

The hyperxmp package*

Scott Pakin
scott+hyxmp@pakin.org

January 9, 2013

Abstract

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

1 Introduction

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

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

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

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

*This document corresponds to `hyperxmp` v2.3, dated 2013/01/08.

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

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

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

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

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

- contact address (`lptc4xmpCore:CreatorContactInfo/CiAdrExtadr`,
`lptc4xmpCore:CreatorContactInfo/CiAdrCity`,
`lptc4xmpCore:CreatorContactInfo/CiAdrRegion`,
`lptc4xmpCore:CreatorContactInfo/CiAdrPcode`,
`lptc4xmpCore:CreatorContactInfo/CiAdrCtry`) and
- contact telephone number(s) (`lptc4xmpCore:CreatorContactInfo/CiTelWork`)
- contact email address(es) (`lptc4xmpCore:CreatorContactInfo/CiEmailWork`)
- contact URL(s) (`lptc4xmpCore:CreatorContactInfo/CiUrlWork`)

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

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

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

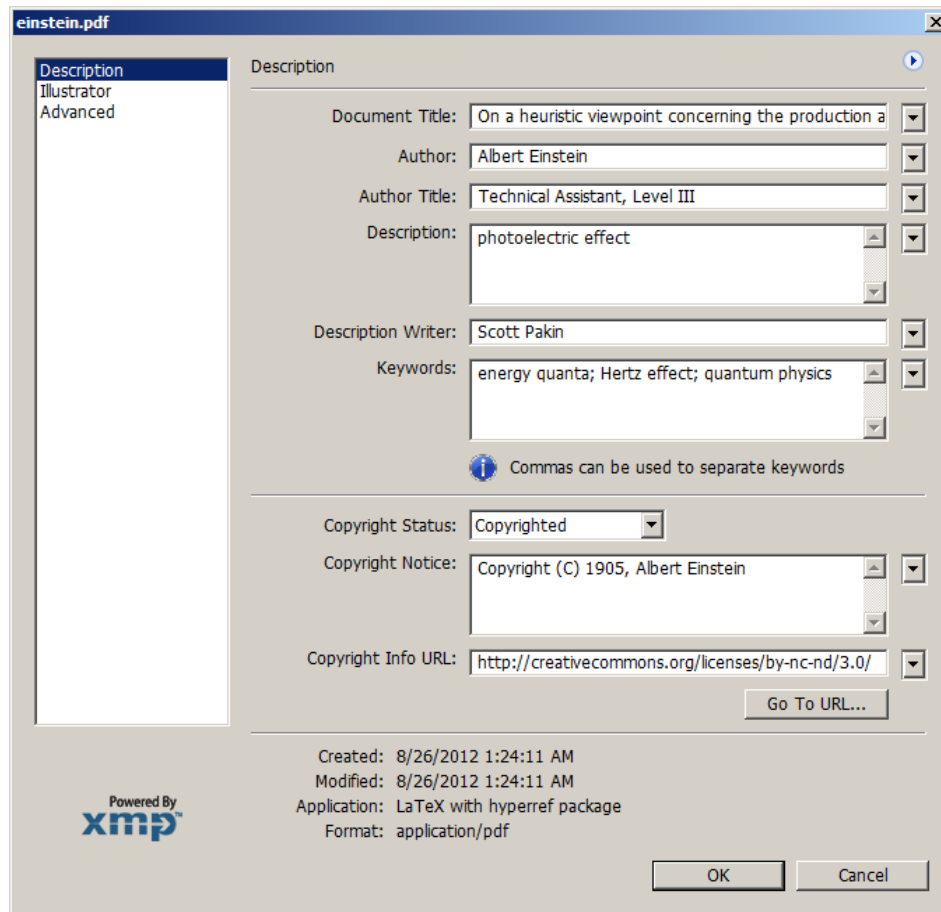


Figure 1: XMP metadata as it appears in Adobe Acrobat

`\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_qL^AT_EX object compression X_qL^AT_EX (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^AT_EX), (2) pass the `--output-driver="xdvipdfmx -z0"` option to X_qL^AT_EX to instruct `xdvipdfmx` to

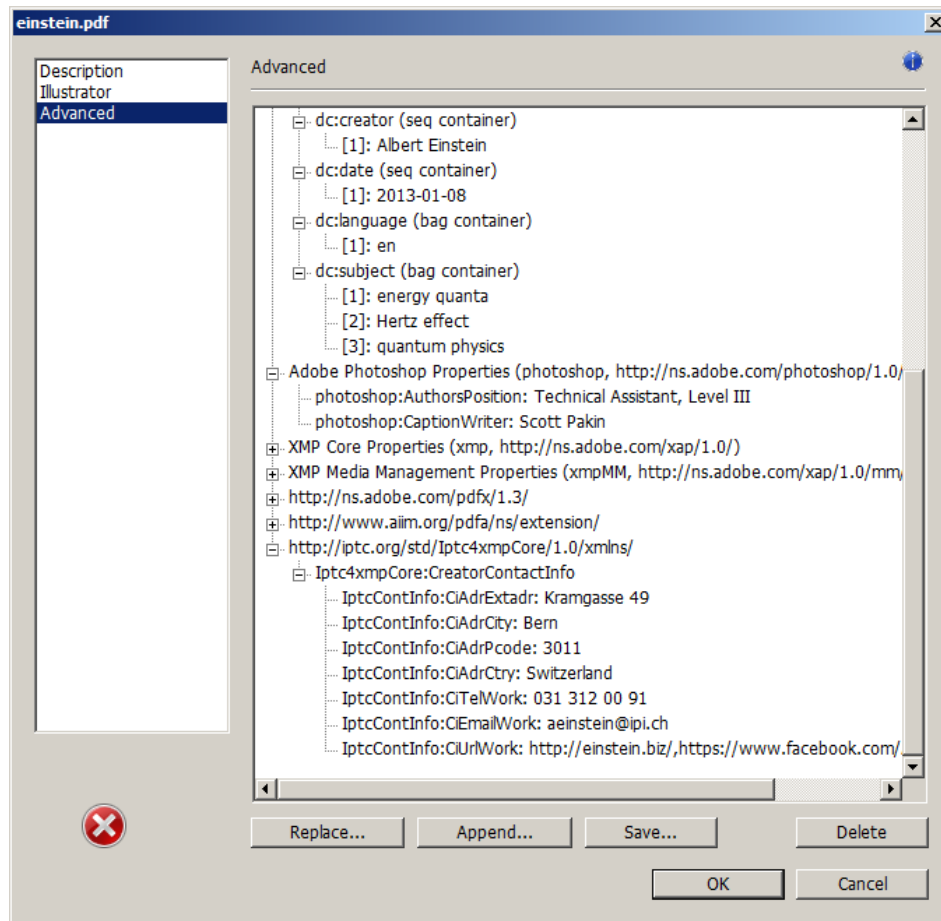


Figure 2: Additional XMP metadata as it appears in Adobe Acrobat

turn off all compression (which will of course make the PDF file substantially larger), or (3) postprocess the generated PDF file by loading it into the commercial version of Adobe Acrobat and re-saving it with the Save As... menu option.

Note 4: Literal commas hyperxmp splits the pdfauthor and pdfkeywords lists at commas. Therefore, when specifying pdfauthor and pdfkeywords, you should separate items with commas. Also, omit “and” and other text that does not belong to any list item. The following examples should serve as clarification:

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

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

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

`\xmpcomma` If you need to include a literal comma within an author or keyword list (where
`\xmpquote` commas normally separate list items) or a street address (where commas normally
separate lines), use the `\xmpcomma` macro to represent it, and wrap the entire entry
containing the comma within `\xmpquote{...}` as shown below:

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

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

As of version 2.2 of `hyperxmp`, it is acceptable to use `\xmpcomma` and `\xmpquote` within any `hyperxmp` option, not just in those in which a comma normally serves as a separator (i.e., lists and multiline fields). Outside of those cases, `\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.

3 Implementation

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

3.1 Initial preparation

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

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

`\hyxmp@at@end` The `\hyxmp@at@end` macro includes code at the end of the document. For pdfT_EX,
`\hyxmp@driver` the standard `\AtEndDocument` works well enough. For all the other backends we
use `\AtEndDvi` from the `atenddvi` package, which is more robust but requires an
addition L^AT_EX 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 pdftitle, pdfauthor, pdfsubject, pdfkeywords, and pdflang options. It also introduces five new options: pdfcopyright, pdflicenseurl, pdfauthortitle, pdfcaptionwriter, and pdfmetalang. For consistency with hyperref's document-metadata naming conventions (which are in turn based on L^AT_EX's document-metadata naming conventions), we do not prefix metadata-related macro names with our package-specific \hyxmp@ prefix. That is, we use names like \@pdfcopyright instead of \hyxmp@pdfcopyright.

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

```

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 }

```

<code>\hyxmp@hypersetup</code> <code>\hypersetup</code>	Redefine <code>hyperref</code> 's <code>\hypersetup</code> command to invoke <code>\hyxmp@redefine@Hyp</code> before performing its normal option processing. <pre> 90 \let\hyxmp@hypersetup=\hypersetup 91 \def\hypersetup{% 92 \hyxmp@redefine@Hyp 93 \hyxmp@hypersetup 94 }</pre>
<code>\hyxmp@find@metadata</code> <code>\hyxmp@concat@metadata</code>	Issue a warning message if the author failed to include any metadata at all. Note that we don't consider <code>\pdfmetlang</code> as metadata as that value is meaningful only when used in conjunction with other information. <pre> 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 hyperxmp\MessageBreak 120 documentation for instructions on how to provide\MessageBreak 121 metadata values to hyperxmp}% 122 \fi 123 }</pre>

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}{%
```

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.

```

126 \ifx\@pdflang\@empty
127 \let\@pdfmetalang=\hyxmp@x@default
128 \else
129 \edef\@pdfmetalang{\@pdflang}%
130 \fi
131 \hyxmp@xmlify\@pdfmetalang

```

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

```

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

```

3.3 Manipulating author-supplied data

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

3.3.1 List manipulation

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

```

\hyxmp@commas@to@list Given a macro name (#1) and a comma-separated list (#2), define the macro name
                        as the elements of the list, each preceded by \@elt. (Executing the macro therefore
                        applies \@elt to each element in turn.)
145 \newcommand*\hyxmp@commas@to@list}[2]{%
146 \gdef#1{%
147 \expandafter\hyxmp@commas@to@list#1\expandafter#1#2,,%
148 }

```

`\hyxmp@commas@to@list@i` Recursively construct macro #1 from comma-separated list #2. Stop if #2 is empty.

```
\next 149 \def\hyxmp@commas@to@list@i#1#2,{%
150   \gdef\hyxmp@sublist{#2}%
151   \ifx\hyxmp@sublist\@empty
152     \let\next=\relax
153   \else
154     \hyxmp@trimspaces\hyxmp@sublist
155     \@cons{#1}{\hyxmp@sublist}%
156     \def\next{\hyxmp@commas@to@list@i{#1}}%
157   \fi
158   \next
159 }
```

`\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.

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

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

```
161 \bgroup
162 \catcode'\^^C=11
163 \gdef\hyxmp@comma{^^C}
164 \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.

```
165 \let\xmpquote=\relax
```

`\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.

```
\hyxmp@temp@str
\hyxmp@temp@list
\@elt
166 \newcommand{\XMPTruncateList}[1]{%
167   \edef\hyxmp@temp@str{\csname hyxmp@#1\endcsname}%
168   \hyxmp@commas@to@list{\hyxmp@temp@list}{\hyxmp@temp@str}%
169   \def\@elt##1{%
170     \expandafter\gdef\csname @#1\endcsname{##1}%
171     \let\@elt=\@gobble
```



```

172 }
173 \hyxmp@temp@list
174 }

```

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.

```

175 \catcode'\Q=3

\hyxmp@trimspaces\x redefines \x to have the same replacement text sans leading
and trailing space tokens.

176 \newcommand{\hyxmp@trimspaces}[1]{%
    Use grouping to emulate a multi-token afterassignment queue.
177 \begingroup
    Put “\toks 0 {” into the afterassignment queue.
178 \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.
179 \expandafter\hyxmp@trimb\expandafter\noexpand#1Q Q}%
    Transfer the trimmed text back into #1.
180 \edef#1{\the\toks0}%
181 }

```

`\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.

```

182 \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.

```

183 \def\hyxmp@trimc#1Q#2{\afterassignment\endgroup \vfuzz\the\vfuzz#1}
184 \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 XqTeX and LuaTeX natively support Unicode. We define the conditional
\hyxmp@unicodetexttrue \ifhyxmp@unicodetex to check for these so we can properly handle encoding
\hyxmp@unicodetextfalse conversions. The trick here is that Unicode TeX implementations compare decimal
64 to hexadecimal 40 (decimal 64), specified with four carets, and take the
TRUE branch; non-Unicode TeX implementations compare decimal 64 to character
“~” (decimal 94), ignore the “~0040” and the rest of the TRUE branch, and
take the FALSE branch.

185 \newif\ifhyxmp@unicodetex
186 \ifnum64='\^^^0040\relax
187   \hyxmp@unicodetexttrue
188 \else
189   \hyxmp@unicodetextfalse
190 \fi

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

191 \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.

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

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

193 \newcommand*{\hyxmp@xmlify}[1]{%
194   \gdef\hyxmp@xmlified{%
Escaped PDF string → PDFDocEncoding/Unicode
195   \EdefUnescapeString\hyxmp@text{#1}%
196   \ifhyxmp@unicodetex
PDFDocEncoding/Unicode → UTF-32BE
197     \hyxmp@is@unicode\hyxmp@text{%
198       \StringEncodingConvert
199       \hyxmp@text\hyxmp@text{utf16be}{utf32be}%
200     }{%
201       \ifxetex
202         \hyxmp@xetex@crap
203       \else
204         \StringEncodingConvert
205         \hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%

```

```

206     \fi
207   }%

  UTF-32BE → UTF-32BE as hex string
208   \EdefEscapeHex\hyxmp@text{\hyxmp@text}%
  UTF-32BE → XML in ASCII
209   \edef\hyxmp@text{%
210     \expandafter
211   }\expandafter\hyxmp@toxml@unicodetex\hyxmp@text
212   \relax\relax\relax\relax\relax\relax\relax\relax
213   \else
  PDFDocEncoding/Unicode → UTF-8
214   \hyxmp@is@unicode\hyxmp@text{%
215     \StringEncodingConvert
216     \hyxmp@text\hyxmp@text{utf16be}{utf8}%
217   }{%
218     \StringEncodingConvert
219     \hyxmp@text\hyxmp@text{pdfdoc}{utf8}%
220   }%
  UTF-8 → UTF-8 as hex string
221   \EdefEscapeHex\hyxmp@text{\hyxmp@text}%
  UTF-8 as hex string → XML in UTF-8 as hex string
222   \edef\hyxmp@text{%
223     \expandafter\hyxmp@toxml\hyxmp@text\@empty\@empty
224   }%
  XML in UTF-8 as hex string → XML in UTF-8
225   \EdefUnescapeHex\hyxmp@text{\hyxmp@text}%
226   \fi
227   \global\let\hyxmp@xmlified\hyxmp@text
228 }

```

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

```

229 \begingroup
230   \lccode'\<=254 %
231   \lccode'\>=255 %
232   \catcode254=12 %
233   \catcode255=12 %
234 \lowercase{\endgroup
235   \def\hyxmp@is@unicode#1{%
236     \expandafter\hyxmp@@is@unicode#1<>\@nil
237   }%
238   \def\hyxmp@@is@unicode#1<>#2\@nil{%
239     \ifx\#1\%
240       \expandafter\@firstoftwo
241     \else
242       \expandafter\@secondoftwo

```

```

243     \fi
244   }%
245 }

```

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

```

246 \def\hyxmp@toxml#1#2{%
247   \ifx#1\@empty
248   \else
249     \ifnum"#1#2='\& %
250       26616D703B% &amp;
251     \else\ifnum"#1#2='\< %
252       266C743B% &lt;
253     \else\ifnum"#1#2='\> %
254       2667743B% &gt;
255     \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.

```

256     \@ifundefined{pdfmark}{%
257       #1#2%
258     }{%
259       \ifnum"#1#2='\( %
260         5C28% \(
261       \else\ifnum"#1#2='\) %
262         5C29% \)
263       \else
264         #1#2%
265       \fi\fi
266     }%
267   \fi\fi\fi
268   \expandafter\hyxmp@toxml
269   \fi
270 }

```

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

`\hyxmp@text`

```

271 \def\hyxmp@toxml@unicodetex#1#2#3#4#5#6#7#8{%
272   \ifx#1\relax
273   \else

```

```

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

```

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

```

296 \def\hyxmp@skipzeros#1{%
297 \ifx#10%
298 \expandafter\hyxmp@skipzeros
299 \fi
300 }

```

`\x` In the case of X_YTEX, the strings defined by `\pdfstringdef` can contain big characters. In this case, the string is treated as Unicode.

```

\hyxmp@xetex@crap \hyxmp@try 301 \begingroup
\hyxmp@crap@result 302 \def\x#1{\endgroup
\hyxmp@text 303 \def\hyxmp@xetex@crap{%
304 \edef\hyxmp@try{%
305 \expandafter\hyxmp@SpaceOther\hyxmp@text#1\@nil
306 }%
307 \let\hyxmp@crap@result=N%
308 \expandafter\hyxmp@crap@test\hyxmp@try\relax
309 \ifx\hyxmp@crap@result Y%
310 \let\hyxmp@text\@empty
311 \expandafter\hyxmp@crap@convert\hyxmp@try\relax
312 \else
313 \StringEncodingConvert\hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
314 \fi
315 }%
316 }
317 \x{ }

```

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

```

318 \begingroup
319 \catcode'\~=12 %
320 \lccode'\~=\' %
321 \lowercase{\endgroup
322 \def\hyxmp@SpaceOther#1 #2\@nil{%
323   #1%
324   \ifx\relax#2\relax
325     \expandafter\@gobble
326   \else
327     ~%
328     \expandafter\@firstofone
329   \fi
330   {\hyxmp@SpaceOther#2\@nil}%
331 }%
332 }

```

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

```

333 \def\hyxmp@crap@test#1{%
334   \ifx#1\relax
335   \else
336     \ifnum'#1>127 %
337       \let\hyxmp@crap@result=Y%
338       \expandafter\expandafter\expandafter\hyxmp@skiptorelax
339     \else
340       \expandafter\expandafter\expandafter\hyxmp@crap@test
341     \fi
342   \fi
343 }

```

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

```

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

```

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

```

\hyxmp@num 345 \def\hyxmp@crap@convert#1{%
\hyxmp@text 346   \ifx#1\relax
347   \else
348     \edef\hyxmp@num{\number'#1}%
349     \ifnum\hyxmp@num>"FFFFFF %
350       \lccode'\!=\intcalcDiv{\hyxmp@num}{\number"1000000}\relax
351       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
352     \edef\hyxmp@num{\intcalcMod{\hyxmp@num}{\number"1000000}}%
353   \else
354     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
355   \fi
356   \ifnum\hyxmp@num>"FFFF %
357     \lccode'\!=\intcalcDiv{\hyxmp@num}{\number"10000}\relax
358     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
359   \edef\hyxmp@num{\intcalcMod{\hyxmp@num}{\number"10000}}%

```

```

360 \else
361 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
362 \fi
363 \ifnum\hyxmp@num>"FF %
364 \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"100}\relax
365 \lowercase{\edef\hyxmp@text{\hyxmp@text!}}}%
366 \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"100}}}%
367 \else
368 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
369 \fi
370 \ifnum\hyxmp@num>0 %
371 \lccode'\!=\hyxmp@num\relax
372 \lowercase{\edef\hyxmp@text{\hyxmp@text!}}}%
373 \else
374 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
375 \fi
376 \expandafter\hyxmp@crap@convert
377 \fi
378 }

```

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

```

379 \begingroup
380 \catcode0=12 %
381 \gdef\hyxmp@zero{^^00}%
382 \endgroup

```

3.4 UUID generation

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

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

```

383 \def\hyxmp@modulo@a#1{%
384 \@tempcntb=\@tempcnta
385 \divide\@tempcntb by #1
386 \multiply\@tempcntb by #1
387 \advance\@tempcnta by -\@tempcntb
388 }

```

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

```

\hyxmp@big@prime@ii 389 \def\hyxmp@big@prime{536870923}
390 \def\hyxmp@big@prime@iii{536870027}

```

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

```

\hyxmp@one@token 391 \def\hyxmp@seed@rng#1{%
392 \@tempcnta=\hyxmp@big@prime
393 \futurelet\hyxmp@one@token\hyxmp@seed@rng@i#1\@empty
394 }

```

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

```

\hyxmp@one@token
\next 395 \def\hyxmp@seed@rng@i{%
396   \ifx\hyxmp@one@token\@empty
397     \let\next=\relax
398   \else
399     \def\next##1{%
400       \multiply\@tempcnta by 3
401       \advance\@tempcnta by '##1
402       \hyxmp@modulo@a{\hyxmp@big@prime}%
403       \futurelet\hyxmp@one@token\hyxmp@seed@rng@i
404     }%
405   \fi
406 \next
407 }

```

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

```

\hyxmp@rand@num
408 \def\hyxmp@set@rand@num{%
409   \@tempcnta=\hyxmp@rand@num
410   \multiply\@tempcnta by 3
411   \advance\@tempcnta by \hyxmp@big@prime@ii
412   \hyxmp@modulo@a{\hyxmp@big@prime}%
413   \xdef\hyxmp@rand@num{\the\@tempcnta}%
414 }

```

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

```

415 \def\hyxmp@append@hex#1{%
416   \hyxmp@set@rand@num
417   \@tempcnta=\hyxmp@rand@num
418   \hyxmp@modulo@a{16}%
419   \ifnum\@tempcnta<10
420     \xdef#1{#1\the\@tempcnta}%
421   \else
422     \advance\@tempcnta by -10
423     \ifcase\@tempcnta
424       \xdef#1{#1a}%
425     \or\xdef#1{#1b}%
426     \or\xdef#1{#1c}%
427     \or\xdef#1{#1d}%
428     \or\xdef#1{#1e}%
429     \or\xdef#1{#1f}%
430   \fi
431 \fi
432 }

```

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

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

```
433 \def\hyxmp@append@hex@iv#1{%  
434   \hyxmp@append@hex#1%  
435   \hyxmp@append@hex#1%  
436   \hyxmp@append@hex#1%  
437   \hyxmp@append@hex#1%  
438 }
```

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

```
439 \def\hyxmp@create@uuid#1{%  
440   \def#1{uuid:}%  
441   \hyxmp@append@hex@iv#1%  
442   \hyxmp@append@hex@iv#1%  
443   \g@addto@macro#1{-}%  
444   \hyxmp@append@hex@iv#1%  
445   \g@addto@macro#1{-}%  
446   \hyxmp@append@hex@iv#1%  
447   \g@addto@macro#1{-}%  
448   \hyxmp@append@hex@iv#1%  
449   \hyxmp@append@hex@iv#1%  
450   \hyxmp@append@hex@iv#1%  
451 }
```

`\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.

```
452 \newcommand*{\hyxmp@def@DocumentID}{%  
453   \edef\hyxmp@seed@string{\jobname:\@pdftitle:\@pdfauthor}%  
454   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%  
455   \edef\hyxmp@rand@num{\the\@tempcnta}%  
456   \hyxmp@create@uuid\hyxmp@DocumentID  
457 }
```

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

```
458 \newcommand*{\hyxmp@def@InstanceID}{%  
459   \edef\hyxmp@seed@string{%  
460     \jobname:\@pdftitle:\@pdfauthor:%  
461     \the\year/\the\month/\the\day:%  
462     \the\time  
463   }%  
464   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%  
465   \edef\hyxmp@rand@num{\the\@tempcnta}%
```

```

466 \hyxmp@create@uuid\hyxmp@InstanceID
467 }

```

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), and IPTC Photo Metadata (Section 3.5.8). The `\hyxmp@construct@packet` macro constructs the XMP packet into `\hyxmp+xml`. It first writes the appropriate XML header, then calls the various schema-writing macros, then injects `\hyxmp@padding` as padding, and finally writes the appropriate XML trailer.

3.5.1 XMP utility functions

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

```

468 \newcommand*{\hyxmp@add@to+xml}[1]{%
469   \bgroup
470   \@tempcnta=0
471   \loop
472     \lccode\@tempcnta=\@tempcnta
473     \advance\@tempcnta by 1
474     \ifnum\@tempcnta<256
475       \repeat
476       \lccode'\_='\ \relax
477       \lccode'\^C='\,\relax
478       \lowercase{\xdef\hyxmp+xml{\hyxmp+xml#1}}%
479   \egroup
480 }

```

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

```

481 \bgroup
482 \catcode'\#=11
483 \gdef\hyxmp@hash{#}
484 \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.

```

485 \bgroup
486 \xdef\hyxmp+xml{%
487   \hyxmp@add@to+xml{%

```

```

488 -----^^J%
489   }
490   \xdef\hyxmp@padding{\hyxmp@xml}%
491 \egroup
492 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
493 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
494 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
495 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
496 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}

\hyxmp@today Define today's date in YYYY-MM-DD format.
497 \xdef\hyxmp@today{\the\year}%
498 \ifnum\month<10
499   \xdef\hyxmp@today{\hyxmp@today-0\the\month}%
500 \else
501   \xdef\hyxmp@today{\hyxmp@today-\the\month}%
502 \fi
503 \ifnum\day<10
504   \xdef\hyxmp@today{\hyxmp@today-0\the\day}%
505 \else
506   \xdef\hyxmp@today{\hyxmp@today-\the\day}%
507 \fi

\hyxmp@x@default Define an x-default string that we can use in comparisons with \@pdfmetalang.
508 \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.
509 \newcommand*{\hyxmp@pdf@schema}{%

\hyxmp@have@any Include an Adobe PDF schema block if at least one of \@pdfkeywords and
\@pdfproducer is defined.
510   \let\hyxmp@have@any=!%
511   \ifx\@pdfkeywords\@empty
512     \ifx\@pdfproducer\@empty
513       \let\hyxmp@have@any=\@empty
514     \fi
515   \fi
516   \ifx\hyxmp@have@any\@empty
517   \else

Add a block of XML to \hyxmp@xml that lists the document's keywords (the
pdf:Keywords property) and the tools used to produce the PDF file (the pdf:Producer
property).
518   \hyxmp@add@to@xml{%
519   -----<rdf:Description rdf:about="^^J%
520   -----xmlns:pdf="http://ns.adobe.com/pdf/1.3/">^^J%
521     }%

```

```

522 \hyxmp@add@simple{pdf:Keywords}{\@pdfkeywords}%
523 \hyxmp@add@simple{pdf:Producer}{\@pdfproducer}%
524 \@ifundefined{pdfminorversion}{}{%
525 \hyxmp@add@simple{pdf:PDFVersion}{1.\the\pdfminorversion}%
526 }%
527 \hyxmp@add@to@xml{%
528 -----</rdf:Description>^^J%
529 }%
530 \fi
531 }

```

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

\hyxmp@string

```

532 \newcommand*{\hyxmp@add@simple}[2]{%
533 \edef\hyxmp@string{#2}%
534 \ifx\hyxmp@string\@empty
535 \else
536 \hyxmp@xmlify{\hyxmp@string}%
537 \hyxmp@add@to@xml{%
538 -----<#1>\hyxmp@xmlified</#1>^^J%
539 }%
540 \fi
541 }

```

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.

```

542 \newcommand*{\hyxmp@rdf@dc}[2]{%
543 \ifx#2\@empty
544 \else
545 \hyxmp@xmlify{#2}%
546 \hyxmp@add@to@xml{%
547 -----<dc:#1>^^J%
548 -----<rdf:Alt>^^J%
549 }%
550 \ifx\@pdfmetalang\hyxmp@x@default
551 \else
552 \hyxmp@add@to@xml{%
553 -----<rdf:li xml:lang="\@pdfmetalang">\hyxmp@xmlified</rdf:li>^^J%
554 }%
555 \fi
556 \hyxmp@add@to@xml{%
557 -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@xmlified</rdf:li>^^J%
558 -----</rdf:Alt>^^J%
559 -----</dc:#1>^^J%
560 }%

```

```

561 \fi%
562 }%

```

`\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.

```

563 \newcommand*{\hyxmp@list@to@xml}[3]{%
564   \ifx#3\@empty
565   \else
566     \hyxmp@add@to@xml{%
567       -----<dc:#1>^^J%
568       -----<rdf:#2>^^J%
569     }%
570   \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`.

```

571   \hyxmp@xmlify{#3}%
572   \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
573   \def\@elt##1{%
574     \hyxmp@add@to@xml{%
575       -----<rdf:li>##1</rdf:li>^^J%
576     }%
577   }%
578   \hyxmp@list
579   \egroup
580   \hyxmp@add@to@xml{%
581     -----</rdf:#2>^^J%
582     -----</dc:#1>^^J%
583   }%
584 \fi
585 }

```

`\hyxmp@dc@schema` Add properties defined by the Dublin Core schema to the `\hyxmp@xml` macro. Specifically, we add entries for the `dc:title` property if the author specified a `pdftitle`, the `dc:description` property if the author specified a `pdfsubject`, the `dc:rights` property if the author specified a `pdfcopyright`, the `dc:creator` property if the author specified a `pdfauthor`, the `dc:subject` property if the author specified `pdfkeywords`, and the `dc:language` property if the author specified `pdflang`. We also specify the `dc:date` property using the date the document was run through L^AT_EX and the `dc:source` property using the base name of the source file with `.tex` appended.

```

586 \newcommand*{\hyxmp@dc@schema}{%
587   \hyxmp@add@to@xml{%
588     -----<rdf:Description rdf:about=""^^J%
589     -----xmlns:dc="http://purl.org/dc/elements/1.1/"^^J%
590     -----<dc:format>application/pdf</dc:format>^^J%
591   }%

```

```

592 \hyxmp@rdf@dc{title}{\@pdftitle}%
593 \hyxmp@rdf@dc{description}{\@pdfsubject}%
594 \hyxmp@rdf@dc{rights}{\@pdfcopyright}%
595 \hyxmp@list@to@xml{creator}{Seq}{\hyxmp@pdfauthor}%
596 \hyxmp@list@to@xml{subject}{Bag}{\hyxmp@pdfkeywords}%
597 \hyxmp@list@to@xml{date}{Seq}{\hyxmp@today}%
598 \hyxmp@add@simple{dc:language}{\@pdflang}%
599 \hyxmp@add@simple{dc:source}{\jobname.tex}%
600 \hyxmp@add@to@xml{%
601 -----</rdf:Description>^^J%
602 }%
603 }

```

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.

```

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

```

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

```

605 \let\hyxmp@rights=\@empty
606 \ifx\@pdflicenseurl\@empty
607 \else
608 \def\hyxmp@rights{YES}%
609 \fi
610 \ifx\@pdfcopyright\@empty
611 \else
612 \def\hyxmp@rights{YES}%
613 \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.

```

614 \ifx\hyxmp@rights\@empty
615 \else
Header
616 \hyxmp@add@to@xml{%
617 -----<rdf:Description rdf:about=""^^J%
618 -----xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/">^^J%
619 }%

```

Copyright indication

```

620 \ifx\@pdfcopyright\@empty
621 \else
622 \hyxmp@add@to@xml{%
623 -----<xmpRights:Marked>True</xmpRights:Marked>^^J%
624 }%
625 \fi

```

License URL

```

626     \hyxmp@add@simple{xmpRights:WebStatement}{\@pdflicenseurl}%
Trailer
627     \hyxmp@add@to@xml{%
628     -----</rdf:Description>^^J%
629     }%
630     \fi
631 }

```

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 T_EX-based workflow.

```

632 \gdef\hyxmp@mm@schema{%
633     \hyxmp@def@DocumentID
634     \hyxmp@def@InstanceID
635     \hyxmp@add@to@xml{%
636     -----<rdf:Description rdf:about=""^^J%
637     -----_xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"^^J%
638     -----<xmpMM:DocumentID>\hyxmp@DocumentID</xmpMM:DocumentID>^^J%
639     -----<xmpMM:InstanceID>\hyxmp@InstanceID</xmpMM:InstanceID>^^J%
640     -----</rdf:Description>^^J%
641     }%
642 }

```

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`.

```

643 \newcommand*{\hyxmp@xmp@basic@schema}{%
644     \hyxmp@add@to@xml{%
645     -----<rdf:Description rdf:about=""^^J%
646     -----_xmlns:xmp="http://ns.adobe.com/xap/1.0/"^^J%
647     }%
648     \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@today}%
649     \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@today}%
650     \hyxmp@add@simple{xmp:MetadataDate}{\hyxmp@today}%
651     \hyxmp@add@simple{xmp:CreatorTool}{\@pdfcreator}%
652     \hyxmp@add@simple{xmp:BaseURL}{\@baseurl}%
653     \hyxmp@add@to@xml{%
654     -----</rdf:Description>^^J%
655     }%
656 }

```

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.

```

657 \gdef\hyxmp@photoshop@schema{%
658   \edef\hyxmp@photoshop@data{\@pdfauthortitle\@pdfcaptionwriter}%
659   \ifx\hyxmp@photoshop@data\@empty
660     \else
661       \hyxmp@add@to@xml{%
662         <rdf:Description rdf:about=""^^J%
663         -----xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/">^^J%
664       }%
665     \fi
666     \hyxmp@add@simple{photoshop:AuthorsPosition}{\@pdfauthortitle}%
667     \hyxmp@add@simple{photoshop:CaptionWriter}{\@pdfcaptionwriter}%
668     \ifx\hyxmp@photoshop@data\@empty
669     \else
670       \hyxmp@add@to@xml{%
671         </rdf:Description>^^J%
672       }%
673     \fi
674 }
```

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.

```

675 \begingroup
676   \catcode'\&=12
677   \catcode'\#=12
678   \gdef\xmplinesep{&#xA;}
679 \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.

```

680 \newcommand*{\hyxmp@list@to@lines}[2]{%
681   \ifx#2\@empty
682     \else
683       \bgroup
684         \hyxmp@add@to@xml{%
685         -----<#1>%
686       }%
```

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

```

687   \def\@elt@first##1{%
688     \hyxmp@add@to@xml{##1}%
689     \let\@elt=\@elt@rest
```



```

690      }%

\@elt@rest  The remaining elements of the list are output with a preceding line separator
              (\xmplinesep).
691      \def\@elt@rest##1{%
692          \hyxmp@add@to@xml{\xmplinesep##1}%
693      }%

\@elt  Re-encode the text from Unicode if necessary. Then redefine \@elt to insert a line
        separator between terms.
694      \let\@elt=\@elt@first
695      \hyxmp@xmlify{#2}%
696      \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
697      \hyxmp@list
698      \hyxmp@add@to@xml{</#1>^^J}%
699      \egroup
700  \fi
701 }

\hyxmp@photometa@schema  Add properties defined by the IPTC Photo Metadata schema [6] to the
\hyxmp@photometa@data    \hyxmp@xml macro. We currently support only the contact-information
                          details structure, viz. the Iptc4xmpCore:CreatorContactInfo/CiAdrExtadr,
                          Iptc4xmpCore:CreatorContactInfo/CiAdrCity, Iptc4xmpCore:CreatorContactInfo/
                          CiAdrRegion, Iptc4xmpCore:CreatorContactInfo/CiAdrPcode,
                          Iptc4xmpCore:CreatorContactInfo/CiAdrCtry, Iptc4xmpCore:CreatorContactInfo/
                          CiTelWork, Iptc4xmpCore:CreatorContactInfo/CiEmailWork, and
                          Iptc4xmpCore:CreatorContactInfo/CiUrlWork properties.
702 \gdef\hyxmp@photometa@schema{%
703   \edef\hyxmp@photometa@data{%
704     \@pdfcontactaddress
705     \@pdfcontactcity
706     \@pdfcontactregion
707     \@pdfcontactpostcode
708     \@pdfcontactcountry
709     \@pdfcontactphone
710     \@pdfcontactemail
711     \@pdfcontacturl
712   }%
713   \ifx\hyxmp@photometa@data\@empty
714   \else
715     \hyxmp@iptc@extensions
716     \hyxmp@add@to@xml{%
717       <rdf:Description rdf:about=""^^J%
718       <xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"^^J%
719       <xmlns:IptcContInfo="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/contactinfo/">^^J%
720       <Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">^^J%
721     }%
722   \fi
723   \hyxmp@list@to@lines{IptcContInfo:CiAdrExtadr}{\@pdfcontactaddress}%

```

```

724 \hyxmp@add@simple{IptcContInfo: CiAdrCity}{\@pdfcontactcity}%
725 \hyxmp@add@simple{IptcContInfo: CiAdrRegion}{\@pdfcontactregion}%
726 \hyxmp@add@simple{IptcContInfo: CiAdrPcode}{\@pdfcontactpostcode}%
727 \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.

```

728 \bgroup
729 \def\xmplinesep{,}%
730 \hyxmp@list@to@lines{IptcContInfo: CiTelWork}{\@pdfcontactphone}%
731 \hyxmp@list@to@lines{IptcContInfo: CiEmailWork}{\@pdfcontactemail}%
732 \hyxmp@list@to@lines{IptcContInfo: CiUrlWork}{\@pdfcontacturl}%
733 \egroup
734 \ifx\hyxmp@photometa@data\@empty
735 \else
736 \hyxmp@add@to@xml{%
737 -----</Iptc4xmpCore:CreatorContactInfo>^^J%
738 -----</rdf:Description>^^J%
739 }%
740 \fi
741 }

```

`\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 [8], 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.

```

742 \newcommand*{\hyxmp@iptc@extensions}{%
743 \hyxmp@add@to@xml{%
744 -----<rdf:Description rdf:about=""^^J%
745 -----xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"^^J%
746 -----xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema\hyxmp@hash"^^J%
747 -----xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property\hyxmp@hash"^^J%
748 -----xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type\hyxmp@hash"^^J%
749 -----xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field\hyxmp@hash"^^J%
750 -----<pdfaExtension:schemas>^^J%
751 -----<rdf:Bag>^^J%
752 -----<rdf:li rdf:parseType="Resource">^^J%
753 -----<pdfaSchema:schema>IPTC Core Schema</pdfaSchema:schema>^^J%
754 -----<pdfaSchema:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaSchema:
755 -----<pdfaSchema:prefix>Iptc4xmpCore</pdfaSchema:prefix>^^J%
756 -----<pdfaSchema:property>^^J%

```

```

757 -----<rdf:Seq>^^J%
758 -----<rdf:li rdf:parseType="Resource">^^J%
759 -----<pdfaProperty:name>CreatorContactInfo</pdfaProperty:name>^^J%
760 -----<pdfaProperty:valueType>contactinfo</pdfaProperty:valueType>^^J%
761 -----<pdfaProperty:category>external</pdfaProperty:category>^^J%
762 -----<pdfaProperty:description>contact information for the document's creator</p
763 -----</rdf:li>^^J%
764 -----</rdf:Seq>^^J%
765 -----</pdfaSchema:property>^^J%
766 -----<pdfaSchema:valueType>^^J%
767 -----<rdf:Seq>^^J%
768 -----<rdf:li rdf:parseType="Resource">^^J%
769 -----<pdfaType:type>contactinfo</pdfaType:type>^^J%
770 -----<pdfaType:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/contactin
771 -----<pdfaType:prefix>IptcContInfo</pdfaType:prefix>^^J%
772 -----<pdfaType:description>contact information</pdfaType:description>^^J%
773 -----<pdfaType:field>^^J%
774 -----<rdf:Seq>^^J%
775 }%

776 \hyxmp@text@resource{CiAdrExtadr}{contact address}%
777 \hyxmp@text@resource{CiAdrCity}{contact city}%
778 \hyxmp@text@resource{CiAdrRegion}{contact region}%
779 \hyxmp@text@resource{CiAdrPcode}{contact postal code}%
780 \hyxmp@text@resource{CiAdrCtry}{contact country}%
781 \hyxmp@text@resource{CiTelWork}{contact telephone number}%
782 \hyxmp@text@resource{CiEmailWork}{contact email address}%
783 \hyxmp@text@resource{CiUrlWork}{contact url}%

784 \hyxmp@add@to@xml{%
785 -----</rdf:Seq>^^J%
786 -----</pdfaType:field>^^J%
787 -----</rdf:li>^^J%
788 -----</rdf:Seq>^^J%
789 -----</pdfaSchema:valueType>^^J%
790 -----</rdf:li>^^J%
791 -----</rdf:Bag>^^J%
792 -----</pdfaExtension:schemas>^^J%
793 -----</rdf:Description>^^J%
794 }%
795 }

```

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

```

796 \newcommand*{\hyxmp@text@resource}[2]{%
797   \hyxmp@add@to@xml{%
798 -----<rdf:li rdf:parseType="Resource">^^J%
799 -----<pdfaField:name>#1</pdfaField:name>^^J%
800 -----<pdfaField:valueType>Text</pdfaField:valueType>^^J%
801 -----<pdfaField:description>#2</pdfaField:description>^^J%
802 -----</rdf:li>^^J%
803   }

```

```
804 }
```

3.5.9 Constructing the XMP packet

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

```
805 \begingroup
806   \ifhyxmp@unicodetex
807     \lccode'\!="FEFF %
808     \lowercase{%
809       \gdef\hyxmp@bom{!}
810     }%
811   \else
812     \catcode'\^^ef=12
813     \catcode'\^^bb=12
814     \catcode'\^^bf=12
815     \gdef\hyxmp@bom{^^ef^^bb^^bf}%
816   \fi
817 \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.

```
818 \def\hyxmp@construct@packet{%
819   \gdef\hyxmp+xml{%
820     \hyxmp@add@to+xml{<?xpacket begin="\hyxmp@bom" %
821 id="W5MOMpCehiHzreSzNTczkc9d"?>^^J%
822 <x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="3.1-702">^^J%
823 ___<rdf:RDF
824 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\hyxmp@hash">^^J%
825   }%
826   \hyxmp@pdf@schema
827   \hyxmp@xmpRights@schema
828   \hyxmp@dc@schema
829   \hyxmp@photoshop@schema
830   \hyxmp@photometa@schema
831   \hyxmp@xmp@basic@schema
832   \hyxmp@mm@schema
833   \hyxmp@add@to+xml{%
834 ___</rdf:RDF>^^J%
835 </x:xmpmeta>^^J%
836 \hyxmp@padding
837 <?xpacket end="w"?>^^J%
838   }%
839 }
```

3.6 Embedding the XMP packet

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

`\hyxmp@embed@packet` Determine which hyperref driver is in use and invoke the appropriate embedding
`\hyxmp@driver` function.

```

840 \newcommand*{\hyxmp@embed@packet}{%
841   \hyxmp@construct@packet
842   \def\hyxmp@driver{hpdfTeX}%
843   \ifx\hyxmp@driver\Hy@driver
844     \hyxmp@embed@packet@pdfTeX
845   \else
846     \def\hyxmp@driver{hdvipdfm}%
847     \ifx\hyxmp@driver\Hy@driver
848       \hyxmp@embed@packet@dviPDFm
849     \else
850       \def\hyxmp@driver{Hxetex}%
851       \ifx\hyxmp@driver\Hy@driver
852         \hyxmp@embed@packet@xetex
853       \else
854         \@ifundefined{pdfmark}{%
855           \PackageWarningNoLine{hyperxmp}{%
856             Unrecognized hyperref driver ‘\Hy@driver’. \MessageBreak
857             \jobname.tex’s XMP metadata will *not* be \MessageBreak
858             embedded in the resulting file}%
859         }{%
860           \hyxmp@embed@packet@pdfmark
861         }%
862       \fi
863     \fi
864   \fi
865 }
```

3.6.1 Embedding using pdfTeX

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

```

866 \newcommand*{\hyxmp@embed@packet@pdfTeX}{%
867   \bgroup
868   \pdfcompresslevel=0
869   \immediate\pdfobj stream attr {%
870     /Type /Metadata
871     /Subtype /XML
872   }{\hyxmp@xml}%
873   \pdfcatalog {/Metadata \the\pdflastobj\space 0 R}%
874   \egroup
875 }
```

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 `dviPDFm`, `dvipsone`, `dvips`, `dviwindo`, `nativePDFm`, `pdfmark`, `ps2PDFm`, `textures`, and `vtexPDFm` options to hyperref but I’ve tested only a few of those.

```

876 \newcommand*{\hyxmp@embed@packet@pdfmark}{%
```

```

877 \pdfmark{%
878   pdfmark=/NamespacePush
879 }%
880 \pdfmark{%
881   pdfmark=/OBJ,
882   Raw={/_objdef \string{hyxmp@Metadata\string} /type /stream}%
883 }%
884 \pdfmark{%
885   pdfmark=/PUT,
886   Raw={\string{hyxmp@Metadata\string}
887     2 dict begin
888       /Type /Metadata def
889       /Subtype /XML def
890       currentdict
891     end
892   }%
893 }%
894 \pdfmark{%
895   pdfmark=/PUT,
896   Raw={\string{hyxmp@Metadata\string} (\hyxmp@xml)}%
897 }%
898 \pdfmark{%
899   pdfmark=/Metadata,
900   Raw={\string{Catalog\string} \string{hyxmp@Metadata\string}}%
901 }%
902 \pdfmark{%
903   pdfmark=/NamespacePop
904 }%
905 }

```

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.

```

906 \newcommand*{\hyxmp@embed@packet@dvipdfm}{%
907   \hyxmp@string@len{\hyxmp@xml}%
908   \special{pdf: object @hyxmp@Metadata
909     <<
910       /Type /Metadata
911       /Subtype /XML
912       /Length \the\@tempcnta
913     >>
914     stream^^J\hyxmp@xml endstream%
915   }%
916   \special{pdf: docview
917     <<
918       /Metadata @hyxmp@Metadata
919     >>

```

```

920 }%
921 }

```

`\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.

```

922 \newcommand*{\hyxmp@string@len}[1]{%
923   \@tempcnta=0
924   \expandafter\hyxmp@count@spaces#1 {} %
925   \expandafter\hyxmp@count@non@spaces#1{}%
926 }

```

`\hyxmp@count@spaces` Count the number of spaces in a given string. We rely on the built-in pattern matching of T_EX's `\def` primitive to pry one word at a time off the head of the input string.

```

927 \def\hyxmp@count@spaces#1 {%
928   \def\hyxmp@one@token{#1}%
929   \ifx\hyxmp@one@token\empty
930     \advance\@tempcnta by -1
931   \else
932     \advance\@tempcnta by 1
933     \expandafter\hyxmp@count@spaces
934   \fi
935 }

```

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

```

936 \newcommand*{\hyxmp@count@non@spaces}[1]{%
937   \def\hyxmp@one@token{#1}%
938   \ifx\hyxmp@one@token\empty
939     \else
940       \advance\@tempcnta by 1
941       \expandafter\hyxmp@count@non@spaces
942     \fi
943 }

```

3.6.4 Embedding using X_YT_EX

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

```

944 \newcommand*{\hyxmp@embed@packet@xetex}{%
945   \special{pdf:stream @hyxmp@Metadata (\hyxmp+xml)
946     <<
947     /Type /Metadata

```

```

948      /Subtype /XML
949    >>
950  }%
951  \special{pdf:put @catalog
952    <<
953      /Metadata @hyxmp@Metadata
954    >>
955  }%
956 }

```

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.

```

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

```

4 Future Work

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

A Sample XMP packet

The following is an example of a complete XMP packet as may be produced by `hyperxmp`. This packet corresponds to the metadata included in the sample \LaTeX 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>
    </rdf:Description>
  </rdf:RDF>
</x:xmpmeta>
</xpacket>

```



```

<xmpRights:WebStatement>
  http://creativecommons.org/licenses/by-nc-nd/3.0/
</xmpRights:WebStatement>
</rdf:Description>
<rdf:Description rdf:about=""
  xmlns:dc="http://purl.org/dc/elements/1.1/">
  <dc:format>application/pdf</dc:format>
  <dc:title>
    <rdf:Alt>
      <rdf:li xml:lang="en">
        On a heuristic viewpoint concerning the production and
        transformation of light
      </rdf:li>
      <rdf:li xml:lang="x-default">
        On a heuristic viewpoint concerning the production and
        transformation of light
      </rdf:li>
    </rdf:Alt>
  </dc:title>
  <dc:description>
    <rdf:Alt>
      <rdf:li xml:lang="en">photoelectric effect</rdf:li>
      <rdf:li xml:lang="x-default">photoelectric effect</rdf:li>
    </rdf:Alt>
  </dc:description>
  <dc:rights>
    <rdf:Alt>
      <rdf:li xml:lang="en">
        Copyright \(C\) 1905, Albert Einstein
      </rdf:li>
      <rdf:li xml:lang="x-default">
        Copyright \(C\) 1905, Albert Einstein
      </rdf:li>
    </rdf:Alt>
  </dc:rights>
  <dc:creator>
    <rdf:Seq>
      <rdf:li>Albert Einstein</rdf:li>
    </rdf:Seq>
  </dc:creator>
  <dc:subject>
    <rdf:Bag>
      <rdf:li>energy quanta</rdf:li>
      <rdf:li>Hertz effect</rdf:li>
      <rdf:li>quantum physics</rdf:li>
    </rdf:Bag>
  </dc:subject>

```

```

</dc:subject>
<dc:date>
  <rdf:Seq>
    <rdf:li>2013-01-09</rdf:li>
  </rdf:Seq>
</dc:date>
<dc:language>en</dc:language>
<dc:source>einstein.tex</dc:source>
</rdf:Description>
<rdf:Description rdf:about=""
  xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/">
  <photoshop:AuthorsPosition>
    Technical Assistant, Level III
  </photoshop:AuthorsPosition>
  <photoshop:CaptionWriter>Scott Pakin</photoshop:CaptionWriter>
</rdf:Description>
<rdf:Description rdf:about=""
  xmlns: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/contactinf
<Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">
    <IptcContInfo:CiAdrExtadr>Kramgasse 49</IptcContInfo:CiAdrExtadr>
    <IptcContInfo:CiAdrCity>Bern</IptcContInfo:CiAdrCity>
    <IptcContInfo:CiAdrPcode>3011</IptcContInfo:CiAdrPcode>
    <IptcContInfo:CiAdrCtry>Switzerland</IptcContInfo:CiAdrCtry>
    <IptcContInfo:CiTelWork>031 312 00 91</IptcContInfo:CiTelWork>
    <IptcContInfo:CiEmailWork>aeinstein@ipi.ch</IptcContInfo:CiEmailWork>
    <IptcContInfo:CiUrlWork>
        http://einstein.biz/,
        https://www.facebook.com/AlbertEinstein
    </IptcContInfo:CiUrlWork>
</Iptc4xmpCore:CreatorContactInfo>
</rdf:Description>
<rdf:Description rdf:about=""
    xmlns:xmp="http://ns.adobe.com/xap/1.0/">
    <xmp:CreateDate>2013-01-09</xmp:CreateDate>
    <xmp:ModifyDate>2013-01-09</xmp:ModifyDate>
    <xmp:MetadataDate>2013-01-09</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.
- [4] Adobe Systems, Inc., San Jose, California. *XMP Specification Part 1: Data model, Serialization, and Core Properties*, July 2010. Available from <http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/xmp/pdfs/XMPSpecificationPart1.pdf>.
- [5] Michael Downes. Around the bend #15, answers, 4th (last) installment. `comp.text.tex` newsgroup posting, January 3, 1994. Archived by Google at <http://groups.google.com/group/comp.text.tex/msg/7da7643b9e8f3b48>.
- [6] 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.
- [7] Internet Assigned Numbers Authority. Language subtag registry, January 11, 2011. Available from <http://www.iana.org/assignments/language-subtag-registry>.
- [8] 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.

Change History

v1.0		$\langle BF \rangle$ to “letter”. Thanks to Daniel Schömer for the bug report
General: Initial version	1	36
v1.1		
<code>\hyxmp@construct@packet</code> : Explicitly set the category codes of characters $\langle EF \rangle$, $\langle BB \rangle$, and	v1.2	General: Added support for the Xe _{La} TeX backend (<code>xdvipdfmx</code>)
		1

Added support for the Photoshop schema	1	\hyxmp@construct@packet: Modified by Heiko Oberdiek to use an appropriate BOM representation via \hyxmp@bom	36
Made the package compatible with ngerman. Thanks to Tobias Mueller for the bug report. . . .	9	\hyxmp@crap@convert: Added by Heiko Oberdiek	22
v1.3		\hyxmp@crap@test: Added by Heiko Oberdiek	22
General: Introduced the pdfmetalang package option, which enables an author to specify the language in which he wrote the document's metadata	15	\hyxmp@dc@schema: Added support for dc:language and dc:source .	29
\hyxmp@reencode: Introduced this macro to re-encode Unicode strings as 8-bit strings before manipulating them into XMP schema. This change addresses a bug reported by Martin Münch	18	\hyxmp@is@unicode: Added by Heiko Oberdiek	19
v1.4		\hyxmp@list@to@xml: Modified by Heiko Oberdiek to use the new Unicode-processing macros . .	29
\hyxmp@mm@schema: Renamed the xapMM namespace prefix to xmpMM	31	\hyxmp@photoshop@schema: Simplified using \hyxmp@add@simple	32
\hyxmp@rdf@dc: Included metadata in the x-default language regardless of the specified metadata language	28	\hyxmp@ProcessKeyvalOptions: Added this macro	13
\hyxmp@xmpRights@schema: Renamed the xapRights namespace prefix to xmpRights	30	\hyxmp@reencode: Replaced with an empty macro by Heiko Oberdiek	18
v1.5		\hyxmp@skiptorelax: Added by Heiko Oberdiek	22
General: Made the XMP inclusion more robust. Thanks to Heiko Oberdiek for the bug report and suggested modifications.	9	\hyxmp@skipzeros: Added by Heiko Oberdiek	21
v2.0		\hyxmp@SpaceOther: Added by Heiko Oberdiek	22
General: Added support for the XMP Basic schema and miscellaneous other bits of metadata	1	\hyxmp@string: Added this macro	28
Heiko Oberdiek's major rewrite of the code to better support native-Unicode T _E X implementations (X _Ǝ T _E X and LuaT _E X) .	1	\hyxmp@toxml: Added by Heiko Oberdiek	20
New \AtBeginDocument code from Heiko Oberdiek to properly encode \pdfmetalang	15	Escaped parentheses written with pdfmarks to prevent dvips from line-wrapping the XMP packet	20
\hyxmp@add@to@xml: Updated also to replace commas	26	\hyxmp@toxml@unicodetex: Added by Heiko Oberdiek	20
\hyxmp@bom: Added by Heiko Oberdiek	36	\hyxmp@xetex@crap: Added by Heiko Oberdiek	21
\hyxmp@comma: Added this macro	16	\hyxmp@xmllify: Completely rewritten by Heiko Oberdiek to better support Unicode-enabled T _E X programs	18
		\hyxmp@xmp@basic@schema: Added this macro	31
		\hyxmp@xmpRights@schema: Modified to include xmpRights:Marked only when pdfcopyright is specified and xmpRights:WebStatement only	

when <code>pdflicenseurl</code> is specified	30	v2.2	General: Added support for the IPTC Photo Metadata schema	1
<code>\hyxmp@zero</code> : Added by Heiko Oberdiek	23	<code>\hyxmp@iptc@extensions</code> : Added this macro to support PDF/A generation	34	
<code>\ifhyxmp@unicodetex</code> : Added by Heiko Oberdiek	18	<code>\hyxmp@list@to@lines</code> : Added this macro	32	
<code>\ProcessKeyvalOptions</code> : Added this macro	13	<code>\hyxmp@photometa@schema</code> : Added this macro	33	
<code>\xmpcomma</code> : Added this macro	16	<code>\hyxmp@text@resource</code> : Added this macro	35	
<code>\xmpquote</code> : Added this macro	16	<code>\xmpcomma</code> : Changed the default from <code>\relax</code> to an ordinary comma	16	
<code>\XMPTruncateList</code> : Added this macro	16	<code>\xmplinesep</code> : Added this macro	32	
v2.1		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	34
General: Enabled <code>hyperxmp</code> and <code>hyperref</code> to be loaded in either order. This addresses a bug report by Yury Donskoy	12			
<code>\hypersetup</code> : Added this macro	13			
<code>\hyxmp@hypersetup</code> : Added this macro	13			
<code>\hyxmp@redefine@Hyp</code> : Added this macro	12			

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	<code>\@pdfauthor</code>	<code>\@pdfcontactregion</code>
<code>\!</code> 350, 357, 364, 371, 807	.. 56, 98, 453, 460	. 35, 107, 706, 725
<code>\"</code>	<code>\@pdfauthor title</code> ..	<code>\@pdfcontacturl</code> ..
<code>\#</code> 19, 99, 658, 666	. 45, 108, 711, 732
<code>\&</code>	<code>\@pdfcaptionwriter</code> .	<code>\@pdfcopyright</code> . 15,
<code>\(</code> 21, 100, 658, 667	109, 594, 610, 620
<code>\)</code>	<code>\@pdfcontactaddress</code>	<code>\@pdfcreator</code>
<code>*</code> 25, 101, 704, 723	<code>\@pdfkeywords</code>
<code>\,</code>	<code>\@pdfcontactcity</code> ..	. 73, 110, 511, 522
<code>\@baseurl</code> 33, 102, 705, 724	<code>\@pdflang</code>
<code>\@elt</code> . 166, 571, 689, 694	<code>\@pdfcontactcountry</code>	. 111, 126, 129, 598
<code>\@elt@first</code> 39, 103, 708, 727	<code>\@pdflicenseurl</code> ..
<code>\@elt@rest</code>	<code>\@pdfcontactemail</code> ..	. 17, 112, 606, 626
<code>\@firstofone</code> 43, 104, 710, 731	<code>\@pdfmetalang</code> 23, 127,
<code>\@firstoftwo</code>	<code>\@pdfcontactphone</code> ..	129, 131, 550, 553
<code>\@gobble</code> 41, 105, 709, 730	<code>\@pdfproducer</code> . 512, 523
<code>\@nil</code>	<code>\@pdfcontactpostcode</code>	<code>\@pdfsubject</code> .. 113, 593
238, 305, 322, 330	. 37, 106, 707, 726	<code>\@pdftitle</code>

. 114, 453, 460, 592	dvipsone (option) 37	\hyxmp@big@prime@ii
\@secondoftwo 242	dviwindo (option) . . . 37 <u>389</u> , 411
\^ 162, 186, 477, 812–814		\hyxmp@bom <u>805</u> , 820
_ 476	E	\hyxmp@comma
\~ 319, 320	\EdefEscapeHex 208, 221	. . . 27, 57, 74, <u>161</u>
	\EdefUnescapeHex . . . 225	\hyxmp@commas@to@list
	\EdefUnescapeString 195	. <u>145</u> , 168, 572, 696
_ 285, 320, 476	\endcsname	\hyxmp@commas@to@list@i
	53, 70, 167, 170, 192 147, <u>149</u>
A	ETX 16, 18	\hyxmp@concat@metadata
ASCII 19	 <u>95</u>
\AtBeginDocument . . . 124	G	\hyxmp@construct@packet
\AtEndDocument 5	\g@addto@macro <u>818</u> , 841
\AtEndDvi 8 443, 445, 447	\hyxmp@count@non@spaces
atenddvi 9	Ghostsript 6 925, <u>936</u>
Author 6, 16		\hyxmp@count@spaces
	H 924, <u>927</u>
B	\Hy@driver 4,	\hyxmp@crap@convert
baseurl (option) 3, 31	843, 847, 851, 856 311, <u>345</u>
BOM 36, 46	hyperref 1, 3–6,	\hyxmp@crap@result .
	10, 12, 14, 15, 37, 47 <u>301</u> , 337
C	\hypersetup <u>90</u>	\hyxmp@crap@test 308, <u>333</u>
CiAdrCity 3, 33	hyperxmp . 1–6, 8–12,	\hyxmp@create@uuid .
CiAdrCtry 3, 33	14–16, 18, 23, 40, 47 <u>439</u> , 456, 466
CiAdrExtadr 3, 33	\hyxmp@is@unicode . <u>229</u>	\hyxmp@dc@schema <u>586</u> , 828
CiAdrPcode 3, 33	\hyxmp@add@simple . .	\hyxmp@def@DocumentID
CiAdrRegion 3, 33	. 522, 523, 525, <u>452</u> , 633
CiEmailWork 3, 33	<u>532</u> , 598, 599,	\hyxmp@def@InstanceID
CiTelWork 3, 33	626, 648–652, <u>458</u> , 634
CiUrlWork 3, 33	666, 667, 724–727	\hyxmp@DocumentID . .
D	\hyxmp@add@to+xml <u>452</u> , 638
\day . . 461, 503, 504, 506 <u>468</u> ,	\hyxmp@dq@code . . <u>1</u> , 957
dc:creator 2, 6, 29	487, 518, 527,	\hyxmp@driver . . . <u>3</u> , <u>840</u>
dc:date 2, 29	537, 546, 552,	\hyxmp@embed@packet
dc:description 2, 29	556, 566, 574, 134, <u>840</u>
dc:format 2	580, 587, 600,	\hyxmp@embed@packet@dvi
dc:language 2, 29, 46	616, 622, 627, 848, <u>906</u>
dc:rights 2, 29	635, 644, 653,	\hyxmp@embed@packet@pdfmark
dc:source 2, 29, 46	661, 670, 684, 860, <u>876</u>
dc:subject 2, 29	688, 692, 698,	\hyxmp@embed@packet@pdftex
dc:title 2, 29	716, 736, 743, 844, <u>866</u>
\define@key	784, 797, 820, 833	\hyxmp@embed@packet@xetex
. 16, 18, 20, 22,	\hyxmp@append@hex 852, <u>944</u>
24, 26, 34, 36, 38, <u>415</u> , 434–437	\hyxmp@find@metadata
40, 42, 44, 46, 56, 73	\hyxmp@append@hex@iv <u>95</u> , 133
dvipdf (option) 37	. <u>433</u> , 441, 442,	\hyxmp@hash
dvipdfm 38	444, 446, 448–450	. <u>481</u> , 746–749, 824
dvips (option) 37	\hyxmp@at@end . . . <u>3</u> , 132	\hyxmp@have@any . . . <u>510</u>
dvips 6, 20, 46	\hyxmp@big@prime . .	\hyxmp@Hyp@pdfauthor <u>50</u>
	. <u>389</u> , 392, 402, 412	

\hyxmp@Hyp@pdfkeywords	\hyxmp@seed@rng	\hyxmp@xmlify
..... 67 391, 454, 464 131, 193,
\hyxmp@hypersetup .. 90	\hyxmp@seed@rng@i ..	536, 545, 571, 695
\hyxmp@InstanceID 393, 395	\hyxmp@xmp@basic@schema
..... 458, 639	\hyxmp@seed@string 643, 831
\hyxmp@iptc@extensions	. 453, 454, 459, 464	\hyxmp@xmpRights@schema
..... 715, 742	\hyxmp@set@rand@num 604, 827
\hyxmp@is@unicode 408, 416	\hyxmp@zero 354,
.... 197, 214, 229	\hyxmp@skiptorelax ..	361, 368, 374, 379
\hyxmp@legal 605 338, 344	
\hyxmp@list 572, 578, 696, 697	\hyxmp@skipzeros ... 296	I
\hyxmp@list@to@lines	\hyxmp@Space@other ..	IETF 4
. 680, 723, 730–732 305, 318	\ifhyxmp@unicodetex
\hyxmp@list@to@xml ..	\hyxmp@string 532 185, 196, 806
.... 563, 595–597	\hyxmp@string@len ..	ifxetex 10
\hyxmp@mm@schema 632, 832 907, 922	\ifxetex 201
\hyxmp@modulo@a ...	\hyxmp@sublist	Info 6
. 383, 402, 412, 418	. 150, 151, 154, 155	intcalc 10
\hyxmp@num 345	\hyxmp@temp@list ... 166	\intcalcDiv 350, 357, 364
\hyxmp@one@token ...	\hyxmp@temp@str ... 166	\intcalcMod 352, 359, 366
.... 391, 395,	\hyxmp@text 193, 271, 301, 345	IPTC . 6, 11, 26, 33, 34, 47
928, 929, 937, 938	\hyxmp@text@resource	lptc4xmpCore:CreatorContactInfo
\hyxmp@padding 485, 836 776–783, 796 3, 33, 47
\hyxmp@pdf@schema ..	\hyxmp@today 497, 597, 648–650	ISO 10
..... 509, 826	\hyxmp@toxml .. 223, 246	
\hyxmp@pdfauthor ...	\hyxmp@toxml@unicodetex	J
..... 47, 56, 595 211, 271	\jobname .. 118, 138,
\hyxmp@pdfkeywords ..	\hyxmp@trimb .. 179, 182	453, 460, 599, 857
..... 47, 73, 596	\hyxmp@trimc .. 182, 183	
\hyxmp@photometa@data	\hyxmp@trimspaces ..	K
..... 702 154, 175	Keywords 6
\hyxmp@photometa@schema	\hyxmp@try 301	\KV@Hyp@pdfauthor .. 56
..... 702, 830	\hyxmp@unicodetextfalse	\KV@Hyp@pdfkeywords 73
\hyxmp@photoshop@data 185	kvoptions 10, 13
..... 657	\hyxmp@unicodetexttrue	
\hyxmp@photoshop@schema 185	L
..... 657, 829	\hyxmp@x@default ...	\lccode 230,
\hyxmp@ProcessKeyvalOptions	. 127, 508, 550, 557	231, 320, 350,
..... 85	\hyxmp@xetex@crap ..	357, 364, 371,
\hyxmp@rand@num 202, 301	472, 476, 477, 807
. 408, 417, 455, 465	\hyxmp+xml 478,	LF 32
\hyxmp@rdf@dc 542, 592–594	485, 818, 872,	\lowercase 234,
\hyxmp@redefine@Hyp	896, 907, 914, 945	321, 351, 358,
..... 49, 87, 92	\hyxmp+xmlified ...	365, 372, 478, 808
\hyxmp@reencode ... 191 193, 538,	Lua ^A T _E X 6, 7
\hyxmp@rights 605, 608, 612, 614	553, 557, 572, 696	Lua _T E _X .. 18, 20, 40, 46

N	pdfLaTeX 3, 5	U
nativepdf (option) ... 37	pdflicenseurl (option) .	\uccode 275, 288
\newif 185 4, 10, 30, 47	Unicode 10, 18–
\next 149 , 395	pdfmark (option) 37	22, 29, 33, 36, 40, 46
ngerman 9, 46	\pdfmark .. 877, 880,	\uppercase 276, 289
\number 348, 350, 352,	884, 894, 898, 902	URL 2–4, 10, 12, 30, 31, 34
357, 359, 364, 366	pdfmetalang (option) 4, 10	\usepackage 139
P	\pdfminorversion ... 525	UTF-16BE 19
\PackageWarningNoLine	\pdfobj 869	UTF-32BE 18, 19
.... 117, 137, 855	pdfproducer (option) .. 3	UTF-8 19
PDF 1–3, 5–8,	\pdfstringdef 16, 18,	UUID 23, 25
15, 20, 25–27, 36, 39	20, 22, 24, 29, 34,	V
PDF/A 34, 47	36, 38, 40, 42, 44, 46	\vfuzz 183
pdf:Keywords 2, 27	pdfsubject (option) ..	vtexpdfmark (option) . 37
pdf:PDFVersion 2 3, 10, 29	X
pdf:Producer 2, 27	pdfTeX ... 9, 20, 37, 40	\x 301
pdfaType:prefix 47	pdftitle (option) 4, 10, 29	xdvipdfmx 7, 39
pdfauthor (option) ...	photoshop:AuthorsPosition	X _{La} TeX 6, 7
... 3, 8, 10, 12, 29 2, 32	X _{ET} TeX 10, 18,
pdfauthortitle (option)	photoshop:CaptionWriter	20, 21, 39, 40, 45, 46
..... 4, 9, 10 2, 32	XML .. 1, 2, 6, 15, 18–
pdfcaptionwriter (op-	PI 26	20, 26–29, 32, 34, 36
tion) 4, 10	\ProcessKeyvalOptions	XMP .. 1–3, 6–10, 15–
\pdfcatalog 873 85	17, 20, 23, 26, 28,
pdfcompresslevel .. 868	ps2pdf (option) 37	30, 31, 34, 37–40, 46
pdfcontactaddress (op-	Q	XMP 46
tion) 4, 6, 7	\Q 175, 184	xmp:BaseURL 2
pdfcontactcity (option) . 4	R	xmp:CreateDate 2
pdfcontactcountry (op-	rdf:li 2	xmp:CreatorTool 2
tion) 4	rdf:Seq 2	xmp:MetadataDate 2
pdfcontactemail (op-	\renewcommand 86	xmp:ModifyDate 2
tion) 4	\RequirePackage 7, 10–14	\xmpcomma 27, 30, 56 , 73 , 160
pdfcontactphone (op-	S	xmpincl 3
tion) 4	\SE->pdfdoc003 192	\xmplinesep 675 , 692, 728
pdfcontactpostcode (op-	\special 908, 916, 945, 951	xmpMM:DocumentID .
tion) 4	stringenc 10 2, 23, 31
pdfcontacturl (option) . 4	\StringEncodingConvert	xmpMM:InstanceID ..
pdfcopyright (option) 198, 2, 23, 31
.. 4, 10, 29, 30, 46	204, 215, 218, 313	\xmpquote 28, 31, 56 , 73 , 165
PDFDocEncoding ...	Subject 6	xmpRights:Marked 2, 30, 46
..... 12, 18, 19	T	xmpRights:WebStatement
pdfescape 10	TeX 18, 2, 30, 46
pdfkeywords (option) .	20, 23, 31, 39, 40, 46	\XMPTruncateList ... 166
... 3, 8, 10, 12, 29	Text 35	Y
pdflang (option)	textures (option) 37	\year 461, 497
... 3, 4, 10, 15, 29	\time 462	
\pdflastobj 873	Title 6	