

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

Scott Pakin  
scott+hyxmp@pakin.org

April 16, 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>
```

---

<sup>\*</sup>This document corresponds to `hyperxmp` v2.3a, dated 2013/04/16.

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

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

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

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

- authors (`dc:creator`)
- base URL (`xmp:BaseURL`)
- copyright (`dc:rights` and `xmpRights:Marked`)
- date (`dc:date`, `xmp:CreateDate`, `xmp:ModifyDate`, and `xmp:MetadataDate`)
- document identifier (`xmpMM:DocumentID`)
- document instance identifier (`xmpMM:InstanceID`)
- file format (`dc:format`)
- keywords (`pdf:Keywords` and `dc:subject`)
- language (`dc:language`)
- L<sup>A</sup>T<sub>E</sub>X file name (`dc:source`)
- license URL (`xmpRights:WebStatement`)
- metadata writer (`photoshop:CaptionWriter`)
- PDF-generating tool (`pdf:Producer` and `xmp:CreatorTool`)
- PDF version (`pdf:PDFVersion`)
- primary author's position/title (`photoshop:AuthorsPosition`)
- summary (`dc:description`)
- title (`dc:title`)

- contact address (`lptc4xmpCore:CreatorContactInfo/CiAdrExtadr`,  
`lptc4xmpCore:CreatorContactInfo/CiAdrCity`,  
`lptc4xmpCore:CreatorContactInfo/CiAdrRegion`,  
`lptc4xmpCore:CreatorContactInfo/CiAdrPcode`, and  
`lptc4xmpCore:CreatorContactInfo/CiAdrCtry`)
- 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 $\text{\LaTeX}$
- $\text{\LaTeX}$  + Dvipdfm
- $\text{\LaTeX}$  + Dvips + Adobe Acrobat Distiller
- X $\text{\LaTeX}$

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

Once the document is compiled, the resulting PDF file will contain an XMP packet that looks something like that shown in Appendix A. Figure 1 is a screenshot of the XMP metadata as it appears in Adobe Acrobat's "Advanced" metadata dialog box. Further clicking on the "Advanced" item within that dialog box displays all of the document's metadata sorted by schema as shown in Figure 2.

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

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

```
\XMPTruncateList{pdfauthor}
```

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

**Note 2: Acrobat multiline-field bug** The IPTC Photo Metadata schema states that "the [contact] address is a multiline field" [6]. `hyperxmp` converts commas in `pdfcontactaddress`'s argument to line breaks in the generated XML. Unfortunately, A bug in Adobe Acrobat—at least in versions 10.0.1 and earlier—causes that PDF reader to discard line breaks in the contact address. Interestingly,

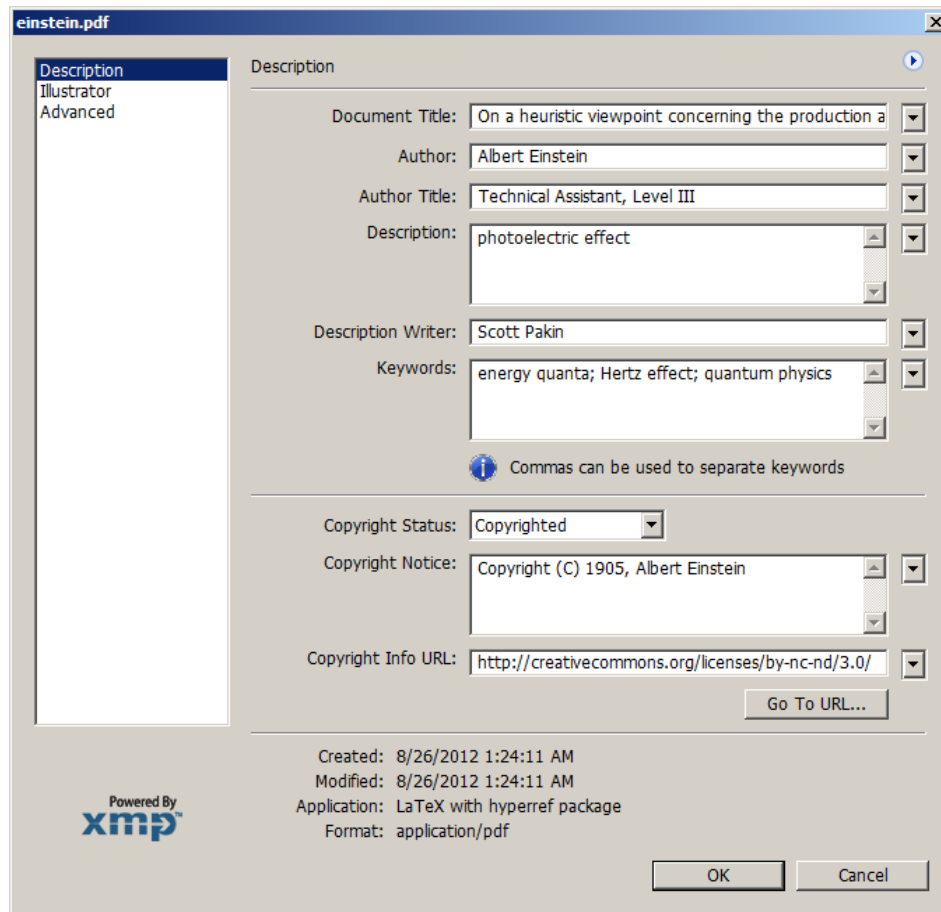


Figure 1: XMP metadata as it appears in Adobe Acrobat

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

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

**Note 3: X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X object compression** X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X (or, more precisely, the `xdvipdfmx` back end), compresses *all* PDF objects, including the ones containing XMP metadata. While Adobe Acrobat can still detect and utilize the XMP metadata, non-PDF-aware applications are unlikely to see the metadata. Three options to consider are to (1) use a different program (e.g., Lua<sup>A</sup>T<sub>E</sub>X), (2) pass the `--output-driver="xdvipdfmx -z0"` option to X<sub>q</sub>L<sup>A</sup>T<sub>E</sub>X to instruct `xdvipdfmx` to

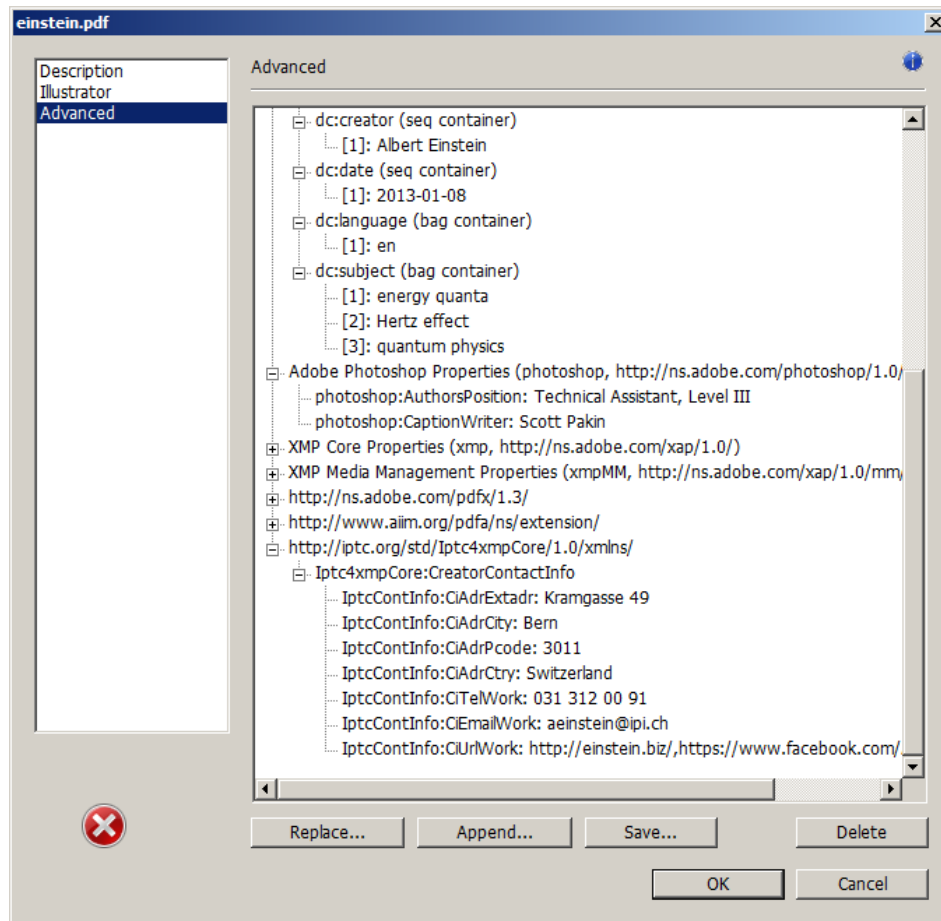


Figure 2: Additional XMP metadata as it appears in Adobe Acrobat

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

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

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

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

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



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

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

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

As of version 2.2 of `hyperxmp`, it is acceptable to use `\xmpcomma` and `\xmpquote` within any `hyperxmp` option, not just in those in which a comma normally serves as a separator (i.e., lists and multiline fields). Outside of 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<sup>A</sup>T<sub>E</sub>X source code for `hyperxmp`. Read this section only if you want to learn how `hyperxmp` is implemented.

#### 3.1 Initial preparation

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

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

`\hyxmp@at@end`    The `\hyxmp@at@end` macro includes code at the end of the document. For pdfT<sub>E</sub>X, the standard `\AtEndDocument` works well enough. For all the other backends we use `\AtEndDvi` from the `atenddvi` package, which is more robust but requires an addition L<sup>A</sup>T<sub>E</sub>X run.

```

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

```

### 3.2 Integration with hyperref

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

We load a bunch of helper packages: kvoptions for package-option processing, pdfescape and stringenc for re-encoding Unicode strings, intcalc for performing integer calculations (division and modulo), and ifxetex for detecting X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X.

```

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

\@pdfcopyright Prepare to store the document's copyright statement.
15 \def\@pdfcopyright{}
16 \define@key{Hyp}{pdfcopyright}{\pdfstringdef\@pdfcopyright{#1}}

\@pdflicenseurl Prepare to store the URL containing the document's license agreement.
17 \def\@pdflicenseurl{}
18 \define@key{Hyp}{pdflicenseurl}{\pdfstringdef\@pdflicenseurl{#1}}

\@pdfauthortitle Prepare to store the author's position/title (e.g., Staff Writer).
19 \def\@pdfauthortitle{}
20 \define@key{Hyp}{pdfauthortitle}{\pdfstringdef\@pdfauthortitle{#1}}

\@pdfcaptionwriter Prepare to store the name of the person who inserted the hyperxmp metadata.
21 \def\@pdfcaptionwriter{}
22 \define@key{Hyp}{pdfcaptionwriter}{\pdfstringdef\@pdfcaptionwriter{#1}}

\@pdfmetalang Prepare to store the natural language of the document's metadata, typically as an
ISO 639-1 two-letter abbreviation.
23 \def\@pdfmetalang{}
24 \define@key{Hyp}{pdfmetalang}{\pdfstringdef\@pdfmetalang{#1}}

```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
56 \define@key{Hyp}{pdfauthor}{}%
57 \let\xmpcomma=\hyxmp@comma
```

```

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

```

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

```

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

```

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

```

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

```

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

```

85 \let\hyxmp@ProcessKeyvalOptions=\ProcessKeyvalOptions
86 \renewcommand*{\ProcessKeyvalOptions}{%
87   \hyxmp@redefine@Hyp
88   \hyxmp@ProcessKeyvalOptions
89 }

```

```

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

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

```

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

```

124 \AtBeginDocument{%
125   \ifpackageloaded{hyperref}{%

```

In older versions of `hyperref`, `\@pdflang` is set to `\@empty` if `pdflang` is not specified. In newer versions of `hyperref`, `\@pdflang` is set to `\relax` if `pdflang` is not specified.

The latter is a bit problematic for `hyperxmp` because it makes `\@pdflang` non-expandable, which causes a literal “`\@pdflang`” to be written as XMP metadata. To avoid that situation we redefine `\@pdflang` as `\@empty` if we see it set to `\relax`.

```
126 \ifx\@pdflang\relax
127 \let\@pdflang=\@empty
128 \fi
```

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

```
129 \ifx\@pdflang\@empty
130 \let\@pdfmetalang=\hyxmp@x@default
131 \else
132 \edef\@pdfmetalang{\@pdflang}%
133 \fi
134 \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`.

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

### 3.3 Manipulating author-supplied data

The author provides metadata information to `hyperxmp` via package options to `hyperref` or via `hyperref`’s `\hypersetup` command. The functions in this section convert author-supplied lists (e.g., `pdfkeywords={foo, bar, baz}`) into L<sup>A</sup>T<sub>E</sub>X lists (e.g., `\@elt {foo} \@elt {bar} \@elt {baz}`) that can be more easily manipulated (Section 3.3.1); trim spaces off the ends of strings (Section 3.3.2); and, in Section 3.3.3, convert text to XML (e.g., from `<scott+hyxmp@pakin.org>` to `&lt;scott+hyxmp@pakin.org&gt;`).

#### 3.3.1 List manipulation

We define a macro for converting a list of comma-separated elements (e.g., the list of PDF keywords) to a list of L<sup>A</sup>T<sub>E</sub>X `\@elt`-separated elements.

`\hyxmp@commas@to@list` Given a macro name (#1) and a comma-separated list (#2), define the macro name as the elements of the list, each preceded by `\@elt`. (Executing the macro therefore applies `\@elt` to each element in turn.)

```

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

```

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

```

\next 152 \def\hyxmp@commas@to@list@i#1#2,{%
153   \gdef\hyxmp@sublist{#2}%
154   \ifx\hyxmp@sublist\@empty
155     \let\next=\relax
156   \else
157     \hyxmp@trimspaces\hyxmp@sublist
158     \@cons{#1}{\hyxmp@sublist}%
159     \def\next{\hyxmp@commas@to@list@i{#1}}%
160   \fi
161   \next
162 }

```

`\xmpcomma` Because `hyperxmp` splits lists at commas, a comma cannot normally be used within a list. We there provide an `\xmpcomma` macro that can expand to either a true comma or a placeholder character depending on the situation. Here, we bind it to a comma so it can be used in *any* `hyperxmp` option, not just those that treat commas specially.

```

163 \def\xmpcomma{,}%

```

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

```

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

```

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

```

168 \let\xmpquote=\relax

```

`\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

```

\@elt

```



Adobe Acrobat display the author list correctly. It's sad that this is necessary, though.

```

169 \newcommand{\XMPTruncateList}[1]{%
170   \edef\hyxmp@temp@str{\csname hyxmp@#1\endcsname}%
171   \hyxmp@commas@to@list{\hyxmp@temp@list}{\hyxmp@temp@str}%
172   \def\@elt##1{%
173     \expandafter\gdef\csname @#1\endcsname{##1}%
174     \let\@elt=\@gobble
175   }
176   \hyxmp@temp@list
177 }

```

### 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.

```

178 \catcode'\Q=3
\hyxmp@trimspaces\x redefines \x to have the same replacement text sans leading
and trailing space tokens.
179 \newcommand{\hyxmp@trimspaces}[1]{%
  Use grouping to emulate a multi-token afterassignment queue.
180   \begingroup
    Put “\toks 0 {” into the afterassignment queue.
181   \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.
182   \expandafter\hyxmp@trimb\expandafter\noexpand#1Q Q}%
    Transfer the trimmed text back into #1.
183   \edef#1{\the\toks0}%
184 }

```

`\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.

```

185 \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.

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

### 3.3.3 Converting text to XML

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

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

```
188 \newif\ifhyxmp@unicodetex
189 \ifnum64='^^^^0040\relax
190   \hyxmp@unicodetextrue
191 \else
192   \hyxmp@unicodetexfalse
193 \fi
```

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

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

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

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

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

```
196 \newcommand*{\hyxmp@xmlify}[1]{%
197   \gdef\hyxmp@xmlified{}
```

Escaped PDF string  $\rightarrow$  PDFDocEncoding/Unicode

```
198   \EdefUnescapeString\hyxmp@text{#1}%
199   \ifhyxmp@unicodetex
```

PDFDocEncoding/Unicode  $\rightarrow$  UTF-32BE

```
200     \hyxmp@is@unicode\hyxmp@text{%
201       \StringEncodingConvert
202       \hyxmp@text\hyxmp@text{utf16be}{utf32be}%
```

```

203   }{%
204     \ifxetex
205       \hyxmp@xetex@crap
206     \else
207       \StringEncodingConvert
208       \hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
209     \fi
210   }%

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

UTF-32BE → XML in ASCII
212   \edef\hyxmp@text{%
213     \expandafter
214     }\expandafter\hyxmp@toxml@unicodetex\hyxmp@text
215     \relax\relax\relax\relax\relax\relax\relax\relax
216   \else

PDFDocEncoding/Unicode → UTF-8
217   \hyxmp@is@unicode\hyxmp@text{%
218     \StringEncodingConvert
219     \hyxmp@text\hyxmp@text{utf16be}{utf8}%
220   }{%
221     \StringEncodingConvert
222     \hyxmp@text\hyxmp@text{pdfdoc}{utf8}%
223   }%

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

UTF-8 as hex string → XML in UTF-8 as hex string
225   \edef\hyxmp@text{%
226     \expandafter\hyxmp@toxml\hyxmp@text\@empty\@empty
227   }%

XML in UTF-8 as hex string → XML in UTF-8
228   \EdefUnescapeHex\hyxmp@text{\hyxmp@text}%
229   \fi
230   \global\let\hyxmp@xmlified\hyxmp@text
231 }

```

`\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.

```

232 \begingroup
233   \lccode'\<=254 %
234   \lccode'\>=255 %
235   \catcode254=12 %
236   \catcode255=12 %
237 \lowercase{\endgroup
238   \def\hyxmp@is@unicode#1{%
239     \expandafter\hyxmp@@is@unicode#1<>\@nil

```

```

240 }%
241 \def\hyxmp@is@unicode#1<>#2\@nil{%
242   \ifx\#1\%
243     \expandafter\@firstoftwo
244   \else
245     \expandafter\@secondoftwo
246   \fi
247 }%
248 }

```

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

```

249 \def\hyxmp@toxml#1#2{%
250   \ifx#1\@empty
251   \else
252     \ifnum"#1#2='\& %
253       26616D703B% &
254     \else\ifnum"#1#2='\< %
255       266C743B% &lt;
256     \else\ifnum"#1#2='\> %
257       2667743B% &gt;
258   \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.

```

259   \@ifundefined{pdfmark}{%
260     #1#2%
261   }{%
262     \ifnum"#1#2='\( %
263       5C28% \(
264     \else\ifnum"#1#2='\) %
265       5C29% \)
266     \else
267       #1#2%
268     \fi\fi
269   }%
270   \fi\fi\fi
271   \expandafter\hyxmp@toxml
272 \fi
273 }

```

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

```

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

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

```

299 \def\hyxmp@skipzeros#1{%
300   \ifx#10%
301     \expandafter\hyxmp@skipzeros
302   \fi
303 }
```

`\x` In the case of  $X_{\text{L}}\TeX$ , the strings defined by `\pdfstringdef` can contain big

`\hyxmp@xetex@crap` characters. In this case, the string is treated as Unicode.

```

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

```

316      \StringEncodingConvert\hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
317      \fi
318    }%
319  }
320  \x{ }

```

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

```

321 \begingroup
322   \catcode'\~=12 %
323   \lccode'\~=\' %
324 \lowercase{\endgroup
325   \def\hyxmp@SpaceOther#1 #2\@nil{%
326     #1%
327     \ifx\relax#2\relax
328       \expandafter\@gobble
329     \else
330       ~%
331       \expandafter\@firstofone
332     \fi
333     {\hyxmp@SpaceOther#2\@nil}%
334   }%
335 }

```

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

```

336 \def\hyxmp@crap@test#1{%
337   \ifx#1\relax
338   \else
339     \ifnum'#1>127 %
340       \let\hyxmp@crap@result=Y%
341       \expandafter\expandafter\expandafter\hyxmp@skiptorelax
342     \else
343       \expandafter\expandafter\expandafter\hyxmp@crap@test
344     \fi
345   \fi
346 }

```

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

```

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

```

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

```

\hyxmp@num 348 \def\hyxmp@crap@convert#1{%
\hyxmp@text 349   \ifx#1\relax
350   \else
351     \edef\hyxmp@num{\number'#1}%
352     \ifnum\hyxmp@num>"FFFFFF %
353       \lccode'\!=\intcalcDiv{\hyxmp@num}{\number"1000000}\relax
354       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
355     \edef\hyxmp@num{\intcalcMod{\hyxmp@num}{\number"1000000}}%
356   \else

```

```

357     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
358   \fi
359   \ifnum\hyxmp@num>"FFFF %
360     \lccode'\!=\intcalDiv{\hyxmp@num}{\number"10000}\relax
361     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
362     \edef\hyxmp@num{\intcalMod{\hyxmp@num}{\number"10000}}%
363   \else
364     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
365   \fi
366   \ifnum\hyxmp@num>"FF %
367     \lccode'\!=\intcalDiv{\hyxmp@num}{\number"100}\relax
368     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
369     \edef\hyxmp@num{\intcalMod{\hyxmp@num}{\number"100}}%
370   \else
371     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
372   \fi
373   \ifnum\hyxmp@num>0 %
374     \lccode'\!=\hyxmp@num\relax
375     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
376   \else
377     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
378   \fi
379   \expandafter\hyxmp@crap@convert
380 \fi
381 }

```

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

```

382 \begingroup
383   \catcode0=12 %
384   \gdef\hyxmp@zero{^^00}%
385 \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  $\text{\TeX}$  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.

```

386 \def\hyxmp@modulo@a#1{%
387   \@tempcntb=\@tempcnta
388   \divide\@tempcntb by #1
389   \multiply\@tempcntb by #1
390   \advance\@tempcnta by -\@tempcntb
391 }

```

`\hyxmp@big@prime` Define a couple of large prime numbers that can still be stored in a  $\text{\TeX}$  counter.  
`\hyxmp@big@prime@ii` 392 \def\hyxmp@big@prime{536870923}

```

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

\hyxmp@seed@rng Seed hyperxmp's random-number generator from a given piece of text.
\hyxmp@one@token 394 \def\hyxmp@seed@rng#1{%
395   \@tempcnta=\hyxmp@big@prime
396   \futurelet\hyxmp@one@token\hyxmp@seed@rng@i#1\@empty
397 }

\hyxmp@seed@rng@i Do all of the work for \hyxmp@seed@rng. For each character code  $c$  of the input
\hyxmp@one@token text, assign  $\text{\@tempcnta} \leftarrow 3 \cdot \text{\@tempcnta} + c \pmod{\text{\hyxmp@big@prime}}$ .
\next 398 \def\hyxmp@seed@rng@i{%
399   \ifx\hyxmp@one@token\@empty
400     \let\next=\relax
401   \else
402     \def\next##1{%
403       \multiply\@tempcnta by 3
404       \advance\@tempcnta by '##1
405       \hyxmp@modulo@a{\hyxmp@big@prime}%
406       \futurelet\hyxmp@one@token\hyxmp@seed@rng@i
407     }%
408   \fi
409 \next
410 }

\hyxmp@set@rand@num Advance \hyxmp@rand@num to the next pseudorandom number in the se-
\hyxmp@rand@num quence. Specifically, we assign  $\text{\hyxmp@rand@num} \leftarrow 3 \cdot \text{\hyxmp@rand@num} + \text{\hyxmp@big@prime@ii} \pmod{\text{\hyxmp@big@prime}}$ . Note that both \@tempcnta
and \@tempcntb are overwritten in the process.
411 \def\hyxmp@set@rand@num{%
412   \@tempcnta=\hyxmp@rand@num
413   \multiply\@tempcnta by 3
414   \advance\@tempcnta by \hyxmp@big@prime@ii
415   \hyxmp@modulo@a{\hyxmp@big@prime}%
416   \xdef\hyxmp@rand@num{\the\@tempcnta}%
417 }

\hyxmp@append@hex Append a randomly selected hexadecimal digit to macro #1. Note that both
\@tempcnta and \@tempcntb are overwritten in the process.
418 \def\hyxmp@append@hex#1{%
419   \hyxmp@set@rand@num
420   \@tempcnta=\hyxmp@rand@num
421   \hyxmp@modulo@a{16}%
422   \ifnum\@tempcnta<10
423     \xdef#1{#1\the\@tempcnta}%
424   \else
425     \advance\@tempcnta by -10
426     \ifcase\@tempcnta

```

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



```

427     \xdef#1{#1a}%
428     \or\xdef#1{#1b}%
429     \or\xdef#1{#1c}%
430     \or\xdef#1{#1d}%
431     \or\xdef#1{#1e}%
432     \or\xdef#1{#1f}%
433   \fi
434 \fi
435 }

```

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

```

436 \def\hyxmp@append@hex@iv#1{%
437   \hyxmp@append@hex#1%
438   \hyxmp@append@hex#1%
439   \hyxmp@append@hex#1%
440   \hyxmp@append@hex#1%
441 }

```

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

```

442 \def\hyxmp@create@uuid#1{%
443   \def#1{uuid:}%
444   \hyxmp@append@hex@iv#1%
445   \hyxmp@append@hex@iv#1%
446   \g@addto@macro#1{-}%
447   \hyxmp@append@hex@iv#1%
448   \g@addto@macro#1{-}%
449   \hyxmp@append@hex@iv#1%
450   \g@addto@macro#1{-}%
451   \hyxmp@append@hex@iv#1%
452   \hyxmp@append@hex@iv#1%
453   \hyxmp@append@hex@iv#1%
454 }

```

`\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.

```

455 \newcommand*{\hyxmp@def@DocumentID}{%
456   \edef\hyxmp@seed@string{\jobname:\@pdftitle:\@pdfauthor}%
457   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
458   \edef\hyxmp@rand@num{\the\@tempcnta}%
459   \hyxmp@create@uuid\hyxmp@DocumentID
460 }

```

`\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.

```

461 \newcommand*{\hyxmp@def@InstanceID}{%
462   \edef\hyxmp@seed@string{%
463     \jobname:\@pdftitle:\@pdfauthor:%
464     \the\year/\the\month/\the\day:%
465     \the\time
466   }%
467   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
468   \edef\hyxmp@rand@num{\the\@tempcnta}%
469   \hyxmp@create@uuid\hyxmp@InstanceID
470 }

```

### 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.

```

471 \newcommand*{\hyxmp@add@to+xml}[1]{%
472   \bgroup
473   \@tempcnta=0
474   \loop
475     \lccode\@tempcnta=\@tempcnta
476     \advance\@tempcnta by 1
477     \ifnum\@tempcnta<256
478       \repeat
479       \lccode'\_='\ \relax
480       \lccode'\^C='\,\relax
481       \lowercase{\xdef\hyxmp+xml{\hyxmp+xml#1}}%
482   \egroup
483 }

```

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

```

484 \bgroup
485 \catcode'\#=11

```

```

486 \gdef\hyxmp@hash{#}
487 \egroup

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

488 \bgroup
489 \xdef\hyxmp@xml{%
490 \hyxmp@add@to@xml{%
491 ----- ^^J%
492 }
493 \xdef\hyxmp@padding{\hyxmp@xml}%
494 \egroup
495 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
496 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
497 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
498 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
499 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}

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

500 \xdef\hyxmp@today{\the\year}%
501 \ifnum\month<10
502 \xdef\hyxmp@today{\hyxmp@today-0\the\month}%
503 \else
504 \xdef\hyxmp@today{\hyxmp@today-\the\month}%
505 \fi
506 \ifnum\day<10
507 \xdef\hyxmp@today{\hyxmp@today-0\the\day}%
508 \else
509 \xdef\hyxmp@today{\hyxmp@today-\the\day}%
510 \fi

\hyxmp@x@default  Define an x-default string that we can use in comparisons with \@pdfmetalang.
511 \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.
512 \newcommand*{\hyxmp@pdf@schema}{%

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

513 \let\hyxmp@have@any=!%
514 \ifx\@pdfkeywords\@empty
515 \ifx\@pdfproducer\@empty
516 \let\hyxmp@have@any=\@empty
517 \fi
518 \fi

```

```

519 \ifx\hyxmp@have@any\@empty
520 \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).
521 \hyxmp@add@to@xml{%
522 -----<rdf:Description rdf:about=""^^J%
523 -----xmlns:pdf="http://ns.adobe.com/pdf/1.3/">^^J%
524 }%
525 \hyxmp@add@simple{pdf:Keywords}{\@pdfkeywords}%
526 \hyxmp@add@simple{pdf:Producer}{\@pdfproducer}%
527 \@ifundefined{pdfminorversion}{}{%
528 \hyxmp@add@simple{pdf:PDFVersion}{1.\the\pdfminorversion}%
529 }%
530 \hyxmp@add@to@xml{%
531 -----</rdf:Description>^^J%
532 }%
533 \fi
534 }

```

**\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.

```

535 \newcommand*{\hyxmp@add@simple}[2]{%
536 \edef\hyxmp@string{#2}%
537 \ifx\hyxmp@string\@empty
538 \else
539 \hyxmp@xmlify{\hyxmp@string}%
540 \hyxmp@add@to@xml{%
541 -----<#1>\hyxmp@xmlified</#1>^^J%
542 }%
543 \fi
544 }

```

### 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.

```

545 \newcommand*{\hyxmp@rdf@dc}[2]{%
546 \ifx#2\@empty
547 \else
548 \hyxmp@xmlify{#2}%
549 \hyxmp@add@to@xml{%
550 -----<dc:#1>^^J%
551 -----<rdf:Alt>^^J%
552 }%
553 \ifx\@pdfmetalang\hyxmp@x@default

```

```

554     \else
555     \hyxmp@add@to@xml{%
556 -----<rdf:li xml:lang="\@pdfmetalang">\hyxmp@xmlified</rdf:li>^^J%
557     }%
558     \fi
559     \hyxmp@add@to@xml{%
560 -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@xmlified</rdf:li>^^J%
561 -----</rdf:Alt>^^J%
562 -----</dc:#1>^^J%
563     }%
564     \fi%
565 }%

```

`\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.

```

566 \newcommand*{\hyxmp@list@to@xml}[3]{%
567     \ifx#3\@empty
568     \else
569     \hyxmp@add@to@xml{%
570 -----<dc:#1>^^J%
571 -----<rdf:#2>^^J%
572     }%
573     \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`.

```

574     \hyxmp@xmlify{#3}%
575     \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
576     \def\@elt##1{%
577     \hyxmp@add@to@xml{%
578 -----<rdf:li>##1</rdf:li>^^J%
579     }%
580     }%
581     \hyxmp@list
582     \egroup
583     \hyxmp@add@to@xml{%
584 -----</rdf:#2>^^J%
585 -----</dc:#1>^^J%
586     }%
587     \fi
588 }

```

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

also specify the `dc:date` property using the date the document was run through L<sup>A</sup>T<sub>E</sub>X and the `dc:source` property using the base name of the source file with `.tex` appended.

```

589 \newcommand*{\hyxmp@dc@schema}{%
590   \hyxmp@add@to@xml{%
591     <rdf:Description rdf:about=""^^J%
592     _____xmlns:dc="http://purl.org/dc/elements/1.1/">^^J%
593     _____<dc:format>application/pdf</dc:format>^^J%
594   }%
595   \hyxmp@rdf@dc{title}{\@pdftitle}%
596   \hyxmp@rdf@dc{description}{\@pdfsubject}%
597   \hyxmp@rdf@dc{rights}{\@pdfcopyright}%
598   \hyxmp@list@to@xml{creator}{Seq}{\hyxmp@pdfauthor}%
599   \hyxmp@list@to@xml{subject}{Bag}{\hyxmp@pdfkeywords}%
600   \hyxmp@list@to@xml{date}{Seq}{\hyxmp@today}%
601   \hyxmp@add@simple{dc:language}{\@pdflang}%
602   \hyxmp@add@simple{dc:source}{\jobname.tex}%
603   \hyxmp@add@to@xml{%
604     _____</rdf:Description>^^J%
605   }%
606 }
```

### 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.

```

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

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

```

608   \let\hyxmp@rights=\@empty
609   \ifx\@pdflicenseurl\@empty
610   \else
611     \def\hyxmp@rights{YES}%
612   \fi
613   \ifx\@pdfcopyright\@empty
614   \else
615     \def\hyxmp@rights{YES}%
616   \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.

```

617   \ifx\hyxmp@rights\@empty
618   \else
```

Header

```

619     \hyxmp@add@to@xml{%
```

```

620 -----<rdf:Description rdf:about=""^^J%
621 -----xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/">^^J%
622  }%
    Copyright indication
623   \ifx\@pdfcopyright\@empty
624   \else
625     \hyxmp@add@to@xml{%
626 -----<xmpRights:Marked>True</xmpRights:Marked>^^J%
627   }%
628   \fi
    License URL
629   \hyxmp@add@simple{xmpRights:WebStatement}{\@pdflicenseurl}%
    Trailer
630   \hyxmp@add@to@xml{%
631 -----</rdf:Description>^^J%
632   }%
633   \fi
634 }

```

### 3.5.5 The XMP Media Management schema

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

```

635 \gdef\hyxmp@mm@schema{%
636   \hyxmp@def@DocumentID
637   \hyxmp@def@InstanceID
638   \hyxmp@add@to@xml{%
639 -----<rdf:Description rdf:about=""^^J%
640 -----xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/">^^J%
641 -----<xmpMM:DocumentID>\hyxmp@DocumentID</xmpMM:DocumentID>^^J%
642 -----<xmpMM:InstanceID>\hyxmp@InstanceID</xmpMM:InstanceID>^^J%
643 -----</rdf:Description>^^J%
644   }%
645 }

```

### 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`.

```

646 \newcommand*{\hyxmp@xmp@basic@schema}{%
647   \hyxmp@add@to@xml{%
648 -----<rdf:Description rdf:about=""^^J%
649 -----xmlns:xmp="http://ns.adobe.com/xap/1.0/">^^J%

```

```

650 }%
651 \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@today}%
652 \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@today}%
653 \hyxmp@add@simple{xmp:MetadataDate}{\hyxmp@today}%
654 \hyxmp@add@simple{xmp:CreatorTool}{\pdfcreator}%
655 \hyxmp@add@simple{xmp:BaseURL}{\@baseurl}%
656 \hyxmp@add@to@xml{%
657 -----</rdf:Description>^^J%
658 }%
659 }

```

### 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.

```

660 \gdef\hyxmp@photoshop@schema{%
661 \edef\hyxmp@photoshop@data{\@pdfauthortitle\@pdfcaptionwriter}%
662 \ifx\hyxmp@photoshop@data\@empty
663 \else
664 \hyxmp@add@to@xml{%
665 -----<rdf:Description rdf:about=""^^J%
666 -----_xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/">^^J%
667 }%
668 \fi
669 \hyxmp@add@simple{photoshop:AuthorsPosition}{\@pdfauthortitle}%
670 \hyxmp@add@simple{photoshop:CaptionWriter}{\@pdfcaptionwriter}%
671 \ifx\hyxmp@photoshop@data\@empty
672 \else
673 \hyxmp@add@to@xml{%
674 -----</rdf:Description>^^J%
675 }%
676 \fi
677 }

```

### 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.

```

678 \begingroup
679 \catcode'\&=12
680 \catcode'\#=12
681 \gdef\xmplinesep{&#xA;}
682 \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.

```

683 \newcommand*{\hyxmp@list@to@lines}[2]{%

```



```

684 \ifx#2\@empty
685 \else
686 \bgroup
687 \hyxmp@add@to@xml{%
688 -----<#1>%
689 }%

```

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

```

690 \def\@elt@first##1{%
691 \hyxmp@add@to@xml{##1}%
692 \let\@elt=\@elt@rest
693 }%

```

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

```

694 \def\@elt@rest##1{%
695 \hyxmp@add@to@xml{\xmplinesep##1}%
696 }%

```

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

```

697 \let\@elt=\@elt@first
698 \hyxmp@xmllify{#2}%
699 \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmllified}%
700 \hyxmp@list
701 \hyxmp@add@to@xml{</#1>^^J}%
702 \egroup
703 \fi
704 }

```

`\hyxmp@photometa@schema` Add properties defined by the IPTC Photo Metadata schema [6] to the `\hyxmp@xml` macro. We currently support only the contact-information details structure, viz. the `lptc4xmpCore:CreatorContactInfo/CiAdrExtadr`, `lptc4xmpCore:CreatorContactInfo/CiAdrCity`, `lptc4xmpCore:CreatorContactInfo/CiAdrRegion`, `lptc4xmpCore:CreatorContactInfo/CiAdrPcode`, `lptc4xmpCore:CreatorContactInfo/CiAdrCtry`, `lptc4xmpCore:CreatorContactInfo/CiTelWork`, `lptc4xmpCore:CreatorContactInfo/CiEmailWork`, and `lptc4xmpCore:CreatorContactInfo/CiUrlWork` properties.

```

705 \gdef\hyxmp@photometa@schema{%
706 \edef\hyxmp@photometa@data{%
707 \@pdfcontactaddress
708 \@pdfcontactcity
709 \@pdfcontactregion
710 \@pdfcontactpostcode
711 \@pdfcontactcountry
712 \@pdfcontactphone
713 \@pdfcontactemail
714 \@pdfcontacturl

```

```

715 }%
716 \ifx\hyxmp@photometa@data\@empty
717 \else
718 \hyxmp@iptc@extensions
719 \hyxmp@add@to@xml{%
720 -----<rdf:Description rdf:about=""^^J%
721 -----xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"^^J%
722 -----xmlns:IptcContInfo="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/contactinfo/">^^J%
723 -----<Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">^^J%
724 }%
725 \fi
726 \hyxmp@list@to@lines{IptcContInfo: CiAdrExtadr}{\@pdfcontactaddress}%
727 \hyxmp@add@simple{IptcContInfo: CiAdrCity}{\@pdfcontactcity}%
728 \hyxmp@add@simple{IptcContInfo: CiAdrRegion}{\@pdfcontactregion}%
729 \hyxmp@add@simple{IptcContInfo: CiAdrPcode}{\@pdfcontactpostcode}%
730 \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.

```

731 \bgroup
732 \def\xmplinesep{,%
733 \hyxmp@list@to@lines{IptcContInfo: CiTelWork}{\@pdfcontactphone}%
734 \hyxmp@list@to@lines{IptcContInfo: CiEmailWork}{\@pdfcontactemail}%
735 \hyxmp@list@to@lines{IptcContInfo: CiUrlWork}{\@pdfcontacturl}%
736 \egroup
737 \ifx\hyxmp@photometa@data\@empty
738 \else
739 \hyxmp@add@to@xml{%
740 -----</Iptc4xmpCore:CreatorContactInfo>^^J%
741 -----</rdf:Description>^^J%
742 }%
743 \fi
744 }

```

`\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.

```

745 \newcommand*{\hyxmp@iptc@extensions}{%
746 \hyxmp@add@to@xml{%
747 -----<rdf:Description rdf:about=""^^J%

```

```

748 _____xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"^^J%
749 _____xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema\hyxmp@hash"^^J%
750 _____xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property\hyxmp@hash"^^J%
751 _____xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type\hyxmp@hash"^^J%
752 _____xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field\hyxmp@hash">^^J%
753 _____<pdfaExtension:schemas>^^J%
754 _____<rdf:Bag>^^J%
755 _____<rdf:li rdf:parseType="Resource">^^J%
756 _____<pdfaSchema:schema>IPTC Core Schema</pdfaSchema:schema>^^J%
757 _____<pdfaSchema:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaSchema:
758 _____<pdfaSchema:prefix>Iptc4xmpCore</pdfaSchema:prefix>^^J%
759 _____<pdfaSchema:property>^^J%
760 _____<rdf:Seq>^^J%
761 _____<rdf:li rdf:parseType="Resource">^^J%
762 _____<pdfaProperty:name>CreatorContactInfo</pdfaProperty:name>^^J%
763 _____<pdfaProperty:valueType>contactinfo</pdfaProperty:valueType>^^J%
764 _____<pdfaProperty:category>external</pdfaProperty:category>^^J%
765 _____<pdfaProperty:description>contact information for the document's creator</p
766 _____</rdf:li>^^J%
767 _____</rdf:Seq>^^J%
768 _____</pdfaSchema:property>^^J%
769 _____<pdfaSchema:valueType>^^J%
770 _____<rdf:Seq>^^J%
771 _____<rdf:li rdf:parseType="Resource">^^J%
772 _____<pdfaType:type>contactinfo</pdfaType:type>^^J%
773 _____<pdfaType:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/contactin
774 _____<pdfaType:prefix>IptcContInfo</pdfaType:prefix>^^J%
775 _____<pdfaType:description>contact information</pdfaType:description>^^J%
776 _____<pdfaType:field>^^J%
777 _____<rdf:Seq>^^J%
778 }%

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

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

```

```

797 }%
798 }

```

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

```

799 \newcommand*{\hyxmp@text@resource}[2]{%
800   \hyxmp@add@to@xml{%
801     -----<rdf:li rdf:parseType="Resource">^^J%
802     -----<pdfaField:name>#1</pdfaField:name>^^J%
803     -----<pdfaField:valueType>Text</pdfaField:valueType>^^J%
804     -----<pdfaField:description>#2</pdfaField:description>^^J%
805     -----</rdf:li>^^J%
806   }
807 }

```

### 3.5.9 Constructing the XMP packet

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

```

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

```

`\hyxmp@construct@packet` Successively add XML data to `\hyxmp@xml` until we have something we can insert into the document's PDF catalog.

`\hyxmp@xml`

```

821 \def\hyxmp@construct@packet{%
822   \gdef\hyxmp@xml{%
823     \hyxmp@add@to@xml{<?xpacket begin="\hyxmp@bom" %
824 id="W5M0MpCehiHzreSzNTczkc9d"?>^^J%
825 <x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="3.1-702">^^J%
826 ___<rdf:RDF
827 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\hyxmp@hash">^^J%
828   }%
829   \hyxmp@pdf@schema
830   \hyxmp@xmpRights@schema
831   \hyxmp@dc@schema
832   \hyxmp@photoshop@schema
833   \hyxmp@photometa@schema
834   \hyxmp@xmp@basic@schema
835   \hyxmp@mm@schema
836   \hyxmp@add@to@xml{%

```

```

837 ___</rdf:RDF>^^J%
838 </x:xmpmeta>^^J%
839 \hyxmp@padding
840 <?xpacket end="w"?>^^J%
841 }%
842 }

```

### 3.6 Embedding the XMP packet

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

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

```

843 \newcommand*{\hyxmp@embed@packet}{%
844   \hyxmp@construct@packet
845   \def\hyxmp@driver{hpdfTEX}%
846   \ifx\hyxmp@driver\Hy@driver
847     \hyxmp@embed@packet@pdfTEX
848   \else
849     \def\hyxmp@driver{hdvipdfm}%
850     \ifx\hyxmp@driver\Hy@driver
851       \hyxmp@embed@packet@dvipdfm
852     \else
853       \def\hyxmp@driver{hXeTeX}%
854       \ifx\hyxmp@driver\Hy@driver
855         \hyxmp@embed@packet@XeTeX
856       \else
857         \ifundefined{pdfmark}{%
858           \PackageWarningNoLine{hyperxmp}{%
859             Unrecognized hyperref driver ‘\Hy@driver’. \MessageBreak
860             \jobname.tex’s XMP metadata will *not* be \MessageBreak
861             embedded in the resulting file}%
862         }{%
863           \hyxmp@embed@packet@pdfmark
864         }%
865       \fi
866     \fi
867   \fi
868 }

```

#### 3.6.1 Embedding using pdfTeX

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

```

869 \newcommand*{\hyxmp@embed@packet@pdfTEX}{%
870   \bgroup
871   \pdfcompresslevel=0
872   \immediate\pdfobj stream attr {%

```

```

873      /Type /Metadata
874      /Subtype /XML
875    }{\hyxmp@xml}%
876    \pdfcatalog {/Metadata \the\pdflastobj\space 0 R}%
877  \egroup
878 }

```

### 3.6.2 Embedding using any pdfmark-based backend

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

```

879 \newcommand*{\hyxmp@embed@packet@pdfmark}{%
880   \pdfmark{%
881     pdfmark=/NamespacePush
882   }%
883   \pdfmark{%
884     pdfmark=/OBJ,
885     Raw={/_objdef \string{hyxmp@Metadata\string} /type /stream}%
886   }%
887   \pdfmark{%
888     pdfmark=/PUT,
889     Raw={\string{hyxmp@Metadata\string}
890       2 dict begin
891         /Type /Metadata def
892         /Subtype /XML def
893         currentdict
894       end
895     }%
896   }%
897   \pdfmark{%
898     pdfmark=/PUT,
899     Raw={\string{hyxmp@Metadata\string} (\hyxmp@xml)}%
900   }%
901   \pdfmark{%
902     pdfmark=/Metadata,
903     Raw={\string{Catalog\string} \string{hyxmp@Metadata\string}}%
904   }%
905   \pdfmark{%
906     pdfmark=/NamespacePop
907   }%
908 }

```

### 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.

```

909 \newcommand*{\hyxmp@embed@packet@dvipdfm}{%

```

```

910 \hyxmp@string@len{\hyxmp@xml}%
911 \special{pdf: object @hyxmp@Metadata
912   <<
913     /Type /Metadata
914     /Subtype /XML
915     /Length \the\@tempcnta
916   >>
917   stream^^J\hyxmp@xml endstream%
918 }%
919 \special{pdf: docview
920   <<
921     /Metadata @hyxmp@Metadata
922   >>
923 }%
924 }

```

`\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.

```

925 \newcommand*{\hyxmp@string@len}[1]{%
926   \@tempcnta=0
927   \expandafter\hyxmp@count@spaces#1 {} %
928   \expandafter\hyxmp@count@non@spaces#1{}%
929 }

```

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

```

930 \def\hyxmp@count@spaces#1 {%
931   \def\hyxmp@one@token{#1}%
932   \ifx\hyxmp@one@token\empty
933     \advance\@tempcnta by -1
934   \else
935     \advance\@tempcnta by 1
936     \expandafter\hyxmp@count@spaces
937   \fi
938 }

```

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

```

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

```

```

945 \fi
946 }

```

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

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

```

947 \newcommand*{\hyxmp@embed@packet@xetex}{%
948   \special{pdf:stream @hyxmp@Metadata (\hyxmp@xml)
949     <<
950       /Type /Metadata
951       /Subtype /XML
952     >>
953   }%
954   \special{pdf:put @catalog
955     <<
956       /Metadata @hyxmp@Metadata
957     >>
958   }%
959 }

```

## 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.

```

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

```

## 4 Future Work

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

## A Sample XMP packet

The following is an example of a complete XMP packet as may be produced by `hyperxmp`. This packet corresponds to the metadata included in the sample L<sup>A</sup>T<sub>E</sub>X document presented on page 5. For clarity, metadata values, either specified explicitly by the document or introduced automatically by `hyperxmp`, are colored blue.

```

<?xpacket begin="\357\273\277" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="3.1-702">

```



```

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

```

```

<dc:creator>
  <rdf:Seq>
    <rdf:li>Albert Einstein</rdf:li>
  </rdf:Seq>
</dc:creator>
<dc:subject>
  <rdf:Bag>
    <rdf:li>energy quanta</rdf:li>
    <rdf:li>Hertz effect</rdf:li>
    <rdf:li>quantum physics</rdf:li>
  </rdf:Bag>
</dc:subject>
<dc:date>
  <rdf:Seq>
    <rdf:li>2013-04-16</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: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:Seq>
            </pdfaType:field>
        </rdf:li>
    </rdf:Seq>
</pdfaSchema:valueType>

```

```

        </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>
        <a href="http://einstein.biz/">http://einstein.biz/</a>,
        <a href="https://www.facebook.com/AlbertEinstein">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>2013-04-16</xmp:CreateDate>
    <xmp:ModifyDate>2013-04-16</xmp:ModifyDate>
    <xmp:MetadataDate>2013-04-16</xmp:MetadataDate>
    <xmp:CreatorTool>LaTeX with hyperref package</xmp:CreatorTool>
    <xmp:BaseURL>
        <a href="http://www.ctan.org/tex-archive/macros/latex/contrib/hyperxmp/">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>
        <a href="http://www.ctan.org/tex-archive/macros/latex/contrib/hyperxmp/">uuid:0595fdce-41dc-e4c4-6c418dc4ce46
    </xmpMM:DocumentID>

```

```

        <xmpMM:InstanceID>
          uuid:efd754c4-1d7f-200a-ef754ce413ea
        </xmpMM:InstanceID>
      </rdf:Description>
    </rdf:RDF>
  </x:xmpmeta>
  <?xpacket end="w"?>

```

## References

- [1] Adobe Systems, Inc., San Jose, California. *Adobe Acrobat X SDK Help, pdfmark Reference*. Available from <http://www.adobe.com/devnet/acrobat/documentation.html>.
- [2] Adobe Systems, Inc. *PostScript Language Reference Manual*. Addison-Wesley, 2nd edition, January 1996, ISBN: 0-201-18127-4.
- [3] Adobe Systems, Inc., San Jose, California. *Document Management—Portable Document Format—Part 1: PDF 1.7*, July 2008. ISO 32000-1 standard document. Available from [http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000\\_2008.pdf](http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000_2008.pdf).
- [4] Adobe Systems, Inc., San Jose, California. *XMP Specification Part 1: Data model, Serialization, and Core Properties*, 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](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](http://www.pdfa.org/wp-content/uploads/2011/08/tn0009_xmp_extension_schemas_in_pdfa-1_2008-03-20.pdf).

## Change History

v1.0	General: Initial version . . . . .	1	Basic schema and miscellaneous other bits of metadata . . . . .	1
v1.1	<code>\hyxmp@construct@packet</code> : Explicitly set the category codes of characters $\langle EF \rangle$ , $\langle BB \rangle$ , and $\langle BF \rangle$ to “letter”. Thanks to Daniel Schömer for the bug report . . . . .	36	Heiko Oberdiek’s major rewrite of the code to better support native-Unicode TeX implementations (XeTeX and LuaTeX) . . . . .	1
v1.2	General: Added support for the XeTeX backend ( <code>xdvipdfmx</code> ) . . . . .	1	New <code>\AtBeginDocument</code> code from Heiko Oberdiek to properly encode <code>\pdfmetalang</code> . . . . .	15
	Added support for the Photoshop schema . . . . .	1	<code>\hyxmp@add@to@xml</code> : Updated also to replace commas . . . . .	26
	Made the package compatible with <code>ngerman</code> . Thanks to Tobias Mueller for the bug report. . . . .	9	<code>\hyxmp@bom</code> : Added by Heiko Oberdiek . . . . .	36
v1.3	General: Introduced the <code>pdfmetalang</code> package option, which enables an author to specify the language in which he wrote the document’s metadata . . . . .	15	<code>\hyxmp@comma</code> : Added this macro . . . . .	16
	<code>\hyxmp@reencode</code> : Introduced this macro to re-encode Unicode strings as 8-bit strings before manipulating them into XMP schema. This change addresses a bug reported by Martin Münch . . . . .	18	<code>\hyxmp@construct@packet</code> : Modified by Heiko Oberdiek to use an appropriate BOM representation via <code>\hyxmp@bom</code> . . . . .	36
v1.4	<code>\hyxmp@mm@schema</code> : Renamed the <code>xapMM</code> namespace prefix to <code>xmpMM</code> . . . . .	31	<code>\hyxmp@crap@convert</code> : Added by Heiko Oberdiek . . . . .	22
	<code>\hyxmp@rdf@dc</code> : Included metadata in the <code>x-default</code> language regardless of the specified metadata language . . . . .	28	<code>\hyxmp@crap@test</code> : Added by Heiko Oberdiek . . . . .	22
	<code>\hyxmp@xmpRights@schema</code> : Renamed the <code>xapRights</code> namespace prefix to <code>xmpRights</code> . . . . .	30	<code>\hyxmp@dc@schema</code> : Added support for <code>dc:language</code> and <code>dc:source</code> . . . . .	30
v1.5	General: Made the XMP inclusion more robust. Thanks to Heiko Oberdiek for the bug report and suggested modifications. . . . .	9	<code>\hyxmp@is@unicode</code> : Added by Heiko Oberdiek . . . . .	19
v2.0	General: Added support for the XMP		<code>\hyxmp@list@to@xml</code> : Modified by Heiko Oberdiek to use the new Unicode-processing macros . . . . .	29
			<code>\hyxmp@photoshop@schema</code> : Simplified using <code>\hyxmp@add@simple</code> . . . . .	32
			<code>\hyxmp@ProcessKeyvalOptions</code> : Added this macro . . . . .	13
			<code>\hyxmp@reencode</code> : Replaced with an empty macro by Heiko Oberdiek . . . . .	18
			<code>\hyxmp@skiptorelax</code> : Added by Heiko Oberdiek . . . . .	22
			<code>\hyxmp@skipzeros</code> : Added by Heiko Oberdiek . . . . .	21
			<code>\hyxmp@SpaceOther</code> : Added by Heiko Oberdiek . . . . .	22
			<code>\hyxmp@string</code> : Added this macro . . . . .	28
			<code>\hyxmp@toxml</code> : Added by Heiko Oberdiek . . . . .	20
			Escaped parentheses written with <code>pdfmarks</code> to prevent dvips from	

line-wrapping the XMP packet	20	<code>\hypersetup</code> : Added this macro	13
<code>\hyxmp@toxml@unicodetex</code> : Added by Heiko Oberdiek	21	<code>\hyxmp@hypersetup</code> : Added this macro	13
<code>\hyxmp@xetex@crap</code> : Added by Heiko Oberdiek	21	<code>\hyxmp@redefine@Hyp</code> : Added this macro	12
<code>\hyxmp@xmlify</code> : Completely rewritten by Heiko Oberdiek to better support Unicode-enabled T <sub>E</sub> X programs	18	v2.2	
<code>\hyxmp@xmp@basic@schema</code> : Added this macro	31	General: Added support for the IPTC Photo Metadata schema	1
<code>\hyxmp@xmpRights@schema</code> : Modified to include <code>xmpRights:Marked</code> only when <code>pdfcopyright</code> is specified and <code>xmpRights:WebStatement</code> only when <code>pdflicenseurl</code> is specified	30	<code>\hyxmp@iptc@extensions</code> : Added this macro to support PDF/A generation	34
<code>\hyxmp@zero</code> : Added by Heiko Oberdiek	23	<code>\hyxmp@list@to@lines</code> : Added this macro	32
<code>\ifhyxmp@unicodetex</code> : Added by Heiko Oberdiek	18	<code>\hyxmp@photometa@schema</code> : Added this macro	33
<code>\ProcessKeyvalOptions</code> : Added this macro	13	<code>\hyxmp@text@resource</code> : Added this macro	36
<code>\xmpcomma</code> : Added this macro	16	<code>\xmpcomma</code> : Changed the default from <code>\relax</code> to an ordinary comma	16
<code>\xmpquote</code> : Added this macro	16	<code>\xmplinesep</code> : Added this macro	32
<code>\XMPTruncateList</code> : Added this macro	16	v2.3	
v2.1		<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	v2.3a	
		General: Bug fix: Redefine <code>\@pdflang</code> as <code>\@empty</code> when <code>hyperref</code> has set it to <code>\relax</code>	15

## 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>\!</code>	353, 360, 367, 374, 810	<code>\*</code>	278, 291
<code>\"</code>	1, 2, 960	<code>\,</code>	480
<code>\#</code>	485, 680	<code>\@baseurl</code>	97, 655
<code>\&amp;</code>	252, 284, 679	<code>\@elt</code>	169, 574, 692, 697
<code>\(</code>	262, 263	<code>\@elt@first</code>	690
<code>\)</code>	264, 265	<code>\@elt@rest</code>	692, 694
		<code>\@firstoftwo</code>	243
		<code>\@gobble</code>	174, 328
		<code>\@nil</code>	239, 241, 308, 325, 333
		<code>\@pdfauthor</code>	...
		<code>\@pdfauthor</code>	56, 98, 456, 463
		<code>\@pdfauthor</code>	...

.. 19, 99, 661, 669		
\@pdfcaptionwriter .		
.. 21, 100, 661, 670		
\@pdfcontactaddress		
.. 25, 101, 707, 726		
\@pdfcontactcity ...		
.. 33, 102, 708, 727		
\@pdfcontactcountry		
.. 39, 103, 711, 730		
\@pdfcontactemail ..		
.. 43, 104, 713, 734		
\@pdfcontactphone ..		
.. 41, 105, 712, 733		
\@pdfcontactpostcode		
.. 37, 106, 710, 729		
\@pdfcontactregion .		
.. 35, 107, 709, 728		
\@pdfcontacturl ...		
.. 45, 108, 714, 735		
\@pdfcopyright . 15,		
109, 597, 613, 623		
\@pdfcreator ..... 654		
\@pdfkeywords .....		
.. 73, 110, 514, 525		
\@pdflang .. 111, 126,		
127, 129, 132, 601		
\@pdflicenseurl ...		
.. 17, 112, 609, 629		
\@pdfmetalang 23, 130,		
132, 134, 553, 556		
\@pdfproducer . 515, 526		
\@pdfsubject .. 113, 596		
\@pdftitle .....		
.. 114, 456, 463, 595		
\@secondoftwo ..... 245		
\^ 165, 189, 480, 815–817		
\_ ..... 479		
\~ ..... 322, 323		
\_ ..... 288, 323, 479		
<b>A</b>		
ASCII ..... 19		
\AtBeginDocument ... 124		
\AtEndDocument ..... 5		
\AtEndDvi ..... 8		
atenddvi ..... 9		
Author ..... 6, 16		
	<b>B</b>	<b>H</b>
	baseurl (option) .... 3, 31	\Hy@driver ..... 4,
	BOM ..... 36, 46	846, 850, 854, 859
		hyperref .. 1, 3–6, 10,
		12, 14, 15, 37, 38, 47
	<b>C</b>	\hypersetup ..... 90
	CiAdrCity ..... 3, 33	hyperxmp . 1–6, 8–12,
	CiAdrCtry ..... 3, 33	14–16, 18, 24, 40, 47
	CiAdrExtadr ..... 3, 33	\hyxmp@is@unicode . 232
	CiAdrPcode ..... 3, 33	\hyxmp@add@simple ..
	CiAdrRegion ..... 3, 33	.. 525, 526, 528,
	CiEmailWork ..... 3, 33	535, 601, 602,
	CiTelWork ..... 3, 33	629, 651–655,
	CiUrlWork ..... 3, 33	669, 670, 727–730
		\hyxmp@add@to+xml ..
	<b>D</b>	..... 471,
	\day .. 464, 506, 507, 509	490, 521, 530,
	dc:creator ..... 2, 6, 29	540, 549, 555,
	dc:date ..... 2, 30	559, 569, 577,
	dc:description ..... 2, 29	583, 590, 603,
	dc:format ..... 2	619, 625, 630,
	dc:language .... 2, 29, 46	638, 647, 656,
	dc:rights ..... 2, 29	664, 673, 687,
	dc:source ..... 2, 30, 46	691, 695, 701,
	dc:subject ..... 2, 29	719, 739, 746,
	dc:title ..... 2, 29	787, 800, 823, 836
	\define@key .....	\hyxmp@append@hex ..
	.. 16, 18, 20, 22,	.... 418, 437–440
	24, 26, 34, 36, 38,	\hyxmp@append@hex@iv
	40, 42, 44, 46, 56, 73	.. 436, 444, 445,
	dvipdf (option) ..... 38	447, 449, 451–453
	dvipdfm ..... 38	\hyxmp@at@end ... 3, 135
	dvips (option) ..... 38	\hyxmp@big@prime ...
	dvips ..... 6, 20, 46	.. 392, 395, 405, 415
	dvipsone (option) .... 38	\hyxmp@big@prime@ii
	dviwindo (option) ... 38	..... 392, 414
		\hyxmp@bom .... 808, 823
	<b>E</b>	\hyxmp@comma .....
	\EdefEscapeHex 211, 224	... 27, 57, 74, 164
	\EdefUnescapeHex ... 228	\hyxmp@commas@to@list
	\EdefUnescapeString 198	.. 148, 171, 575, 699
	\endcsname .....	\hyxmp@commas@to@list@i
	53, 70, 170, 173, 195	..... 150, 152
	ETX ..... 16, 18	\hyxmp@concat@metadata
		..... 95
	<b>G</b>	\hyxmp@construct@packet
	\g@addto@macro ....	..... 821, 844
	.... 446, 448, 450	\hyxmp@count@non@spaces
	Ghostscript ..... 6	..... 928, 939



\hyxmp@count@spaces	\hyxmp@mm@schema	\hyxmp@sublist
..... 927, <u>930</u>	635, 835	.... 153, 154, 157, 158
\hyxmp@crap@convert	\hyxmp@modulo@a	\hyxmp@temp@list
..... 314, <u>348</u>	... 386, 405, 415, 421	... <u>169</u>
\hyxmp@crap@result	\hyxmp@num	\hyxmp@temp@str
..... 304, 340	..... <u>348</u>	... <u>169</u>
\hyxmp@crap@test	\hyxmp@one@token	\hyxmp@text
311, <u>336</u>	.... 394, 398,	..... 196, 274, 304, <u>348</u>
\hyxmp@create@uuid	931, 932, 940, 941	\hyxmp@text@resource
.... 442, 459, 469	\hyxmp@padding	.... 779–786, <u>799</u>
\hyxmp@dc@schema	<u>488</u> , 839	\hyxmp@today
<u>589</u> , 831	\hyxmp@pdf@schema	.... 500, 600, 651–653
\hyxmp@def@DocumentID	..... <u>512</u> , 829	\hyxmp@toxml
..... 455, 636	\hyxmp@pdfauthor	.. 226, <u>249</u>
\hyxmp@def@InstanceID	..... 47, <u>56</u> , 598	\hyxmp@toxml@unicodetex
..... 461, 637	\hyxmp@pdfkeywords	..... 214, <u>274</u>
\hyxmp@DocumentID	..... 47, 73, 599	\hyxmp@trimb
..... 455, 641	\hyxmp@photometa@data	.. 182, <u>185</u>
\hyxmp@dq@code	..... <u>705</u>	\hyxmp@trimc
.. <u>1</u> , 960	\hyxmp@photometa@schema	.. 185, <u>186</u>
\hyxmp@driver	..... 705, 833	\hyxmp@trimspace
... <u>3</u> , <u>843</u>	\hyxmp@photoshop@data	..... 157, <u>178</u>
\hyxmp@embed@packet	..... <u>660</u>	\hyxmp@try
..... 137, <u>843</u>	\hyxmp@photoshop@schema	..... <u>304</u>
\hyxmp@embed@packet@dvi	..... 660, 832	\hyxmp@unicodetexfalse
..... 851, 909	\hyxmp@ProcessKeyvalOptions	..... <u>188</u>
\hyxmp@embed@packet@pdfmark	..... <u>85</u>	\hyxmp@unicodetextrue
..... 863, <u>879</u>	\hyxmp@rand@num	..... <u>188</u>
\hyxmp@embed@packet@pdf	.. 411, 420, 458, 468	\hyxmp@x@default
..... 847, <u>869</u>	\hyxmp@rdf@dc	.. 130, <u>511</u> , 553, 560
\hyxmp@embed@packet@xetex	.... 545, 595–597	\hyxmp@xetex@crap
..... 855, <u>947</u>	\hyxmp@redefine@Hyp	..... 205, <u>304</u>
\hyxmp@find@metadata	..... 49, 87, 92	\hyxmp@xml
..... 95, 136	\hyxmp@reencode	..... 481,
\hyxmp@hash	... <u>194</u>	<u>488</u> , 821, 875,
.. 484, 749–752, 827	\hyxmp@rights	899, 910, 917, 948
\hyxmp@have@any	.. 608, 611, 615, 617	\hyxmp@xmlified
... <u>513</u>	\hyxmp@seed@rng	.... 196, 541,
\hyxmp@Hyp@pdfauthor	.... 394, 457, 467	556, 560, 575, 699
<u>50</u>	\hyxmp@seed@rng@i	\hyxmp@xmlify
\hyxmp@Hyp@pdfkeywords	..... 396, <u>398</u>	.... 134, <u>196</u> ,
..... 67	\hyxmp@seed@string	539, 548, 574, 698
\hyxmp@hypersetup	.. 456, 457, 462, 467	\hyxmp@xmp@basic@schema
.. <u>90</u>	\hyxmp@set@rand@num	..... 646, 834
\hyxmp@InstanceID	..... <u>411</u> , 419	\hyxmp@xmpRights@schema
..... 461, 642	\hyxmp@skiptorelax	..... 607, 830
\hyxmp@iptc@extensions	..... 341, <u>347</u>	\hyxmp@zero
..... 718, <u>745</u>	\hyxmp@skipzeros	.... 357,
\hyxmp@is@unicode	\hyxmp@SpaceOther	364, 371, 377, <u>382</u>
.... 200, 217, <u>232</u>	..... 308, <u>321</u>	
\hyxmp@legal	\hyxmp@string	
..... 608	..... <u>535</u>	
\hyxmp@list	\hyxmp@string@len	
.. 575, 581, 699, 700	..... 910, <u>925</u>	
\hyxmp@list@to@lines		
.. 683, 726, 733–735		
\hyxmp@list@to@xml		
.... 566, 598–600		

# I

IETF	4
\ifhyxmp@unicodetex	.... <u>188</u> , 199, 809
ifxetex	10
\ifxetex	204
Info	6
intcalc	10

<code>\intcalcDiv</code> 353, 360, 367	<code>pdfaType:prefix</code> . . . . . 47	<code>photoshop:AuthorsPosition</code>
<code>\intcalcMod</code> 355, 362, 369	<code>pdfauthor (option)</code> . . .	..... 2, 32
<code>IPTC</code> . 6, 11, 26, 33, 34, 47	..... 3, 8, 10, 12, 29	<code>photoshop:CaptionWriter</code>
<code>lptc4xmpCore:CreatorContactInfo</code>	<code>pdfauthorTitle (option)</code>	..... 2, 32
..... 3, 33, 47	..... 4, 9, 10	<code>PI</code> . . . . . 26
<code>ISO</code> . . . . . 10	<code>pdfcaptionwriter (option)</code> . . . . . 4, 10	<code>\ProcessKeyvalOptions</code>
	<code>\pdfcatalog</code> . . . . . 876	..... 85
	<code>\pdfcompresslevel</code> . . 871	<code>ps2pdf (option)</code> . . . . . 38
	<code>pdfcontactaddress (option)</code> . . . . . 4, 6, 7	
<b>J</b>	<code>pdfcontactcity (option)</code> . 4	<b>Q</b>
<code>\jobname</code> . . 118, 141, 456, 463, 602, 860	<code>pdfcontactcountry (option)</code> . . . . . 4	<code>\Q</code> . . . . . 178, 187
	<code>pdfcontactemail (option)</code> . . . . . 4	
<b>K</b>	<code>pdfcontactphone (option)</code> . . . . . 4	<b>R</b>
<code>Keywords</code> . . . . . 6	<code>pdfcontactpostcode (option)</code> . . . . . 4	<code>rdf:li</code> . . . . . 2
<code>\KV@Hyp@pdfauthor</code> . . 56	<code>pdfcontactregion (option)</code> . . . . . 4	<code>rdf:Seq</code> . . . . . 2
<code>\KV@Hyp@pdfkeywords</code> 73	<code>pdfcontacturl (option)</code> . 4	<code>\renewcommand</code> . . . . . 86
<code>kvoptions</code> . . . . . 10, 13	<code>pdfcopyright (option)</code> . . . 4, 10, 29, 30, 47	<code>\RequirePackage</code> 7, 10–14
	<code>PDFDocEncoding</code> . . . . . 12, 18, 19	
<b>L</b>	<code>pdfescape</code> . . . . . 10	<b>S</b>
<code>\lccode</code> . . . . . 233, 234, 323, 353, 360, 367, 374, 475, 479, 480, 810	<code>pdfkeywords (option)</code> . . . . . 3, 8, 10, 12, 29	<code>\SE-&gt;pdfdoc003</code> . . . . . 195
<code>LF</code> . . . . . 32	<code>pdflang (option)</code> . . . . . 3, 4, 10, 14, 15, 29	<code>\special</code> 911, 919, 948, 954
<code>\lowercase</code> . . . . . 237, 324, 354, 361, 368, 375, 481, 811	<code>\pdflastobj</code> . . . . . 876	<code>stringenc</code> . . . . . 10
<code>LuaL<sup>A</sup>T<sub>E</sub>X</code> . . . . . 6, 7	<code>pdfL<sup>A</sup>T<sub>E</sub>X</code> . . . . . 3, 5	<code>\StringEncodingConvert</code>
<code>LuaT<sub>E</sub>X</code> . . 18, 21, 40, 46	<code>pdflicenseurl (option)</code> . . . . . 4, 10, 30, 47	..... 201, 207, 218, 221, 316
	<code>pdfmark (option)</code> . . . . . 38	<code>Subject</code> . . . . . 6
<b>M</b>	<code>\pdfmark</code> . . 880, 883, 887, 897, 901, 905	
<code>Metadata</code> . . . . . 6, 37, 40	<code>pdfmetalang (option)</code> 4, 10	<b>T</b>
<code>\month</code> 464, 501, 502, 504	<code>\pdfminorversion</code> . . . 528	<code>T<sub>E</sub>X</code> . . . . . 18, 20, 21, 23, 31, 39, 40, 47
	<code>\pdfobj</code> . . . . . 872	<code>Text</code> . . . . . 36
<b>N</b>	<code>pdfproducer (option)</code> . . 3	<code>textures (option)</code> . . . . . 38
<code>nativepdf (option)</code> . . . 38	<code>\pdfstringdef</code> 16, 18, 20, 22, 24, 29, 34, 36, 38, 40, 42, 44, 46	<code>\time</code> . . . . . 465
<code>\newif</code> . . . . . 188	<code>pdfsubject (option)</code> . . . . . 3, 10, 29	<code>Title</code> . . . . . 6
<code>\next</code> . . . . . 152, 398	<code>pdfT<sub>E</sub>X</code> . . . 9, 20, 37, 40	
<code>ngerman</code> . . . . . 9, 46	<code>pdftitle (option)</code> 4, 10, 29	<b>U</b>
<code>\number</code> 351, 353, 355, 360, 362, 367, 369		<code>\uccode</code> . . . . . 278, 291
		<code>Unicode</code> . . . . . 10, 18–22, 29, 33, 36, 40, 46
<b>P</b>		<code>\uppercase</code> . . . . . 279, 292
<code>\PackageWarningNoLine</code>		<code>URL</code> 2–4, 10, 12, 30, 31, 34
..... 117, 140, 858		<code>\usepackage</code> . . . . . 142
<code>PDF</code> . . . 1–3, 5–8, 15, 20, 25–28, 36, 37, 40		<code>UTF-16BE</code> . . . . . 19
<code>PDF/A</code> . . . . . 34, 47		<code>UTF-32BE</code> . . . . . 18, 19
<code>pdf:Keywords</code> . . . . . 2, 28		<code>UTF-8</code> . . . . . 19
<code>pdf:PDFVersion</code> . . . . . 2		<code>UUID</code> . . . . . 23, 25, 26
<code>pdf:Producer</code> . . . . . 2, 28		<b>V</b>
		<code>\vfuzz</code> . . . . . 186
		<code>vtexpdfmark (option)</code> . 38

X			
\x	.....	<a href="#">304</a>	
xdvipdfmx	.....	7, 40	
X <sub>2</sub> L <sup>A</sup> T <sub>E</sub> X	.....	6, 7	
X <sub>2</sub> L <sub>E</sub> X	.	10, 18, 21, 40, 46	
XML	1, 2, 6, 15, 18–21,		
	26, 28, 29, 32, 34, 36		
XMP	.....	1–3,	
	6–10, 15–17, 20,		
	23, 26–28, 30, 31,		
	34, 37, 38, 40, 46, 47		
xmp:BaseURL	.....	2	xmpMM:InstanceID ..
xmp:CreateDate	.....	2	..... 2, 23, 31
xmp:CreatorTool	.....	2	\xmpquote .....
xmp:MetadataDate	....	2	28, 31, <a href="#">56</a> , <a href="#">73</a> , <a href="#">168</a>
xmp:ModifyDate	.....	2	xmpRights:Marked 2, 30, 47
\xmpcomma	.....		xmpRights:WebStatement
		27, 30, <a href="#">56</a> , <a href="#">73</a> , <a href="#">163</a>	..... 2, 30, 47
xmpincl	.....	3	\XMPTruncateList ... <a href="#">169</a>
\xmplinesep	<a href="#">678</a> , 695, <a href="#">731</a>		
xmpMM:DocumentID	.		<b>Y</b>
	.....	2, 23, 31	\year ..... 464, 500