rdf:parseType="Resource">^^J%
798 _____<pdfaSchema:schema>IPTC Core Schema</pdfaSchema:schema>^^J%
799 _____<pdfaSchema:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaSchema:
800 _____<pdfaSchema:prefix>Iptc4xmpCore</pdfaSchema:prefix>^^J%
801 _____<pdfaSchema:property>^^J%
802 _____<rdf:Seq>^^J%
803 _____<rdf:li rdf:parseType="Resource">^^J%
804 _____<pdfaProperty:name>CreatorContactInfo</pdfaProperty:name>^^J%
805 _____<pdfaProperty:valueType>contactinfo</pdfaProperty:valueType>^^J%
806 _____<pdfaProperty:category>external</pdfaProperty:category>^^J%
807 _____<pdfaProperty:description>contact information for the document's creator</p
808 _____</rdf:li>^^J%
809 _____</rdf:Seq>^^J%
810 _____</pdfaSchema:property>^^J%
811 _____<pdfaSchema:valueType>^^J%
812 _____<rdf:Seq>^^J%
813 _____<rdf:li rdf:parseType="Resource">^^J%
814 _____<pdfaType:type>contactinfo</pdfaType:type>^^J%
815 _____<pdfaType:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/contactin
816 _____<pdfaType:prefix>IptcContInfo</pdfaType:prefix>^^J%
817 _____<pdfaType:description>contact information</pdfaType:description>^^J%
818 _____<pdfaType:field>^^J%
819 _____<rdf:Seq>^^J%
820 }%

821 \hyxmp@text@resource{CiAdrExtadr}{contact address}%
822 \hyxmp@text@resource{CiAdrCity}{contact city}%
823 \hyxmp@text@resource{CiAdrRegion}{contact region}%
824 \hyxmp@text@resource{CiAdrPcode}{contact postal code}%
825 \hyxmp@text@resource{CiAdrCtry}{contact country}%
826 \hyxmp@text@resource{CiTelWork}{contact telephone number}%
827 \hyxmp@text@resource{CiEmailWork}{contact email address}%
828 \hyxmp@text@resource{CiUrlWork}{contact url}%

829 \hyxmp@add@to+xml{%
830 _____</rdf:Seq>^^J%
831 _____</pdfaType:field>^^J%
832 _____</rdf:li>^^J%
833 _____</rdf:Seq>^^J%
834 _____</pdfaSchema:valueType>^^J%
835 _____</rdf:li>^^J%
836 _____</rdf:Bag>^^J%
837 _____</pdfaExtension:schemas>^^J%
838 _____</rdf:Description>^^J%
839 }%
840 }

```

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

```

841 \newcommand*{\hyxmp@text@resource}[2]{%
842   \hyxmp@add@to@xml{%
843     -----<rdf:li rdf:parseType="Resource">^^J%
844     -----<pdfaField:name>#1</pdfaField:name>^^J%
845     -----<pdfaField:valueType>Text</pdfaField:valueType>^^J%
846     -----<pdfaField:description>#2</pdfaField:description>^^J%
847     -----</rdf:li>^^J%
848   }
849 }
```

### 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.

```

850 \newcommand*{\hyxmp@pdfa@id@schema}{%
851   \ifHy@pdfa
852     \hyxmp@add@to@xml{%
853       -----<rdf:Description rdf:about=""^^J%
854       -----_xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/">^^J%
855     }%
856     \hyxmp@add@simple{pdfaid:part}{1}%
857     \hyxmp@add@simple{pdfaid:conformance}{B}%
858     \hyxmp@add@to@xml{%
859       -----</rdf:Description>^^J%
860     }%
861   \fi
862 }
```

### 3.5.10 Combining schemata into an XMP packet

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

```

863 \begingroup
864   \ifhyxmp@unicodetex
865     \lccode'\!="FEFF %
866     \lowercase{%
867       \gdef\hyxmp@bom{!}
868     }%
869   \else
870     \catcode'\^^ef=12
871     \catcode'\^^bb=12
872     \catcode'\^^bf=12
873     \gdef\hyxmp@bom{^^ef^^bb^^bf}%
874   \fi
875 \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.

```

876 \def\hyxmp@construct@packet{%
877   \gdef\hyxmp@xml{%
878     \hyxmp@add@to@xml{<?xpacket begin="\hyxmp@bom" %
879 id="W5M0MpCehiHzreSzNTczkc9d"?>^^J%
880 <x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="3.1-702">^^J%
881 ___<rdf:RDF
882 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\hyxmp@hash">^^J%
883   }%
884   \hyxmp@pdf@schema
885   \hyxmp@xmpRights@schema
886   \hyxmp@dc@schema
887   \hyxmp@photoshop@schema
888   \hyxmp@photometa@schema
889   \hyxmp@xmp@basic@schema
890   \hyxmp@pdfa@id@schema
891   \hyxmp@mm@schema
892   \hyxmp@add@to@xml{%
893 ___</rdf:RDF>^^J%
894 </x:xmpmeta>^^J%
895 \hyxmp@padding
896 <?xpacket end="w"?>^^J%
897   }%
898 }

```

### 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`

```

899 \newcommand*{\hyxmp@embed@packet}{%
900   \hyxmp@construct@packet
901   \def\hyxmp@driver{hpdftex}%
902   \ifx\hyxmp@driver\Hy@driver
903     \hyxmp@embed@packet@pdftex
904   \else
905     \def\hyxmp@driver{hdvipdfm}%
906     \ifx\hyxmp@driver\Hy@driver
907       \hyxmp@embed@packet@dvipdfm
908     \else
909       \def\hyxmp@driver{hxdetex}%
910       \ifx\hyxmp@driver\Hy@driver
911         \hyxmp@embed@packet@xdetex
912       \else
913         \@ifundefined{pdfmark}{%
914           \PackageWarningNoLine{hyperxmp}{%
915             Unrecognized hyperref driver ‘\Hy@driver’. \MessageBreak

```

```

916         \jobname.tex's XMP metadata will *not* be\MessageBreak
917         embedded in the resulting file}%
918     }{%
919         \hyxmp@embed@packet@pdfmark
920     }%
921     \fi
922 \fi
923 \fi
924 }

```

### 3.6.1 Embedding using pdfTeX

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

```

925 \newcommand*{\hyxmp@embed@packet@pdfTeX}{%
926     \bgroup
927     \pdfcompresslevel=0
928     \immediate\pdfobj stream attr {%
929         /Type /Metadata
930         /Subtype /XML
931     }{\hyxmp@xml}%
932     \pdfcatalog {/Metadata \the\pdflastobj\space 0 R}%
933     \egroup
934 }

```

### 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.

```

935 \newcommand*{\hyxmp@embed@packet@pdfmark}{%
936     \pdfmark{%
937         pdfmark=/NamespacePush
938     }%
939     \pdfmark{%
940         pdfmark=/OBJ,
941         Raw={/_objdef \string\hyxmp@Metadata\string} /type /stream}%
942     }%
943     \pdfmark{%
944         pdfmark=/PUT,
945         Raw={\string\hyxmp@Metadata\string}
946         2 dict begin
947             /Type /Metadata def
948             /Subtype /XML def
949             currentdict
950         end
951     }%
952 }%
953 \pdfmark{%
954     pdfmark=/PUT,

```



```

955     Raw={\string{hyxmp@Metadata\string} (\hyxmp+xml)}%
956   }%
957   \pdfmark{%
958     pdfmark=/Metadata,
959     Raw={\string{Catalog\string} \string{hyxmp@Metadata\string}}%
960   }%
961   \pdfmark{%
962     pdfmark=/NamespacePop
963   }%
964 }

```

### 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.

```

965 \newcommand*{\hyxmp@embed@packet@dvipdfm}{%
966   \hyxmp@string@len{\hyxmp+xml}%
967   \special{pdf: object @hyxmp@Metadata
968     <<
969       /Type /Metadata
970       /Subtype /XML
971       /Length \the\@tempcnta
972     >>
973     stream^^J\hyxmp+xml endstream%
974   }%
975   \special{pdf: docview
976     <<
977       /Metadata @hyxmp@Metadata
978     >>
979   }%
980 }

```

`\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.

```

981 \newcommand*{\hyxmp@string@len}[1]{%
982   \@tempcnta=0
983   \expandafter\hyxmp@count@spaces#1 {} %
984   \expandafter\hyxmp@count@non@spaces#1{}%
985 }

```

`\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.

```

986 \def\hyxmp@count@spaces#1 {%
987   \def\hyxmp@one@token{#1}%

```

```

988 \ifx\hyxmp@one@token\@empty
989 \advance\@tempcnta by -1
990 \else
991 \advance\@tempcnta by 1
992 \expandafter\hyxmp@count@spaces
993 \fi
994 }

```

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

```

995 \newcommand*\hyxmp@count@non@spaces}[1]{%
996 \def\hyxmp@one@token{#1}%
997 \ifx\hyxmp@one@token\@empty
998 \else
999 \advance\@tempcnta by 1
1000 \expandafter\hyxmp@count@non@spaces
1001 \fi
1002 }

```

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

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

```

1003 \newcommand*\hyxmp@embed@packet@xetex{%
1004 \special{pdf:stream @hyxmp@Metadata (\hyxmp@xml)
1005 <<
1006 /Type /Metadata
1007 /Subtype /XML
1008 >>
1009 }%
1010 \special{pdf:put @catalog
1011 <<
1012 /Metadata @hyxmp@Metadata
1013 >>
1014 }%
1015 }

```

## 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.

```

1016 \catcode'\@=" \hyxmp@dq@code

```

## 4 Future Work

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

## 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>T<sub>E</sub>X 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>
    </rdf:Description>
  </rdf:RDF>
</x:xmpmeta>
</xpacket>
```

```

        </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>2014-01-02T22:51:00-07:00</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: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
    </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:li rdf:parseType="Resource">
        <pdfaField:name>CiTelWork</pdfaField:name>
        <pdfaField:valueType>Text</pdfaField:valueType>
        <pdfaField:description>contact telephone number</pdfaField:description>
    </rdf:li>
    <rdf:li rdf:parseType="Resource">
        <pdfaField:name>CiEmailWork</pdfaField:name>
        <pdfaField:valueType>Text</pdfaField:valueType>
        <pdfaField:description>contact email address</pdfaField:description>
    </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/contactinfo">
    <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>
    </Iptc4xmpCore:CreatorContactInfo>

```

```

<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>2014-01-02T22:51:00-07:00</xmp:CreateDate>
  <xmp:ModifyDate>2014-01-02T22:51:00-07:00</xmp:ModifyDate>
  <xmp:MetadataDate>2014-01-02T22:51:00-07:00</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://www.images.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		Added support for the Photoshop schema . . . . .	1
General: Initial version . . . . .	1		
v1.1		Made the package compatible with <code>ngerman</code> . Thanks to Tobias Mueller for the bug report. . . . .	9
<code>\hyxmp@construct@packet</code> : Explicitly set the category codes of characters $\langle EF \rangle$ , $\langle BB \rangle$ , and $\langle BF \rangle$ to “letter”. Thanks to Daniel Schömer for the bug report . . . . .	38		
v1.2			
General: Added support for the X <sub>Y</sub> TeX backend ( <code>xdvipdfmx</code> ) . . .	1		
		v1.3	
		General: Introduced the <code>pdfmetalang</code> package option, which enables an author to specify the language in which he wrote the document’s metadata	15
		<code>\hyxmp@reencode</code> : Introduced this	



macro to re-encode Unicode strings as 8-bit strings before manipulating them into XMP schema. This change addresses a bug reported by Martin Münch	18	\hyxmp@list@to@xml: Modified by Heiko Oberdiek to use the new Unicode-processing macros	31
v1.4		\hyxmp@photoshop@schema: Simplified using \hyxmp@add@simple	34
\hyxmp@mm@schema: Renamed the xapMM namespace prefix to xmpMM	33	\hyxmp@ProcessKeyvalOptions: Added this macro	13
\hyxmp@rdf@dc: Included metadata in the x-default language regardless of the specified metadata language	30	\hyxmp@reencode: Replaced with an empty macro by Heiko Oberdiek	18
\hyxmp@xmpRights@schema: Renamed the xapRights namespace prefix to xmpRights	32	\hyxmp@skiptorelax: Added by Heiko Oberdiek	23
v1.5		\hyxmp@skipzeros: Added by Heiko Oberdiek	21
General: Made the XMP inclusion more robust. Thanks to Heiko Oberdiek for the bug report and suggested modifications.	10	\hyxmp@SpaceOther: Added by Heiko Oberdiek	22
v2.0		\hyxmp@string: Added this macro	30
General: Added support for the XMP Basic schema and miscellaneous other bits of metadata	1	\hyxmp@toxml: Added by Heiko Oberdiek	20
Heiko Oberdiek's major rewrite of the code to better support native-Unicode T <sub>E</sub> X implementations (X <sub>Ǝ</sub> T <sub>E</sub> X and LuaT <sub>E</sub> X)	1	Escaped parentheses written with pdfmarks to prevent dvips from line-wrapping the XMP packet	21
New \AtBeginDocument code from Heiko Oberdiek to properly encode \@pdfmetalang	15	\hyxmp@toxml@unicodetex: Added by Heiko Oberdiek	21
\hyxmp@add@to@xml: Updated also to replace commas	27	\hyxmp@xetex@crap: Added by Heiko Oberdiek	22
\hyxmp@bom: Added by Heiko Oberdiek	38	\hyxmp@xmlify: Completely rewritten by Heiko Oberdiek to better support Unicode-enabled T <sub>E</sub> X programs	19
\hyxmp@comma: Added this macro	16	\hyxmp@xmp@basic@schema: Added this macro	33
\hyxmp@construct@packet: Modified by Heiko Oberdiek to use an appropriate BOM representation via \hyxmp@bom	38	\hyxmp@xmpRights@schema: Modified to include xmpRights:Marked only when pdfcopyright is specified and xmpRights:WebStatement only when pdflicenseurl is specified	32
\hyxmp@crap@convert: Added by Heiko Oberdiek	23	\hyxmp@zero: Added by Heiko Oberdiek	23
\hyxmp@crap@test: Added by Heiko Oberdiek	22	\ifhyxmp@unicodetex: Added by Heiko Oberdiek	18
\hyxmp@dc@schema: Added support for dc:language and dc:source	31	\ProcessKeyvalOptions: Added this macro	13
\hyxmp@is@unicode: Added by Heiko Oberdiek	20	\xmpcomma: Added this macro	16
		\xmpquote: Added this macro	17
		\XMPTruncateList: Added this macro	17

v2.1	General: Enabled <code>hyperxmp</code> and <code>hyperref</code> to be loaded in either order. This addresses a bug report by Yury Donskoy . . . . .	12	v2.4	General: Added support for the PDF/A Identification schema, as requested by Florian Breitwieser . . . . .	1
	<code>\hypersetup</code> : Added this macro . . . . .	14		<code>\hyxmp@add@simple@var</code> : Added this macro . . . . .	30
	<code>\hyxmp@hypersetup</code> : Added this macro . . . . .	14		<code>\hyxmp@create@uuid</code> : Modified this macro to produce a proper version 4 (random or pseudorandom) UUID . . . . .	25
	<code>\hyxmp@redefine@Hyp</code> : Added this macro . . . . .	12		<code>\hyxmp@dc@schema</code> : Made <code>dc:language</code> a <code>Bag</code> instead of an individual item so as to conform to the latest XMP specifications, a detail identified by Florian Breitwieser . . . . .	31
v2.2	General: Added support for the IPTC Photo Metadata schema . . . . .	1		<code>\hyxmp@parse@time</code> : Added this macro . . . . .	28
	<code>\hyxmp@iptc@extensions</code> : Added this macro to support PDF/A generation . . . . .	36		<code>\hyxmp@parse@tz</code> : Added this macro . . . . .	28
	<code>\hyxmp@list@to@lines</code> : Added this macro . . . . .	34		<code>\hyxmp@parse@tz@char</code> : Added this macro . . . . .	28
	<code>\hyxmp@photometa@schema</code> : Added this macro . . . . .	35		<code>\hyxmp@pdf@schema</code> : Made <code>\hyxmp@pdf@schema</code> <i>always</i> include the Adobe PDF schema, even when empty. Florian Breitwieser noted that this is necessary for PDF/A-1b compliance . . . . .	29
	<code>\hyxmp@text@resource</code> : Added this macro . . . . .	37		<code>\hyxmp@pdf@to@xmp@date</code> : Added this macro . . . . .	28
	<code>\xmpcomma</code> : Changed the default from <code>\relax</code> to an ordinary comma . . . . .	16		<code>\hyxmp@pdfa@id@schema</code> : Added this macro . . . . .	38
	<code>\xmplinesep</code> : Added this macro . . . . .	34		<code>\hyxmp@today</code> : Modified the code to parse the time and timezone from <code>\pdfcreationdate</code> , as proposed by Florian Breitwieser . . . . .	29
v2.3	<code>\hyxmp@iptc@extensions</code> : Gave the <code>lptc4xmpCore:CreatorContactInfo</code> fields a unique <code>pdfaType:prefix</code> to better support conversion of the document to PDF/A . . . . .	36		<code>\hyxmp@today@define</code> : Added this macro . . . . .	28
v2.3a	General: Bug fix: Redefine <code>\@pdflang</code> as <code>\@empty</code> when <code>hyperref</code> has set it to <code>\relax</code> . . . . .	15		<code>\xmptilde</code> : Added this macro . . . . .	17
v2.3b	<code>\XMPTruncateList</code> : Made all definitions local to avoid spurious Too many unprocessed floats errors when running with memoir . . . . .	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	\@pdftitle . . . . .	\define@key . . . . .
\# . . . . . 499, 722	. . . . . 114, 474, 480, 637	. . . . . 16, 18, 20, 22,
\& . . . . . 256, 288, 721	\@secondoftwo . . . . . 249	24, 26, 34, 36, 38,
\@baseurl . . . . . 97, 697	\^ . . . . . 165, 193, 494, 870–872	40, 42, 44, 46, 56, 73
\@elt . . . . . 173, 616, 734, 739	\_ . . . . . 493	dvipdf (option) . . . . . 40
\@elt@first . . . . . 732	\~ . . . . . 170, 326, 327	dvipdfm . . . . . 41
\@elt@rest . . . . . 734, 736		dvips (option) . . . . . 40
\@firstofone . . . . . 335		dvips . . . . . 6, 20, 21
\@firstoftwo . . . . . 247		dvipsone (option) . . . . . 40
\@gobble . . . . . 178, 332	\_ . . . . . 292, 327, 493	dviwindo (option) . . . . . 40
\@pdfauthor . . . . .	A	E
. . . . . 56, 98, 474, 480	ASCII . . . . . 19	\EdefEscapeHex . . . . . 215, 228
\@pdfauthortitle . . .	\AtBeginDocument . . . 124	\EdefUnescapeHex . . . 232
. . . . . 19, 99, 703, 711	\AtEndDocument . . . . . 5	\EdefUnescapeString . . . 202
\@pdfcaptionwriter . .	\AtEndDvi . . . . . 8	ETX . . . . . 16, 18
. . . . . 21, 100, 703, 712	atenddvi . . . . . 10	G
\@pdfcontactaddress . .	Author . . . . . 6, 17	Ghostscript . . . . . 6
. . . . . 25, 101, 749, 768	B	H
\@pdfcontactcity . . .	baseurl (option) 3, 9, 10, 33	\Hy@driver . . . . . 4,
. . . . . 33, 102, 750, 769	BOM . . . . . 38	902, 906, 910, 915
\@pdfcontactcountry . .	C	hyperref . . . . . 1, 3–6, 10,
. . . . . 39, 103, 753, 772	CiAdrCity . . . . . 2, 35	12, 14, 15, 29, 39, 40
\@pdfcontactemail . . .	CiAdrCtry . . . . . 2, 35	\hypersetup . . . . . 90
. . . . . 43, 104, 755, 776	CiAdrExtadr . . . . . 2, 35	hyperxmp . . . . . 1–6, 8–12,
\@pdfcontactphone . . .	CiAdrPcode . . . . . 2, 35	14–16, 18, 24, 29, 43
. . . . . 41, 105, 754, 775	CiAdrRegion . . . . . 2, 35	\hyxmp@@isunicode . . 236
\@pdfcontactpostcode . .	CiEmailWork . . . . . 2, 35	\hyxmp@cadd@simple . .
. . . . . 37, 106, 752, 771	CiTelWork . . . . . 2, 35	. . . . . 562, 568,
\@pdfcontactregion . . .	CiUrlWork . . . . . 2, 35	644, 671, 693–
. . . . . 35, 107, 751, 770	D	697, 711, 712,
\@pdfcontacturl . . . .	Date . . . . . 29	769–772, 856, 857
. . . . . 45, 108, 756, 777	\day . . . . . 542, 543, 545	\hyxmp@cadd@simple@var
\@pdfcopyright . . . 15,	dc:creator . . . . . 2, 6, 31	. . . . . 559, 560, 578
109, 639, 655, 665	dc:date . . . . . 2, 31	\hyxmp@cadd@to+xml . .
\@pdfcreator . . . . . 696	dc:description . . . . . 3, 31	. . . . . 485,
\@pdfkeywords . . . 73, 110	dc:format . . . . . 2	504, 555, 564,
\@pdflang . . . . . 111, 126,	dc:language . . . . . 2, 31	573, 582, 591,
127, 129, 132, 643	dc:rights . . . . . 2, 31	597, 601, 611,
\@pdflicenseurl . . . .	dc:source . . . . . 2, 31	619, 625, 632,
. . . . . 17, 112, 651, 671	dc:subject . . . . . 2, 31	645, 661, 667,
\@pdfmetalang 23, 130,	dc:title . . . . . 3, 31	672, 680, 689,
132, 134, 595, 598		
\@pdfsubject . . . . . 113, 638		

698, 706, 715,	\hyxmp@embed@packet@dvi	\hyxmp@pdfauthor ...
729, 733, 737,	..... 907, 965	..... 47, 56, 640
743, 761, 781,	\hyxmp@embed@packet@pdfmark	\hyxmp@pdfkeywords .
788, 829, 842,	..... 919, 935	..... 47, 73, 641
852, 858, 878, 892	\hyxmp@embed@packet@pdftex	\hyxmp@photometa@data
\hyxmp@append@hex ..	..... 903, 925	..... 747
. 422, 441–443, 447	\hyxmp@embed@packet@xetex	\hyxmp@photometa@schema
\hyxmp@append@hex@iii	..... 911, 1003	..... 747, 888
. 440, 446, 456, 467	\hyxmp@find@metadata	\hyxmp@photoshop@data
\hyxmp@append@hex@iv	..... 95, 136	..... 702
..... 445, 451,	\hyxmp@hash .....	\hyxmp@photoshop@schema
452, 454, 469–471	. 498, 791–794, 882	..... 702, 887
\hyxmp@at@end ... 3, 135	\hyxmp@Hyp@pdfauthor 50	\hyxmp@ProcessKeyvalOptions
\hyxmp@big@prime ...	\hyxmp@Hyp@pdfkeywords	..... 85
. 396, 399, 409, 419	..... 67	\hyxmp@rand@num 415,
\hyxmp@big@prime@ii	\hyxmp@hypersetup .. 90	424, 459, 476, 482
..... 396, 418	\hyxmp@InstanceID ..	\hyxmp@rdf@dc .....
\hyxmp@bom .... 863, 878	..... 479, 684	.... 587, 637–639
\hyxmp@comma .....	\hyxmp@iptc@extensions	\hyxmp@redefine@Hyp
... 27, 57, 74, 164	..... 760, 787	..... 49, 87, 92
\hyxmp@commas@to@list	\hyxmp@is@unicode ..	\hyxmp@reencode ... 198
. 148, 175, 617, 741	.... 204, 221, 236	\hyxmp@rights .....
\hyxmp@commas@to@list@i	\hyxmp@legal .....	. 650, 653, 657, 659
..... 150, 152	\hyxmp@list .....	\hyxmp@seed@rng ...
\hyxmp@concat@metadata	. 617, 623, 741, 742	.... 398, 475, 481
..... 95	\hyxmp@list@to@lines	\hyxmp@seed@rng@i ..
\hyxmp@construct@packet	. 725, 768, 775–777	..... 400, 402
..... 876, 900	\hyxmp@list@to@xml ..	\hyxmp@seed@string ..
\hyxmp@count@non@spaces	.... 608, 640–643	. 474, 475, 480, 481
..... 984, 995	\hyxmp@mm@schema 677, 891	\hyxmp@set@rand@num
\hyxmp@count@spaces	\hyxmp@modulo@a 390,	.... 415, 423, 458
..... 983, 986	409, 419, 425, 460	\hyxmp@skiptorelax ..
\hyxmp@crap@convert	\hyxmp@num .....	..... 345, 351
..... 318, 352	\hyxmp@one@token ...	\hyxmp@skipzeros ... 303
\hyxmp@crap@result ..	..... 398, 402,	\hyxmp@SpaceOther ..
..... 308, 344	987, 988, 996, 997	..... 312, 325
\hyxmp@crap@test 315, 340	\hyxmp@padding 502, 895	\hyxmp@string .....
\hyxmp@create@uuid ..	\hyxmp@parse@time ..	\hyxmp@string@len ..
.... 449, 477, 483	..... 516, 518	..... 966, 981
\hyxmp@dc@schema 631, 886	\hyxmp@parse@tz ...	\hyxmp@sublist .....
\hyxmp@def@DocumentID	.... 525, 528, 532	. 153, 154, 157, 158
..... 473, 678	\hyxmp@parse@tz@char	\hyxmp@temp@list ... 173
\hyxmp@def@InstanceID	..... 520, 522	\hyxmp@temp@str ... 173
..... 479, 679	\hyxmp@pdf@schema ..	\hyxmp@text .....
\hyxmp@DocumentID ..	..... 554, 884	. 200, 278, 308, 352
..... 473, 683	\hyxmp@pdf@to@xmp@date	\hyxmp@text@resource
\hyxmp@dq@code . 1, 1016	..... 514, 551	.... 821–828, 841
\hyxmp@driver ... 3, 899	\hyxmp@pdfa@id@schema	\hyxmp@today .....
\hyxmp@embed@packet	..... 850, 890	. 480, 536, 538,
..... 137, 899		

540, 543, 545, 548, 642, 693–695	ISO . . . . . 11	pdfcontactcountry (option) . . . . . 4
<code>\hyxmp@today@define</code> . . . . . 535, 549	<b>J</b>	pdfcontactemail (option) . . . . . 4
<code>\hyxmp@toxml</code> . . 230, 253	<code>\jobname</code> . . 118, 141, 474, 480, 644, 916	pdfcontactphone (option) . . . . . 4
<code>\hyxmp@toxml@unicodetex</code> . . . . . 218, 278	<b>K</b>	pdfcontactpostcode (option) . . . . . 4
<code>\hyxmp@trimb</code> . . 186, 189	Keywords . . . . . 6, 29	pdfcontactregion (option) . . . . . 4
<code>\hyxmp@trimc</code> . . 189, 190	<code>\KV@Hyp@pdfauthor</code> . . 56	pdfcontacturl (option) 4, 9
<code>\hyxmp@trimspaces</code> . . . . . . . 157, 182	<code>\KV@Hyp@pdfkeywords</code> 73	pdfcopyright (option) . . . . . . 4, 31, 32
<code>\hyxmp@try</code> . . . . . 308	<code>kvoptions</code> . . . . . 10, 13	<code>\pdfcreationdate</code> . . 551
<code>\hyxmp@unicodetexfalse</code> . . . . . 192	<b>L</b>	PDFDocEncoding . . . . . . . . 12, 18, 19
<code>\hyxmp@unicodetextrue</code> . . . . . 192	LF . . . . . 34	pdfescape . . . . . 10
<code>\hyxmp@x@default</code> . . . . . 130, 553, 595, 602	LuaL <sup>A</sup> T <sub>E</sub> X . . . . . 6, 7	pdfkeywords (option) . . . . 3, 8, 10, 12, 31
<code>\hyxmp@xetex@crap</code> . . . . . . . 209, 308	LuaT <sub>E</sub> X . . . . . 18, 21, 43	pdflang (option) . . . . . . 3, 4, 10, 15, 31
<code>\hyxmp@xml</code> . . . . . 495, 502, 876, 931, 955, 966, 973, 1004	<b>M</b>	<code>\pdflastobj</code> . . . . . 932
<code>\hyxmp@xmlified</code> . . . . . 200, 574, 583, 598, 602, 617, 741	Metadata . . . . . 6, 39, 42	pdfL <sup>A</sup> T <sub>E</sub> X . . . . . 3, 5
<code>\hyxmp@xmlify</code> . . . . . . . 134, 200, 572, 581, 590, 616, 740	<code>\month</code> . . . . . 537, 538, 540	pdflicenseurl (option) . . . . . . 4, 9, 32
<code>\hyxmp@xmp@basic@schema</code> . . . . . 688, 889	<b>N</b>	pdfmark (option) . . . . 40
<code>\hyxmp@xmpRights@schema</code> . . . . . 649, 885	nativepdf (option) . . . 40	<code>\pdfmark</code> . . 936, 939, 943, 953, 957, 961
<code>\hyxmp@zero</code> . . . . 361, 368, 375, 381, 386	<code>\newif</code> . . . . . 192	pdfmetalang (option) . . 4
<b>I</b>	<code>\next</code> . . . . . 152, 402	<code>\pdfminorversion</code> . . 562
IETF . . . . . 4	ngerman . . . . . 9	<code>\pdfobj</code> . . . . . 928
<code>\ifHy@pdfa</code> . . . . . 851	<code>\number</code> 355, 357, 359, 364, 366, 371, 373	pdfproducer (option) 3, 10
<code>\ifhyxmp@unicodetex</code> . . . . 192, 203, 864	<b>P</b>	<code>\pdfstringdef</code> 16, 18, 20, 22, 24, 29, 34, 36, 38, 40, 42, 44, 46
<code>ifxetex</code> . . . . . 10	<code>\PackageWarningNoLine</code> . . . . 117, 140, 914	pdfsubject (option) . . . . . . . 3, 10, 31
<code>\ifxetex</code> . . . . . 208	PDF 1–3, 5–8, 15, 16, 20, 26, 28, 29, 38, 39, 42	pdfT <sub>E</sub> X . . 10, 20, 40, 43
<code>Info</code> . . . . . 6	PDF/A . . 3, 27, 29, 36, 38	pdftitle (option) 4, 10, 31
<code>intcalc</code> . . . . . 10	pdf:Keywords . . . . . 2, 29	photoshop:AuthorsPosition . . . . . 3, 34
<code>\intcalcDiv</code> 357, 364, 371	pdf:PDFVersion . . . . 3, 29	photoshop:CaptionWriter . . . . . 2, 34
<code>\intcalcMod</code> 359, 366, 373	pdf:Producer . . . . . 3, 29	PI . . . . . 26
<code>IPTC</code> . . . . 6, 11, 27, 35, 36	pdfaid:conformance . . . 3	<code>\ProcessKeyvalOptions</code> . . . . . 85
<code>lptc4xmpCore:CreatorContactInfo</code> . . . . . 2, 35	pdfaid:part . . . . . 3	Producer . . . . . 29
	pdfauthor (option) . . . . . . 3, 8, 10, 12, 31	ps2pdf (option) . . . . 40
	pdfauthortitle (option) 4, 9	
	pdfcaptionwriter (option) . . . . . 4	
	<code>\pdfcatalog</code> . . . . . 932	
	<code>\pdfcompresslevel</code> . . 927	
	pdfcontactaddress (option) . . . . 4, 6, 7	
	pdfcontactcity (option) . 4	

<b>Q</b>		
\Q	182, 191	
<b>R</b>		
rdf:li	2	
rdf:Seq	2	
\renewcommand	86	
\RequirePackage	7, 10–14	
<b>S</b>		
\SE->pdfdoc@03	199	
\special	967, 975, 1004, 1010	
stringenc	10	
\StringEncodingConvert	205, 211, 222, 225, 320	
Subject	6	
<b>T</b>		
T <sub>E</sub> X	18, 20, 21, 24, 33, 41–43	
Text	38	
textures (option)	40	
Title	6	
<b>U</b>		
Unicode	10, 18–22, 31, 35, 38, 43	
URL	2, 4, 9, 10, 12, 32, 33, 36	
UTF-16BE	20	
UTF-32BE	19	
UTF-8	19, 20	
UUID	24–26	
<b>V</b>		
\vfuzz	190	
vtexpdfmark (option)	40	
<b>X</b>		
\x	308	
xdvipdfmx	7, 42	
X <sub>Y</sub> L <sub>A</sub> T <sub>E</sub> X	6, 7	
X <sub>Y</sub> L <sub>A</sub> T <sub>E</sub> X	10, 18, 21, 22, 42, 43	
XML	1, 2, 6, 16, 18–21, 26, 27, 29–31, 34, 36, 38	
XMP	1–3, 6–10, 15–17, 20, 21, 24, 26–30, 32, 33, 36, 40–43	
xmp:BaseURL	2	
xmp:CreateDate	2	
xmp:CreatorTool	3	
xmp:MetadataDate	2	
xmp:ModifyDate	2	
\xmpcomma	27, 30, 56, 73, 163	
xmpincl	3	
\xmplinesep	720, 737, 773	
xmpMM:DocumentID	2, 24, 33	
xmpMM:InstanceID	2, 24, 33	
\xmpquote	28, 31, 56, 73, 168	
xmpRights:Marked	2, 32	
xmpRights:WebStatement	2, 32	
\xmptilde	169	
\XMPTruncateList	173	
<b>Y</b>		
\year	536	