

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

Scott Pakin  
scott+hyxmp@pakin.org

November 26, 2017

## 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` v3.4, dated 2017/11/26.

```

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

```

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

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

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

- authors (`dc:creator`)
- base URL (`xmp:BaseURL`)
- contact address (`lptc4xmpCore:CreatorContactInfo/CiAdrExtadr`,  
`lptc4xmpCore:CreatorContactInfo/CiAdrCity`,  
`lptc4xmpCore:CreatorContactInfo/CiAdrRegion`,  
`lptc4xmpCore:CreatorContactInfo/CiAdrPcode`, and  
`lptc4xmpCore:CreatorContactInfo/CiAdrCtry`)
- contact email address(es) (`lptc4xmpCore:CreatorContactInfo/CiEmailWork`)
- contact telephone number(s) (`lptc4xmpCore:CreatorContactInfo/CiTelWork`)
- contact URL(s) (`lptc4xmpCore:CreatorContactInfo/CiUrlWork`)
- copyright (`dc:rights` and `xmpRights:Marked`)
- date (`dc:date`, `xmp:CreateDate`, `xmp:ModifyDate`, and `xmp:MetadataDate`)
- document identifier (`xmpMM:DocumentID`)
- document instance identifier (`xmpMM:InstanceID`)
- document type (`dc:type`)
- file format (`dc:format`)
- keywords (`pdf:Keywords` and `dc:subject`)
- language (`dc:language`)
- L<sup>A</sup>T<sub>E</sub>X file name (`dc:source`)
- license URL (`xmpRights:WebStatement`)

- metadata writer (`photoshop:CaptionWriter`)
- PDF version (`pdf:PDFVersion`)
- PDF-generating tool (`pdf:Producer` and `xmp:CreatorTool`)
- PDF/A compliance level and version (`pdfaid:part` and `pdfaid:conformance`)
- primary author’s position/title (`photoshop:AuthorsPosition`)
- summary (`dc:description`)
- title (`dc:title`)

More types of metadata may be added in a future release.

**How does `hyperxmp` compare to the `xmpincl` package?** The short answer is that `xmpincl` is more flexible but `hyperxmp` is easier to use. With `xmpincl`, the author manually constructs a file of arbitrary XMP data and the package merely embeds it within the generated PDF file. With `hyperxmp`, the author specifies values for various predefined metadata types and the package formats those values as XMP and embeds the result within the generated PDF file.

`xmpincl` can embed XMP only when running under `pdfLATEX` and only when in PDF-generating mode. `hyperxmp` additionally works with a few other PDF-producing `LATEX` backends.

`hyperxmp` and `xmpincl` can complement each other. An author may want to use `hyperxmp` to produce a basic set of XMP code, then extract the XMP code from the PDF file with a text editor, augment the XMP code with any metadata not supported by `hyperxmp`, and use `xmpincl` to include the modified XMP code in the PDF file.

## 2 Usage

`hyperxmp` works by postprocessing some of the package options honored by `hyperref`. To use `hyperxmp`, merely put a `\usepackage{hyperxmp}` in your document’s preamble. That line can appear anywhere before the `hyperref` PDF options are specified (i.e., with either `\usepackage[...]{hyperref}` or `\hypersetup{...}`). `hyperxmp` will construct its XMP data using the following `hyperref` options:

- `baseurl`
- `pdfauthor`
- `pdfcreationdate`
- `pdfkeywords`
- `pdflang`

- pdfmoddate
- pdfproducer
- pdfsubject
- pdftitle

hyperxmp instructs hyperref also to accept the following options, which have meaning only to hyperxmp:

- pdfaformance
- pdfapart
- pdfauthortitle
- pdfcaptionwriter
- pdfcontactaddress
- pdfcontactcity
- pdfcontactcountry
- pdfcontactemail
- pdfcontactphone
- pdfcontactpostcode
- pdfcontactregion
- pdfcontacturl
- pdfcopyright
- pdfdate
- pdflicenseurl
- pdfmetadate
- pdfmetalang
- pdfsource
- pdftype

The two most obscure—but alphabetically first—of the above, `pdfaconformance` and `pdfapart`, are used in conjunction with `hyperref`’s `pdfa` option to claim a particular PDF/A standard by which the document abides. They default to `pdfapart=1` and `pdfaconformance=B`, indicating the PDF/A-1B standard. These can be changed (with caution) to assert that the document abides by a different standard (e.g., PDF/A-2U).

`pdfauthor` 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).

XMP metadata can include a number of dates (in fact, timestamps, as they include both date and time components). `pdfdate` specifies the document date. It is analogous to the  $\text{\LaTeX}$  `\date` command, and, like `\date`, defaults to the date the document was built. It must be specified in either XMP format [4] or PDF format [3]. XMP dates are written in the form `YYYY-MM-DDThh:mm:ss+TT:tt`.<sup>1</sup> A W3C recommendation [12] discusses this format in more detail, but as an example, 14 hours, 15 minutes, 9 seconds past midnight U.S. Mountain Daylight Time (UTC-6) on the 23rd day of September in the year 2014 should be written as `2014-09-23T14:15:09-06:00`. This can be truncated (with loss of information) to `2014-09-23T14:15:09` or `2014-09-23T14:15` or `2014-09-23` or `2014-09` or `2014` but no other subsets. PDF dates are written in the form `D:YYYYMMDDhhmmss+TT’tt’`. The same date in the preceding example would be written as `D:20140923141509-06’00’` in PDF format.

The document’s creation date, modification date, and metadata date are normally set automatically, but `pdfcreationdate`, `pdfmoddate`, and `pdfmetadate` can be used to override the defaults. Like `pdfdate`, `pdfmetadate` can be specified in either XMP or PDF format. However, because `hyperref` defines `pdfcreationdate` and `pdfmoddate` and expects these to be written as PDF dates, `hyperxmp` concomitantly accepts these two dates only in PDF format as well. Note that it’s rare that a document would need to specify any of `pdfcreationdate`, `pdfmoddate`, or `pdfmetadate`.

`pdfcopyright` defines the copyright text, and `pdflicenseurl` identifies a URL that points to the document’s license agreement.

`pdfmetalang` indicates the natural language in which certain metadata—specifically, the document’s title, subject, and copyright statement—are written. The language should be specified using an IETF language tag [8], for example, “en” for English, “en-US” for specifically United States English, “de” for German,

---

<sup>1</sup>Although allowed by XMP, `hyperxmp` does not currently accept fractions of a second in timestamps.

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 are always included in addition to the specified metadata language, as the user reading the document may not have specified a language preference.

A rarely needed option, `pdfsource`, overrides the name of the L<sup>A</sup>T<sub>E</sub>X source file. It defaults to “`\jobname.tex`” but can be replaced by any other string. If `pdfsource` is given an empty argument, no document source will be specified at all.

`pdftype` describes the type of document being produced. This refers to “the nature or genre of the resource” [4] such as “poem”, “novel” or “working paper”, as opposed to the file format (always “application/pdf” when generated by `hyperxmp`). Although `pdftype` can be assigned an arbitrary piece of text, the XMP specification recommends selecting types from a “controlled vocabulary” such as the DCMI Type Vocabulary [5]. The DCMI Type Vocabulary currently consists of only “Collection”, “Dataset”, “Event”, “Image”, “InteractiveResource”, “MovingImage”, “PhysicalObject”, “Service”, “Software”, “Sound”, “StillImage”, and “Text”. `pdftype` defaults to “Text”, which refers to “books, letters, dissertations, poems, newspapers, articles, archives of mailing lists,” [5] and other forms of text—all things L<sup>A</sup>T<sub>E</sub>X is commonly used to typeset.

It is usually more convenient to provide values for those options using `hyperref`'s `\hypersetup` command than on the `\usepackage` command line. See the `hyperref` manual for more information. The following is a sample L<sup>A</sup>T<sub>E</sub>X document that provides values for most of the metadata options that `hyperxmp` recognizes:

```
\documentclass{article}
\usepackage{hyperxmp}
\usepackage[unicode]{hyperref}
\title{%
  On a heuristic viewpoint concerning the production and
  transformation of light}
\author{Albert Einstein}
\date{March 17, 1905}
\hypersetup{%
  pdftitle={%
    On a heuristic viewpoint concerning the production and
    transformation of light},
  pdfauthor={Albert Einstein},
  pdfauthortitle={Technical Assistant, Level III},
  pdfdate={1905-03-17},
  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},
pdfmetalang={en},
baseurl={http://mirror.ctan.org/macros/latex/contrib/hyperxmp/}
}
\XMPLangAlt{de}{pdftitle={Über einen die Erzeugung und Verwandlung
  des Lichtes betreffenden heuristischen Gesichtspunkt}}
\begin{document}
\maketitle
A profound formal difference exists between the theoretical
concepts that physicists have formed about gases and other
ponderable bodies, and Maxwell's theory of electromagnetic
processes in so-called empty space\dots
\end{document}

```

Compile the document to PDF using any of the following approaches:

- pdf $\LaTeX$
- Lua $\LaTeX$
- $\LaTeX$  + Dvipdfm
- $\LaTeX$  + Dvips + Adobe Acrobat Distiller
- Xe $\LaTeX$

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

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

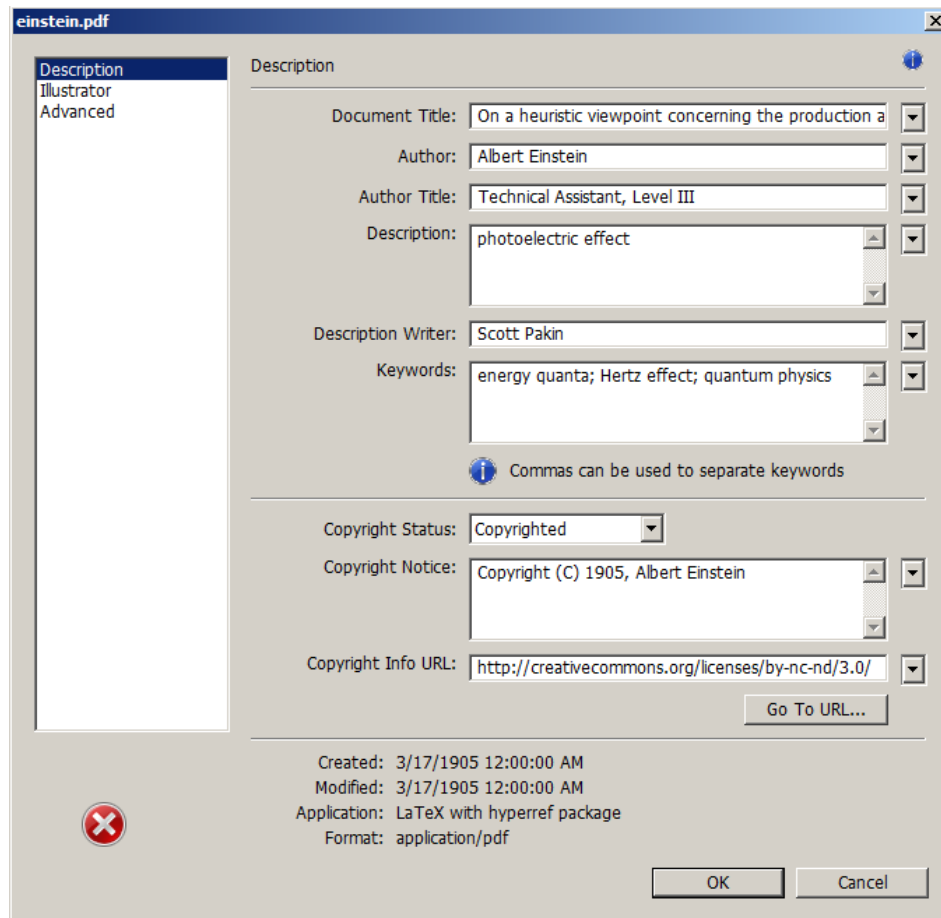


Figure 1: XMP metadata as it appears in Adobe Acrobat

**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,



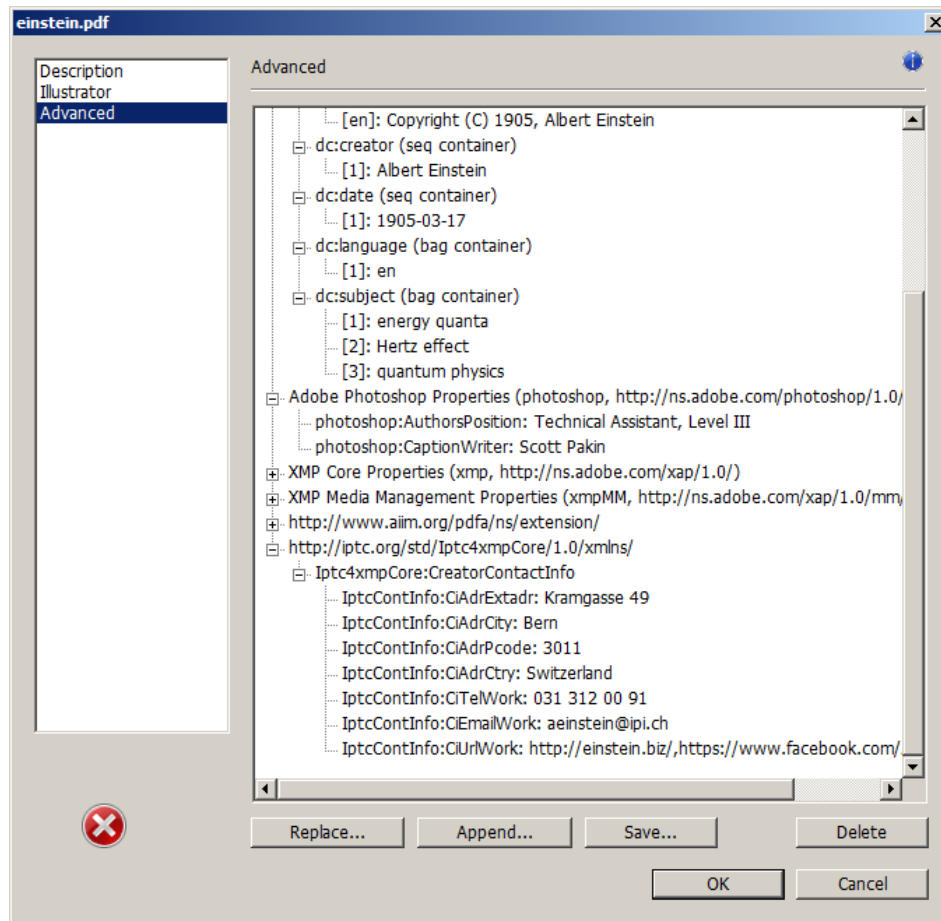


Figure 2: Additional XMP metadata as it appears in Adobe Acrobat

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” [7]. `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,

`\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: Object compression** One intention of XMP is that metadata embedded in a file be readable even without knowledge of the file’s format. That is, the metadata are expected to appear as plain text. Although `hyperxmp` does its best to honor that intention, it faces a few challenges:

1. When run with versions of `LuaLaTeX` earlier than 0.85, `hyperxmp` leaves all PDF objects uncompressed. This is due to `LuaLaTeX` treating object compression as a global parameter, unlike `pdfLaTeX`, which treats it as a local parameter. Hence, when `hyperxmp` requests that the XMP packet be left uncompressed, `LuaLaTeX` in fact leaves *all* PDF streams uncompressed. Beginning with version 3.0, `hyperxmp` includes a workaround that correctly leaves only the XMP metadata uncompressed, but this workaround is implemented only for `LuaLaTeX` v0.85 onwards.
2. `XLaTeX` (or, more precisely, the `xdvipdfmx` back end) exhibits the opposite problem. It 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., `LuaLaTeX`), (2) pass the `--output-driver="xdvipdfmx -z0"` option to `XLaTeX` to instruct `xdvipdfmx` to turn off all compression (which will of course make the PDF file substantially larger), or (3) postprocess the generated PDF file by loading it into the commercial version of Adobe Acrobat and re-saving it with the Save As... menu option.

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

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

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

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

`\xmpcomma` If you need to include a literal comma within an author or keyword list (where  
`\xmpquote` commas normally separate list items) or a street address (where commas normally

separate lines), use the `\xmpcomma` macro to represent it, and wrap the entire entry containing the comma within `\xmpquote{...}` as shown below:

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

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

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

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

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

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

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

**Note 6: Automatically specified metadata**    `pdftitle` defaults to the document’s title as specified by `\title{...}`. `pdfauthor` defaults to the document’s author(s) as specified by `\author{...}`. `pdfdate` defaults to the current date and time. `pdfmetalang` defaults to the same value as `pdflang` if non-empty, “x-default” otherwise. An implication of automatic metadata specification is that an author can simply include `\usepackage{hyperxmp}` in a document’s preamble and benefit from a modicum of XMP metadata with no additional effort.

`\XMPLangAlt`    **Note 7: Multilingual metadata**    The `pdfmetalang` option specifies the language in which the document’s metadata is written. It defaults to the value of `pdflang`, which specifies the document language. As of version 3.3 of `hyperxmp`, it is

possible to include certain metadata—specifically, the document’s title, subject, and copyright statement—in more than one language. The `\XMPLangAlt` macro provides this functionality. Usage is as follows:

```
\XMPLangAlt {<language>} { <option>=<text>, ... }
```

where *<language>* is an ISO 639-1 two-letter country code with an optional ISO 3166-1 two-letter region code (e.g., “en” for English or “en-US” for specifically US English); *<option>* is one of “`pdftitle`”, “`pdfsubject`”, or “`pdfcopyright`”; and *<text>* is the text as expressed in the specified language. By way, of example, the following code provides the document title in English then specifies an alternative title to use in four other languages:

```
\hypersetup{%
  pdfmetalang={en},
  pdftitle={English title}
}
\XMPLangAlt{de}{pdftitle={Deutscher Titel}}
\XMPLangAlt{fr}{pdftitle={Titre fran\c{c}ais}}
\XMPLangAlt{it}{pdftitle={Titolo italiano}}
\XMPLangAlt{rm}{pdftitle={Titel rumantsch}}
```

## 3 Implementation

This section presents the commented L<sup>A</sup>T<sub>E</sub>X source code for `hyperxmp`. Read this section only if you want to learn how `hyperxmp` is implemented.

### 3.1 Initial preparation

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

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

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

```
\hyxmp@driver
3 \def\hyxmp@driver{hpdftex}
4 \ifx\hyxmp@driver\Hy@driver
5   \let\hyxmp@at@end=\AtEndDocument
6 \else
```

```

7 \RequirePackage{atendddvi}
8 \let\hyxmp@at@end=\AtEndDvi
9 \fi

```

### 3.2 Integration with hyperref

An important design decision underlying `hyperxmp` is that the package should integrate seamlessly with `hyperref`. To that end, `hyperxmp` takes its XMP metadata from `hyperref`'s `baseurl`, `pdfauthor`, `pdfkeywords`, `pdflang`, `pdfproducer`, `pdfsubject`, and `pdftitle` options. It also introduces a number of new options, which are listed on pages 4–5. 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), `ifxetex` for detecting X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X, and `ifmtarg` for testing if a macro argument is empty or all spaces.

```

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

```

`\@ifmtargexp` `\@ifmtarg` and `\@ifnotmtarg` do not expand their first argument. Define `\@ifnotmtargexp` `\@ifmtargexp` and `\@ifnotmtargexp` as expanding versions of those macros.

```

16 \def\@ifmtargexp#1{\expandafter\@ifmtarg\expandafter{#1}}
17 \def\@ifnotmtargexp#1{\expandafter\@ifnotmtarg\expandafter{#1}}

```

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

```

18 \newcommand{\hyxmp@pdfstringdef}[2]{%
19 \let\hyxmp@textunderscore=\textunderscore
20 \let\textunderscore=\hyxmp@uscore
21 \pdfstringdef{#1}{#2}%
22 \let\textunderscore=\hyxmp@textunderscore
23 }

```

`\@pdfdatetime` Prepare to store the document's date and (optionally) time. Whether specified by the author in XMP format or PDF format (cf. Section 3.3.2) we always store `\@pdfdatetime` as an XMP-format string.

```

24 \def\@pdfdatetime{}
25 \define@key{Hyp}{pdfdate}{%
26   \begingroup
27     \Hy@unicodefalse

\@pdfmetadatetime Expand pdfdate's argument and convert it to XMP format.
28   \edef\next{%
29     \noexpand\hyxmp@pdfstringdef\noexpand\@pdfdatetime{%
30       \noexpand\hyxmp@as@xmp@date{#1}}%
31   }%
32   \next
33 \endgroup
34 }

\@pdfcopyright Prepare to store the document's metadata date and (optionally) time. Whether
                specified by the author in XMP format or PDF format (cf. Section 3.3.2) we always
                store \@pdfmetadatetime as an XMP-format string.
35 \def\@pdfmetadatetime{}
36 \define@key{Hyp}{pdfmetadate}{%
37   \begingroup
38     \Hy@unicodefalse

\@pdfcopyright Prepare to store the document's copyright statement.
39   \edef\next{%
40     \noexpand\hyxmp@pdfstringdef\noexpand\@pdfmetadatetime{%
41       \noexpand\hyxmp@as@xmp@date{#1}}%
42   }%
43   \next
44 \endgroup
45 }

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

\@pdftype Prepare to store the document's logical type, which defaults to "Text".
48 \def\@pdftype{Text}
49 \define@key{Hyp}{pdftype}{\hyxmp@pdfstringdef\@pdftype{#1}}

\@pdflicenseurl Prepare to store the document's license agreement.
50 \def\@pdflicenseurl{}
51 \define@key{Hyp}{pdflicenseurl}{\hyxmp@pdfstringdef\@pdflicenseurl{#1}}

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

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

```

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

```
56 \def\@pdfmetalang{}
57 \define@key{Hyp}{pdfmetalang}{\hyxmp@pdfstringdef\@pdfmetalang{#1}}
```

`\@pdfapart` Prepare to store the PDF/A part ID, which defaults to “1”.

```
58 \def\@pdfapart{1}
59 \define@key{Hyp}{pdfapart}{\hyxmp@pdfstringdef\@pdfapart{#1}}
```

`\@pdfaconformance` Prepare to store the PDF/A conformance ID, which defaults to “B”.

```
60 \def\@pdfaconformance{B}
61 \define@key{Hyp}{pdfaconformance}{\hyxmp@pdfstringdef\@pdfaconformance{#1}}
```

`\@pdfsource` Prepare to store the document’s source, which defaults to the value of `\jobname`.

```
62 \edef\@pdfsource{\jobname.tex}
63 \define@key{Hyp}{pdfsource}{\hyxmp@pdfstringdef\@pdfsource{#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.

```
64 \def\@pdfcontactaddress{}
65 \define@key{Hyp}{pdfcontactaddress}{%
66   \let\xmpcomma=\hyxmp@comma
67   \def\xmpquote##1{##1}%
68   \hyxmp@pdfstringdef\@pdfcontactaddress{#1}%
69   \def\xmpcomma{,}%
70   \let\xmpquote=\relax
71 }
```

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

```
72 \def\@pdfcontactcity{}
73 \define@key{Hyp}{pdfcontactcity}{\hyxmp@pdfstringdef\@pdfcontactcity{#1}}
```

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

```
74 \def\@pdfcontactregion{}
75 \define@key{Hyp}{pdfcontactregion}{\hyxmp@pdfstringdef\@pdfcontactregion{#1}}
```

<code>\@pdfcontactpostcode</code>	Prepare to store the postal code of the document's contact person/institution. 76 <code>\def\@pdfcontactpostcode{}</code> 77 <code>\define@key{Hyp}{pdfcontactpostcode}{\hyxmp@pdfstringdef\@pdfcontactpostcode{#1}}</code>
<code>\@pdfcontactcountry</code>	Prepare to store the country of the document's contact person/institution. 78 <code>\def\@pdfcontactcountry{}</code> 79 <code>\define@key{Hyp}{pdfcontactcountry}{\hyxmp@pdfstringdef\@pdfcontactcountry{#1}}</code>
<code>\@pdfcontactphone</code>	Prepare to store the telephone number of the document's contact person/institution. 80 <code>\def\@pdfcontactphone{}</code> 81 <code>\define@key{Hyp}{pdfcontactphone}{\hyxmp@pdfstringdef\@pdfcontactphone{#1}}</code>
<code>\@pdfcontactemail</code>	Prepare to store the email address of the document's contact person/institution. 82 <code>\def\@pdfcontactemail{}</code> 83 <code>\define@key{Hyp}{pdfcontactemail}{\hyxmp@pdfstringdef\@pdfcontactemail{#1}}</code>
<code>\@pdfcontacturl</code>	Prepare to store the URL of the document's contact person/institution. 84 <code>\def\@pdfcontacturl{}</code> 85 <code>\define@key{Hyp}{pdfcontacturl}{\hyxmp@pdfstringdef\@pdfcontacturl{#1}}</code>

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

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

<code>\hyxmp@pdfauthor</code> <code>\hyxmp@pdfkeywords</code>	Prepare to store the name of the author and a list of keywords. 86 <code>\def\hyxmp@pdfauthor{}</code> 87 <code>\def\hyxmp@pdfkeywords{}</code>
<code>\hyxmp@redefine@Hyp</code>	If not already redefined, redefine <code>hyperref</code> 's <code>pdfauthor</code> and <code>pdfkeywords</code> options to properly handle <code>\xmpcomma</code> and <code>\xmpquote</code> . 88 <code>\newcommand*{\hyxmp@redefine@Hyp}{%</code>
<code>\hyxmp@Hyp@pdfauthor</code>	Store the old definition of <code>\KV@Hyp@pdfauthor</code> in <code>\hyxmp@Hyp@pdfauthor</code> , but only if we see that <code>\KV@Hyp@pdfauthor</code> is defined and <code>\hyxmp@Hyp@pdfauthor</code> isn't. Otherwise, we'd be defining <code>\hyxmp@Hyp@pdfauthor</code> in terms of itself and creating an infinite loop. 89 <code>\@ifundefined{KV@Hyp@pdfauthor}{\}{%</code>



```

90   \ifundefined{hyxmp@Hyp@pdfauthor}{%
91     \expandafter\let\expandafter\hyxmp@Hyp@pdfauthor
92       \csname KV@Hyp@pdfauthor\endcsname
93   }{}%
94 }%

```

`\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, `\and` 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). In case `pdfauthor` is left unspecified and we copy `\author`'s argument to `pdfauthor`, we temporarily redefine `\and` as the list separator when producing a structured list and as “and” when producing an unstructured list.

```

95   \define@key{Hyp}{pdfauthor}{%
96     \let\xmpcomma=\hyxmp@comma
97     \def\xmpquote####1{####1}%
98     \let\hyxmp@and=\and
99     \def\and{,}%
100    \hyxmp@Hyp@pdfauthor{##1}%
101    \global\let\hyxmp@pdfauthor=\@pdfauthor
102    \def\and{and\space}%
103    \def\xmpcomma{,}%
104    \def\xmpquote####1{"####1"%
105    \hyxmp@Hyp@pdfauthor{##1}%
106    \def\xmpcomma{,}%
107    \let\xmpquote=\relax
108    \let\and=\hyxmp@and
109  }%

```

`\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`.

```

110  \ifundefined{KV@Hyp@pdfkeywords}{}%
111  \ifundefined{hyxmp@Hyp@pdfkeywords}{%
112    \expandafter\let\expandafter\hyxmp@Hyp@pdfkeywords
113      \csname KV@Hyp@pdfkeywords\endcsname
114  }{}%
115 }%

```

`\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).

```

116 \define@key{Hyp}{pdfkeywords}{%
117   \let\xmpcomma=\hyxmp@comma
118   \def\xmpquote####1{####1}%
119   \hyxmp@Hyp@pdfkeywords{##1}%
120   \global\let\hyxmp@pdfkeywords=\@pdfkeywords
121   \def\xmpcomma{,%}
122   \def\xmpquote####1{"####1"%
123   \hyxmp@Hyp@pdfkeywords{##1}%
124   \def\xmpcomma{,%}
125   \let\xmpquote=\relax
126 }%
127 }

```

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

```

128 \let\hyxmp@ProcessKeyvalOptions=\ProcessKeyvalOptions
129 \renewcommand*{\ProcessKeyvalOptions}{%
130   \hyxmp@redefine@Hyp
131   \hyxmp@ProcessKeyvalOptions
132 }

```

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

```

133 \let\hyxmp@hypersetup=\hypersetup
134 \def\hypersetup{%
135   \hyxmp@redefine@Hyp
136   \hyxmp@hypersetup
137 }

```

`\hyxmp@find@metadata`   Issue a warning message if the author failed to specify any metadata at all. This excludes metadata that is included automatically such as the current timestamp. `\hyxmp@concat@metadata` Note that we don't consider `\@pdfmetalang` as metadata as that value is meaningful only when used in conjunction with other information. We also don't examine `\@pdfapart` or `\@pdfaconformance` because those have nonempty default values.

```

138 \newcommand*{\hyxmp@find@metadata}{%
139   \edef\hyxmp@concat@metadata{%
140     \@baseurl
141     \@pdfauthor
142     \@pdfauthortitle
143     \@pdfcaptionwriter
144     \@pdfcontactaddress
145     \@pdfcontactcity
146     \@pdfcontactcountry
147     \@pdfcontactemail
148     \@pdfcontactphone
149     \@pdfcontactpostcode
150     \@pdfcontactregion

```

```

151 \pdfcontacturl
152 \pdfcopyright
153 \pdfcreationdate
154 \pdfdatetime
155 \pdfkeywords
156 \pdflang
157 \pdflicenseurl
158 \pdfmetadatetitle
159 \pdfmoddate
160 \pdfsubject
161 \pdftitle
162 \pdftype
163 }%
164 \ifx\hyxmp@concat@metadata\@empty
165 \PackageWarningNoLine{hyperxmp}{%
166 \jobname.tex did not specify any metadata to\MessageBreak
167 include in the XMP packet.\space\space Please see the\MessageBreak
168 hyperxmp documentation for instructions on how to\MessageBreak
169 provide metadata values to hyperxmp}%
170 \fi
171 }

```

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.

```

172 \AtBeginDocument{%
173 \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`.

```

174 \ifx\pdflang\relax
175 \let\pdflang=\@empty
176 \fi

```

If the author 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.

```

177 \ifx\pdfmetalang\@empty
178 \ifx\pdflang\@empty
179 \let\pdfmetalang=\hyxmp@x@default
180 \else
181 \edef\pdfmetalang{\pdflang}%
182 \fi
183 \fi
184 \hyxmp@xmlify\pdfmetalang

```

If the author explicitly specified the document date, override the compilation timestamp with the specified date.

```

185 \ifx\@pdfdatetime\@empty
186 \else
187 \edef\hyxmp@today{\@pdfdatetime}%
188 \fi

```

If the author left pdftitle blank but specified \title, use the title for pdftitle. Likewise, if the author left pdfauthor blank but specified \author, use the author for pdfauthor.

```

189 \@ifmtargexp{\@pdftitle}{%
190 \ifnotmtargexp{\@title}{%
191 \hypersetup{pdftitle={\@title}}}%
192 }%
193 }%
194 {}%
195 \@ifmtargexp{\@pdfauthor}{%
196 \ifnotmtargexp{\@author}{%
197 \hypersetup{pdfauthor={\@author}}}%
198 }%
199 }%
200 {}%

```

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.

```

201 \hyxmp@at@end{%
202 \hyxmp@find@metadata
203 \hyxmp@embed@packet
204 }%
205 }{%
206 \PackageWarningNoLine{hyperxmp}{%
207 \jobname.tex failed to include a\MessageBreak
208 \string\usepackage\string{hyperref\string}
209 in the preamble.\MessageBreak
210 Consequently, all hyperxmp functionality will be\MessageBreak
211 disabled}%
212 }%
213 }

```

### 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 3.3.3); and, in Section 3.3.4, 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.)
214 \newcommand*{\hyxmp@commas@to@list}[2]{%
215   \gdef#1{%
216     \expandafter\hyxmp@commas@to@list@i\expandafter#1#2,,%
217   }

\hyxmp@commas@to@list@i  Recursively construct macro #1 from comma-separated list #2. Stop if #2 is empty.
\next 218 \def\hyxmp@commas@to@list@i#1#2,{%
219   \gdef\hyxmp@sublist{#2}%
220   \ifx\hyxmp@sublist\@empty
221     \let\next=\relax
222   \else
223     \hyxmp@trimspaces\hyxmp@sublist
224     \@cons{#1}{\hyxmp@sublist}%
225     \def\next{\hyxmp@commas@to@list@i{#1}}%
226   \fi
227   \next
228 }

\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.
229 \def\xmpcomma{,}%

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

230 \bgroup
231   \catcode'\^C=11
232   \gdef\hyxmp@comma{^C}
233 \egroup

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

234 \bgroup
235   \catcode'\^U=11
236   \gdef\hyxmp@uscore{^U}
237 \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.

```
238 \let\xmpquote=\relax
```

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

```
239 \bgroup
240 \catcode'\~=12%
241 \gdef\xmptilde{~}%
242 \egroup
```

`\XMPTruncateList` As a workaround for Adobe Acrobat's inability to display author lists correctly (cf. “Acrobat Author bug” on page 8) we introduce a hack that replaces a list with its first element. One can then write “`\XMPTruncateList{pdfauthor}`” and have Adobe Acrobat display the author list correctly. It's sad that this is necessary, though.

```
243 \newcommand{\XMPTruncateList}[1]{%
244 \edef\hyxmp@temp@str{\csname hyxmp@#1\endcsname}%
245 \hyxmp@commas@to@list{\hyxmp@temp@list}{\hyxmp@temp@str}%
246 \def\@elt##1{%
247 \expandafter\gdef\csname @#1\endcsname{##1}%
248 \let\@elt=\@gobble
249 }
250 \hyxmp@temp@list
251 }}
```

### 3.3.2 Date manipulation

hyperxmp needs to manipulate two types of date (really, timestamp) formats: PDF format and XMP format. PDF timestamps are of the form “D:YYYYMMDDhhmmss+TT'tt'” (e.g., D:20171126225445-07'00') [3], while XMP timestamps are of the form “YYYY-MM-DDThh:mm:ss+TT:tt” (e.g., 2017-11-26T22:54:45-07:00) [4]. The `\hyxmp@as@pdf@date` and `\hyxmp@as@xmp@date` macros defined in this section facilitate timestamp conversions to PDF and XMP formats, respectively.

`\hyxmp@first@char` Return the first character of a string. This macro is fully expandable.

```
\hyxmp@first@char@i 252 \def\hyxmp@first@char#1{\hyxmp@first@char@i#1\relax}
253 \def\hyxmp@first@char@i#1#2\relax{#1}
```

`\hyxmp@as@xmp@date` If necessary, convert a timestamp to XMP format. That is, if the timestamp is in PDF format, convert it; otherwise, leave it unmodified. This macro is fully expandable.

```

254 \def\hyxmp@as@xmp@date#1{%
255   \expandafter\ifx\hyxmp@first@char@i#1\relax D%
256     \hyxmp@pdf@to@xmp@date{#1}%
257   \else
258     #1%
259   \fi
260 }

```

`\hyxmp@pdf@to@xmp@date` Convert a timestamp from PDF format to XMP format. This macro is fully expandable.

```

261 \def\hyxmp@pdf@to@xmp@date#1:#2#3#4#5#6#7#8#9{%
262   #2#3#4#5-#6#7-#8#9%
263   \hyxmp@parse@time
264 }

```

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

```

265 \def\hyxmp@parse@time#1#2#3#4#5#6{%
266   T#1#2:#3#4:#5#6%
267   \hyxmp@parse@tz@char
268 }

```

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

```

269 \def\hyxmp@parse@tz@char#1{%
270   #1%
271   \ifx#1-%
272     \expandafter\hyxmp@parse@tz
273   \else
274     \ifx#1+%
275       \expandafter\hyxmp@parse@tz
276     \fi
277   \fi
278 }

```

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

```

279 \def\hyxmp@parse@tz#1'#2'{%
280   #1:#2%
281 }

```

`\hyxmp@as@pdf@date` If necessary, convert a timestamp to PDF format. That is, if the timestamp is in XMP format, convert it; otherwise, leave it unmodified. This macro is fully expandable.

```

282 \def\hyxmp@as@pdf@date#1{%
283   \expandafter\ifx\hyxmp@first@char@i#1\relax D%
284     #1%
285   \else
286     \hyxmp@xmp@to@pdf@date{#1}%
287   \fi
288 }

```

`\hyxmp@xmp@to@pdf@date` Convert a timestamp from XMP format to PDF format. This macro is fully expandable.

```

289 \def\hyxmp@xmp@to@pdf@date#1{%
290   D:\hyxmp@xmp@to@pdf@date@i#1\relax\relax
291 }

```

`\hyxmp@xmp@to@pdf@date@i` Parse the year for `\hyxmp@xmp@to@pdf@date`.

```

292 \def\hyxmp@xmp@to@pdf@date@i#1#2#3#4#5#6{%
293   #1#2#3#4%
294   \ifx#5-%
295     \expandafter\hyxmp@xmp@to@pdf@date@ii\expandafter#6%
296   \fi
297 }

```

`\hyxmp@xmp@to@pdf@date@ii` Parse the month for `\hyxmp@xmp@to@pdf@date`.

```

298 \def\hyxmp@xmp@to@pdf@date@ii#1#2#3#4{%
299   #1#2%
300   \ifx#3-%
301     \expandafter\hyxmp@xmp@to@pdf@date@iii\expandafter#4%
302   \fi
303 }

```

`\hyxmp@xmp@to@pdf@date@iii` Parse the day for `\hyxmp@xmp@to@pdf@date`.

```

304 \def\hyxmp@xmp@to@pdf@date@iii#1#2#3#4{%
305   #1#2%
306   \ifx#3T%
307     \expandafter\hyxmp@xmp@to@pdf@date@iv\expandafter#4%
308   \fi
309 }

```

`\hyxmp@xmp@to@pdf@date@iv` Parse the hour for `\hyxmp@xmp@to@pdf@date`.

```

310 \def\hyxmp@xmp@to@pdf@date@iv#1#2#3#4{%
311   #1#2%
312   \ifx#3:%
313     \expandafter\hyxmp@xmp@to@pdf@date@v\expandafter#4%
314   \fi
315 }

```



`\hyxmp@xmp@to@pdf@date@v` Parse the minute for `\hyxmp@xmp@to@pdf@date`.

```

316 \def\hyxmp@xmp@to@pdf@date@v#1#2#3#4{%
317   #1#2%
318   \ifx#3:%
319     \expandafter\hyxmp@xmp@to@pdf@date@vi\expandafter#4%
320   \fi
321 }

```

`\hyxmp@gobbletwo` This is exactly the same as L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>'s `\@gobbletwo` but needs to be a different literal for `\hyxmp@xmp@to@pdf@date@vii`'s pattern-matching to work.

```

322 \let\@hyxmp@gobbletwo=\@gobbletwo

```

`\hyxmp@xmp@to@pdf@date@vi` Parse the second for `\hyxmp@xmp@to@pdf@date`. The challenge here is that we need to handle four cases for the character following the seconds—“+”, “-”, “Z”, and no character—without sacrificing expandability. Our tricky solution is to insert a `\@gobbletwo` as a sentinel and let `\hyxmp@xmp@to@pdf@date@vi` discard everything up to that sentinel (i.e., all the other conditionals).

```

323 \def\hyxmp@xmp@to@pdf@date@vi#1#2#3#4{%
324   #1#2%
325   \ifx#3+%
326     +\expandafter\hyxmp@xmp@to@pdf@date@vii
327   \fi
328   \ifx#3-%
329     -\expandafter\hyxmp@xmp@to@pdf@date@vii
330   \fi
331   \ifx#3Z%
332     Z%
333   \fi
334   \ifx#3\relax
335     \expandafter\@hyxmp@gobbletwo
336   \fi
337   \@gobbletwo #4%
338 }

```

`\hyxmp@xmp@to@pdf@date@vii` Parse the time-zone hours for `\hyxmp@xmp@to@pdf@date`.

```

339 \def\hyxmp@xmp@to@pdf@date@vii#1\@gobbletwo#2#3#4#5{%
340   #2#3%
341   \ifx#4:%
342     \expandafter\hyxmp@xmp@to@pdf@date@viii\expandafter#5%
343   \fi
344 }

```

`\hyxmp@xmp@to@pdf@date@viii` Parse the time-zone minutes for `\hyxmp@xmp@to@pdf@date`.

```

345 \def\hyxmp@xmp@to@pdf@date@viii#1#2#3#4{%
346   '#1#2'%
347 }

```

`\hyxmp@today@define` Use T<sub>E</sub>X primitives to define a given macro as today's date in YYYY-MM-DDThh:mm format.

```

348 \def\hyxmp@today@define#1{%
    The date is a straightforward representation of TEX's \year, \month, and \day
    primitives, with the latter two zero-padded to two digits apiece.
349 \xdef#1{\the\year}%
350 \ifnum\month<10
351     \xdef#1{#1-0\the\month}%
352 \else
353     \xdef#1{#1-\the\month}%
354 \fi
355 \ifnum\day<10
356     \xdef#1{#1-0\the\day}%
357 \else
358     \xdef#1{#1-\the\day}%
359 \fi

```

T<sub>E</sub>X does not provide the time in terms of separate hours and minutes but rather as the total number of minutes since midnight (`\time`). There's no mechanism in T<sub>E</sub>X to query the number of seconds since midnight or the timezone so we omit those fields when defining macro #1.

```

360 \@tempcnta=\time
361 \divide\@tempcnta by 60%
362 \ifnum\@tempcnta<10%
363     \xdef#1{#1T0\the\@tempcnta}%
364 \else
365     \xdef#1{#1T\the\@tempcnta}%
366 \fi
367 \multiply\@tempcnta by -60%
368 \advance\@tempcnta by \time
369 \ifnum\@tempcnta<10%
370     \xdef#1{#1:0\the\@tempcnta}%
371 \else
372     \xdef#1{#1:\the\@tempcnta}%
373 \fi
374 }

```

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

```

375 \@ifundefined{pdffeedback}{%
376     \@ifundefined{pdfcreationdate}{%

```

Case 1: Neither `\pdffeedback` nor `\pdfcreationdate` is defined (X<sub>q</sub>L<sub>A</sub>T<sub>E</sub>X and regular L<sub>A</sub>T<sub>E</sub>X).

```

377     \hyxmp@today@define\hyxmp@today
378 }{%

```

Case 2: `\pdfcreationdate` is defined (pdfL<sub>A</sub>T<sub>E</sub>X and pre-0.85 LuaL<sub>A</sub>T<sub>E</sub>X).

```

379     \edef\hyxmp@today{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
380 }%
381 }{%

```

Case 3: `\pdffeedback` is defined (Lua $\TeX$  0.85+).

```
382 \edef\hyxmp@today{\expandafter\hyxmp@pdf@to@xmp@date\pdffeedback creationdate}%
383 }
```

### 3.3.3 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 [6]. Inline comments are also taken from the solution text.

```
384 \catcode'\Q=3
\hyxmp@trimspaces\x redefines \x to have the same replacement text sans leading
and trailing space tokens.
385 \newcommand{\hyxmp@trimspaces}[1]{%
  Use grouping to emulate a multi-token afterassignment queue.
386 \begingroup
  Put “\toks 0 {” into the afterassignment queue.
387 \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.
388 \expandafter\hyxmp@trimb\expandafter\noexpand#1Q Q}%
  Transfer the trimmed text back into #1.
389 \edef#1{\the\toks0}%
390 }
```

`\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.

```
391 \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.

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

### 3.3.4 Converting text to XML

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

```

\ifhyxmp@unicodetex XqTeX and LuaTeX natively support Unicode. We define the conditional
\hyxmp@unicodetexttrue \ifhyxmp@unicodetex to check for these so we can properly handle encoding
\hyxmp@unicodetextfalse conversions. The trick here is that Unicode TEX implementations compare decimal
64 to hexadecimal 40 (decimal 64), specified with four carets, and take the
TRUE branch; non-Unicode TEX implementations compare decimal 64 to character
“^” (decimal 94), ignore the “^^0040” and the rest of the TRUE branch, and
take the FALSE branch.
394 \newif\ifhyxmp@unicodetex
395 \ifnum64='\^^^0040\relax
396   \hyxmp@unicodetexttrue
397 \else
398   \hyxmp@unicodetextfalse
399 \fi

\hyxmp@reencode This is now a placeholder macro needed only for \@pdfmetalang in the
\begin{document}.
400 \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.
401 \expandafter\def\csname SE->pdfdoc@03\endcsname{0003}

\SE->pdfdoc@15 Preserve NAK (^^U), which is normally an invalid character in PDFDocEncoding.
We use it in hyperxmp (and specifically in \hyxmp@xmlify below) as a placeholder
for an underscore character.
402 \expandafter\def\csname SE->pdfdoc@15\endcsname{0015}

\hyxmp@xmlify Given a piece of text defined using \pdfstringdef (i.e., with many special charac-
\hyxmp@xmlified ters redefined to have category code 11), set \hyxmp@xmlified to the same text
\hyxmp@text but with all occurrences of “<” replaced with &lt;; all occurrences of “>” replaced
with &gt;; and all occurrences of “&” replaced with &amp;.
403 \newcommand*{\hyxmp@xmlify}[1]{%
404   \gdef\hyxmp@xmlified{%
Escaped PDF string → PDFDocEncoding/Unicode
405   \EdefUnescapeString\hyxmp@text{#1}%
406   \ifhyxmp@unicodetex
PDFDocEncoding/Unicode → UTF-32BE
407     \hyxmp@is@unicode\hyxmp@text{%
408       \StringEncodingConvert
409       \hyxmp@text\hyxmp@text{utf16be}{utf32be}%
410     }{%

```

```

411     \ifxetex
412       \hyxmp@xetex@crap
413     \else
414       \StringEncodingConvert
415       \hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
416     \fi
417   }%

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

  UTF-32BE → XML in ASCII
419   \edef\hyxmp@text{%
420     \expandafter
421   }\expandafter\hyxmp@toxml@unicodetex\hyxmp@text
422   \relax\relax\relax\relax\relax\relax\relax\relax
423   \else

  PDFDocEncoding/Unicode → UTF-8
424   \hyxmp@is@unicode\hyxmp@text{%
425     \StringEncodingConvert
426     \hyxmp@text\hyxmp@text{utf16be}{utf8}%
427   }{%
428     \StringEncodingConvert
429     \hyxmp@text\hyxmp@text{pdfdoc}{utf8}%
430   }%

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

  UTF-8 as hex string → XML in UTF-8 as hex string
432   \edef\hyxmp@text{%
433     \expandafter\hyxmp@toxml\hyxmp@text\@empty\@empty
434   }%

  XML in UTF-8 as hex string → XML in UTF-8
435   \EdefUnescapeHex\hyxmp@text{\hyxmp@text}%
436   \fi
437   \global\let\hyxmp@xmlified\hyxmp@text
438 }

```

`\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.

```

439 \begingroup
440   \lccode'\<=254 %
441   \lccode'\>=255 %
442   \catcode254=12 %
443   \catcode255=12 %
444 \lowercase{\endgroup
445   \def\hyxmp@is@unicode#1{%
446     \expandafter\hyxmp@@is@unicode#1<>\@nil
447   }%

```

```

448 \def\hyxmp@is@unicode#1<>#2\@nil{%
449   \ifx\#1\%
450     \expandafter\@firstoftwo
451   \else
452     \expandafter\@secondoftwo
453   \fi
454 }%
455 }

```

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

```

456 \def\hyxmp@toxml#1#2{%
457   \ifx#1\empty
458   \else
459     \ifnum"#1#2='\& %
460       26616D703B% &amp;
461     \else\ifnum"#1#2='\< %
462       266C743B% &lt;
463     \else\ifnum"#1#2='\> %
464       2667743B% &gt;
465     \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.

```

466   \@ifundefined{pdfmark}{%
467     #1#2%
468   }{%
469     \ifnum"#1#2='\( %
470       5C28% \(
471     \else\ifnum"#1#2='\) %
472       5C29% \)
473     \else
474       #1#2%
475     \fi\fi
476   }%
477   \fi\fi\fi
478   \expandafter\hyxmp@toxml
479 \fi
480 }

```

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

```

481 \def\hyxmp@toxml@unicodetex#1#2#3#4#5#6#7#8{%
482   \ifx#1\relax
483   \else
484     \ifnum"#1#2#3#4#5#6#7#8>127 %
485       \uccode'\*="#1#2#3#4#5#6#7#8\relax
486       \uppercase{%
487         \edef\hyxmp@text{\hyxmp@text *}%
488       }%
489     \else\ifnum"#7#8='< %
490       \edef\hyxmp@text{\hyxmp@text &lt;}%
491     \else\ifnum"#7#8='& %
492       \edef\hyxmp@text{\hyxmp@text &amp;}%
493     \else\ifnum"#7#8='> %
494       \edef\hyxmp@text{\hyxmp@text &gt;}%
495     \else\ifnum"#7#8='\ %
496       \edef\hyxmp@text{\hyxmp@text\space}%
497     \else
498       \uccode'\*="#7#8\relax
499       \uppercase{%
500         \edef\hyxmp@text{\hyxmp@text *}%
501       }%
502     \fi\fi\fi\fi\fi
503     \expandafter\hyxmp@toxml@unicodetex
504   \fi
505 }
```

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

```

506 \def\hyxmp@skipzeros#1{%
507   \ifx#10%
508     \expandafter\hyxmp@skipzeros
509   \fi
510 }
```

`\x` In the case of  $X_{\text{f}}\TeX$ , the strings defined by `\pdfstringdef` can contain big characters. In this case, the string is treated as Unicode.

```

\hyxmp@xetex@crap \hyxmp@try 511 \begingroup
\hyxmp@crap@result 512 \def\x#1{\endgroup
\hyxmp@text 513   \def\hyxmp@xetex@crap{%
514     \edef\hyxmp@try{%
515       \expandafter\hyxmp@SpaceOther\hyxmp@text#1@nil
516     }%
517     \let\hyxmp@crap@result=N%
518     \expandafter\hyxmp@crap@test\hyxmp@try\relax
519     \ifx\hyxmp@crap@result Y%
520       \let\hyxmp@text\@empty
521       \expandafter\hyxmp@crap@convert\hyxmp@try\relax
522     \else
```

```

523      \StringEncodingConvert\hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
524      \fi
525    }%
526  }
527  \x{ }

```

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

```

528 \begingroup
529   \catcode'\~=12 %
530   \lccode'\~=\' %
531 \lowercase{\endgroup
532   \def\hyxmp@SpaceOther#1 #2\@nil{%
533     #1%
534     \ifx\relax#2\relax
535       \expandafter\@gobble
536     \else
537       ~%
538       \expandafter\@firstofone
539     \fi
540     {\hyxmp@SpaceOther#2\@nil}%
541   }%
542 }

```

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

```

543 \def\hyxmp@crap@test#1{%
544   \ifx#1\relax
545   \else
546     \ifnum'#1>127 %
547       \let\hyxmp@crap@result=Y%
548       \expandafter\expandafter\expandafter\hyxmp@skiptorelax
549     \else
550       \expandafter\expandafter\expandafter\hyxmp@crap@test
551     \fi
552   \fi
553 }

```

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

```

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

```

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

```

\hyxmp@num 555 \def\hyxmp@crap@convert#1{%
\hyxmp@text 556   \ifx#1\relax
557   \else
558     \edef\hyxmp@num{\number'#1}%
559     \ifnum\hyxmp@num>"FFFFFF %
560       \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"1000000}\relax
561       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
562     \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"1000000}}%
563   \else

```



```

564     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
565   \fi
566   \ifnum\hyxmp@num>"FFFF %
567     \lccode'\!=\intcalDiv{\hyxmp@num}{\number"10000}\relax
568     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
569     \edef\hyxmp@num{\intcalMod{\hyxmp@num}{\number"10000}}%
570   \else
571     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
572   \fi
573   \ifnum\hyxmp@num>"FF %
574     \lccode'\!=\intcalDiv{\hyxmp@num}{\number"100}\relax
575     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
576     \edef\hyxmp@num{\intcalMod{\hyxmp@num}{\number"100}}%
577   \else
578     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
579   \fi
580   \ifnum\hyxmp@num>0 %
581     \lccode'\!=\hyxmp@num\relax
582     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
583   \else
584     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
585   \fi
586   \expandafter\hyxmp@crap@convert
587 \fi
588 }

```

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

```

589 \begingroup
590   \catcode0=12 %
591   \gdef\hyxmp@zero{^^00}%
592 \endgroup

```

### 3.3.5 Providing metadata in multiple languages

Certain XMP tags—`dc:title`, `dc:description`, and `dc:rights` (and others? Let me know.)—can be expressed in multiple languages. The same text is used for both language `pdfmetalang` (default: `pdflang`) and language “`x-default`”. To express the same metadata in multiple languages, we provide an `\XMPLangAlt` macro to construct a list of alternative forms for a piece of metadata.

`\hyxmp@alt@title` Each of these macros is a list in which each element is of the form “`\do <language>`  
`\hyxmp@alt@description` `<text>`” in which `<language>` is an ISO 639-1 two-letter country code with an optional  
`\hyxmp@alt@rights` ISO 3166-1 two-letter region code. For example, `\hyxmp@alt@title` may contain  
 an element, “`\do {es-MX} {Este es mi documento}`”.

```

593 \def\hyxmp@alt@title{}
594 \def\hyxmp@alt@description{}
595 \def\hyxmp@alt@rights{}

```

`\hyxmp@LA@accept` This macro wraps `\define@key` to make the option “`#1= $\langle value \rangle$ ” append  $\langle value \rangle$  to list #2.`

```

596 \newcommand{\hyxmp@LA@accept}[2]{%
597   \define@key{hyxmp@LA}{#1}{%

```

`\hyxmp@value` As Niklas Beisert observed, if the option passed to the current key contains L<sup>A</sup>T<sub>E</sub>X code, this code will be included in the XMP packet, which is undesirable. Hence, we first clean up the string using `\hyxmp@pdfstringdef`.

```

598   \hyxmp@pdfstringdef\hyxmp@value{##1}%
599   \xdef#2{#2\noexpand\do {\hyxmp@cur@lang} {\hyxmp@value}}%
600 }
601 }

```

Define  $\langle key \rangle = \langle value \rangle$  options for appending to each of the `\hyxmp@alt $\langle tag \rangle$`  lists.

```

602 \hyxmp@LA@accept{pdftitle}{\hyxmp@alt@title}
603 \hyxmp@LA@accept{pdfsubject}{\hyxmp@alt@description}
604 \hyxmp@LA@accept{pdfcopyright}{\hyxmp@alt@rights}

```

`\XMPLangAlt` Argument `#1` is a language expressed as a two-letter country code and optional two-letter region code. Argument `#2` is a list of  $\langle key \rangle = \langle value \rangle$  pairs. Keys correspond to `\hypersetup` options such as “`pdftitle`”, “`pdfsubject`”, and “`pdfcopyright`”. Values are the alternative-language form of the text provided for the corresponding option.

```

605 \newcommand{\XMPLangAlt}[2]{%
606   \let\do=\relax

```

`\hyxmp@cur@lang` Store the provided language, which will be used during option processing.

```

607   \edef\hyxmp@cur@lang{#1}%
608   \setkeys{hyxmp@LA}{#2}%
609 }

```

### 3.4 UUID generation

We use a linear congruential generator to produce pseudorandom version 4 UUIDs [9]. True, this method has its flaws but it’s simple to implement in T<sub>E</sub>X and is good enough for producing the XMP `xmpMM:DocumentID` and `xmpMM:InstanceID` fields.

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

```

610 \def\hyxmp@modulo@a#1{%
611   \@tempcntb=\@tempcnta
612   \divide\@tempcntb by #1
613   \multiply\@tempcntb by #1
614   \advance\@tempcnta by -\@tempcntb
615 }

```

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

```

\hyxmp@big@prime@ii 616 \def\hyxmp@big@prime{536870923}
                    617 \def\hyxmp@big@prime@ii{536870027}

```

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

```

\hyxmp@one@token 618 \def\hyxmp@seed@rng#1{%
                    619   \@tempcnta=\hyxmp@big@prime
                    620   \futurelet\hyxmp@one@token\hyxmp@seed@rng@i#1\@empty
                    621 }

```

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

```

\hyxmp@one@token 622 \def\hyxmp@seed@rng@i{%
                    623   \ifx\hyxmp@one@token\@empty
                    624     \let\next=\relax
                    625   \else
                    626     \def\next##1{%
                    627       \multiply\@tempcnta by 3
                    628       \advance\@tempcnta by '##1
                    629       \hyxmp@modulo@a{\hyxmp@big@prime}%
                    630       \futurelet\hyxmp@one@token\hyxmp@seed@rng@i
                    631     }%
                    632   \fi
                    633   \next
                    634 }

```

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

```

\hyxmp@rand@num 635 \def\hyxmp@set@rand@num{%
                    636   \@tempcnta=\hyxmp@rand@num
                    637   \multiply\@tempcnta by 3
                    638   \advance\@tempcnta by \hyxmp@big@prime@ii
                    639   \hyxmp@modulo@a{\hyxmp@big@prime}%
                    640   \xdef\hyxmp@rand@num{\the\@tempcnta}%
                    641 }

```

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

```

642 \def\hyxmp@append@hex#1{%
643   \hyxmp@set@rand@num
644   \@tempcnta=\hyxmp@rand@num
645   \hyxmp@modulo@a{16}%
646   \ifnum\@tempcnta<10
647     \xdef#1{#1\the\@tempcnta}%
648   \else

```

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

```

649     \advance\@tempcnta by -10
650     \ifcase\@tempcnta
651         \xdef#1{#1a}%
652         \or\xdef#1{#1b}%
653         \or\xdef#1{#1c}%
654         \or\xdef#1{#1d}%
655         \or\xdef#1{#1e}%
656         \or\xdef#1{#1f}%
657     \fi
658 \fi
659 }

```

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

```

660 \def\hyxmp@append@hex@iii#1{%
661     \hyxmp@append@hex#1%
662     \hyxmp@append@hex#1%
663     \hyxmp@append@hex#1%
664 }

```

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

```

665 \def\hyxmp@append@hex@iv#1{%
666     \hyxmp@append@hex@iii#1%
667     \hyxmp@append@hex#1%
668 }

```

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

```

669 \def\hyxmp@create@uuid#1{%
670     \def#1{uuid:}%
671     \hyxmp@append@hex@iv#1%
672     \hyxmp@append@hex@iv#1%
673     \g@addto@macro#1{-}%
674     \hyxmp@append@hex@iv#1%
675     \g@addto@macro#1{-4}%
676     \hyxmp@append@hex@iii#1%
677     \g@addto@macro#1{-}%

```

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

```

678     \hyxmp@set@rand@num
679     \@tempcnta=\hyxmp@rand@num
680     \hyxmp@modulo@a{4}%
681     \ifcase\@tempcnta
682         \g@addto@macro#1{8}%
683         \or\g@addto@macro#1{9}%
684         \or\g@addto@macro#1{a}%

```

```

685     \or\g@addto@macro#1{b}%
686     \fi
687     \hyxmp@append@hex@iii#1%
688     \g@addto@macro#1{-}%
689     \hyxmp@append@hex@iv#1%
690     \hyxmp@append@hex@iv#1%
691     \hyxmp@append@hex@iv#1%
692 }

```

`\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.

```

693 \newcommand*{\hyxmp@def@DocumentID}{%
694   \edef\hyxmp@seed@string{\jobname:\@pdftitle:\@pdfauthor}%
695   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
696   \edef\hyxmp@rand@num{\the\@tempcnta}%
697   \hyxmp@create@uuid\hyxmp@DocumentID
698 }

```

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

```

699 \newcommand*{\hyxmp@def@InstanceID}{%
700   \edef\hyxmp@seed@string{\jobname:\@pdftitle:\@pdfauthor:\hyxmp@today}%
701   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
702   \edef\hyxmp@rand@num{\the\@tempcnta}%
703   \hyxmp@create@uuid\hyxmp@InstanceID
704 }

```

### 3.5 Constructing the XMP packet

An XMP packet “shall consist of the following, in order: a header PI, the serialized XMP data model (the XMP packet) with optional white-space padding, and a trailer PI” [4]. (“PI” is an abbreviation for “processing instructions”). The serialized XMP includes blocks of XML for various XMP schemata: Adobe PDF (Section 3.5.2), Dublin Core (Section 3.5.3), XMP Rights Management (Section 3.5.4), XMP Media Management (Section 3.5.5), XMP Basic (Section 3.5.6), Photoshop (Section 3.5.7), IPTC Photo Metadata (Section 3.5.8), and PDF/A Identification (Section 3.5.9). The `\hyxmp@construct@packet` macro (Section 3.5.10) constructs the XMP packet into `\hyxmp+xml`. It first writes the appropriate XML header, then calls the various schema-writing macros, then injects `\hyxmp@padding` as padding, and finally writes the appropriate XML trailer.

#### 3.5.1 XMP utility functions

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

```

705 \newcommand*{\hyxmp@add@to@xml}[1]{%
706   \bgroup
707     \@tempcnta=0
708     \ifhyxmp@unicodetex
709       \@tempcntb=65536%
710     \else
711       \@tempcntb=256%
712     \fi
713   \loop
714     \lccode\@tempcnta=\@tempcnta
715     \advance\@tempcnta by 1
716     \ifnum\@tempcnta<\@tempcntb
717   \repeat
718   \lccode'\_='\ \relax
719   \lccode'\^C='\,\relax
720   \lccode'\^U='\_\relax
721   \lowercase{\xdef\hyxmp@new@xml{#1}}%
722   \xdef\hyxmp@xml{\hyxmp@xml\hyxmp@new@xml}%
723 \egroup
724 }

```

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

```

725 \bgroup
726 \catcode'\#=11
727 \gdef\hyxmp@hash{#}
728 \egroup

```

`\hyxmp@padding` The XMP specification recommends leaving approximately 2000 bytes of whitespace at the end of each XMP packet to facilitate editing the packet in place [4].  
`\hyxmp@xml` `\hyxmp@padding` is defined to contain 32 lines of 63 spaces and a newline apiece for a total of 2048 characters of whitespace.

```

729 \bgroup
730 \xdef\hyxmp@xml{%
731   \hyxmp@add@to@xml{%
732     -----^^J%
733   }
734   \xdef\hyxmp@padding{\hyxmp@xml}%
735 \egroup
736 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
737 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
738 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
739 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
740 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}

```

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

```

741 \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.

```

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

743 \hyxmp@add@to@xml{%
744 -----<rdf:Description rdf:about=""^^J%
745 -----xmlns:pdf="http://ns.adobe.com/pdf/1.3/">^^J%
746 }%
747 \hyxmp@add@simple@var{pdf:Keywords}{@pdfkeywords}%
748 \hyxmp@add@simple@var{pdf:Producer}{@pdfproducer}%
749 \@ifundefined{pdfvariable}{%
750     \@ifundefined{pdfminorversion}{%
        Case 1: Neither \pdfvariable nor \pdfminorversion is defined (XeLaTeX and
        regular LATEX).
751     }{%
        Case 2: \pdfminorversion is defined (pdfLATEX and pre-0.85 LuaLATEX).
752         \hyxmp@add@simple{pdf:PDFVersion}{1.\the\pdfminorversion}%
753     }%
754 }{%
        Case 3: \pdfvariable is defined (LuaLATEX 0.85+).
755     \hyxmp@add@simple{pdf:PDFVersion}{1.\the\pdfvariable minorversion}%
756 }%
757 \hyxmp@add@to@xml{%
758 -----</rdf:Description>^^J%
759 }%
760 }

```

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

```

\hyxmp@string
761 \newcommand*{\hyxmp@add@simple}[2]{%
762     \edef\hyxmp@string{#2}%
763     \ifx\hyxmp@string\empty
764     \else
765         \hyxmp@xmlify{\hyxmp@string}%
766         \hyxmp@add@to@xml{%
767 -----<#1>\hyxmp@xmlified</#1>^^J%

```

```

768     }%
769   \fi
770 }

```

`\hyxmp@add@simple@var` Given an XMP tag (#1) and a variable name (#2), if the string is defined, add a begin tag, the string, and an end tag to the packet. The “simple” in the macro name indicates that the string is output without variations for different languages. `\hyxmp@add@simple@var` differs from `\hyxmp@add@simple` in that the former includes defined but empty values in the XMP packet while the latter excludes both undefined and defined but empty values.

```

771 \newcommand*{\hyxmp@add@simple@var}[2]{%
772   \expandafter\ifx\csname#2\endcsname\relax
773   \else
774     \hyxmp@xmlify{\csname#2\endcsname}%
775     \hyxmp@add@to@xml{%
776       -----<#1>\hyxmp@xmlified</#1>^^J%
777     }%
778   \fi
779 }

```

### 3.5.3 The Dublin Core schema

`\hyxmp@rdf@dc` Given an optional `\if<something>` statement (#1), a Dublin Core property (#2) and a macro containing some `\pdfstringdef`-defined text (#3), append the appropriate block of XML to the `\hyxmp@xml` macro.

```

780 \newcommand*{\hyxmp@rdf@dc}[3][\iffalse]{%
  Set \@tempswatrue only if the given text is nonempty or the provided conditional
  evaluates to TRUE.
781   \ifmtargexp{#3}{\@tempswafalse}{\@tempswatrue}%
782   #1
783   \@tempswatrue
784   \fi
  Append the corresponding XML only if \@tempswatrue.
785   \if@tempswa
786     \hyxmp@xmlify{#3}%

```

`\hyxmp@value` Store the XML-ified version of #3 in `\hyxmp@value` so we can reuse `\hyxmp@xmlified` if necessary.

```

787   \let\hyxmp@value=\hyxmp@xmlified
788   \hyxmp@add@to@xml{%
789     -----<dc:#2>^^J%
790     -----<rdf:Alt>^^J%
791   }%
792   \ifx\@pdfmetalang\hyxmp@x@default
793   \else
794     \hyxmp@xmlify{\@pdfmetalang}%
795     \hyxmp@add@to@xml{%

```



```

796 -----<rdf:li xml:lang="\hyxmp@xmlified">\hyxmp@value</rdf:li>^^J%
797     }%
798     \fi
799     \hyxmp@add@to@xml{%
800 -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@value</rdf:li>^^J%
801     }%

```

Include variants of the text expressed in other languages, as specified by the author using \XMLLangAlt (Section 3.3.5).

```

802     \def\do##1##2{
803         \hyxmp@xmlify{##2}%
804         \hyxmp@add@to@xml{%
805 -----<rdf:li xml:lang="##1">\hyxmp@xmlified</rdf:li>^^J%
806         }%
807     }%
808     \csname hyxmp@alt@#2\endcsname

```

Complete this XMP element.

```

809     \hyxmp@add@to@xml{%
810 -----</rdf:Alt>^^J%
811 -----</dc:#2>^^J%
812     }%
813     \fi
814 }%

```

`\hyxmp@list@to@xml` Given an optional `\if<something>` statement (#1), a Dublin Core property (#2), an RDF array (#3), and a macro containing a comma-separated list (#4), append the appropriate block of XML to the `\hyxmp@xml` macro.

```

815 \newcommand*{\hyxmp@list@to@xml}[4][\iffalse]{%

```

Set `\@tempswattrue` only if the given list is nonempty or the provided conditional evaluates to TRUE.

```

816     \ifmtargexp{#4}{\@tempswafalse}{\@tempswattrue}%
817     #1
818     \@tempswattrue
819     \fi

```

Append the corresponding XML only if `\@tempswattrue`.

```

820     \if@tempswa
821         \hyxmp@add@to@xml{%
822 -----<dc:#2>^^J%
823 -----<rdf:#3>^^J%
824         }%
825     \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`.

```

826         \hyxmp@xmlify{#4}%
827         \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
828         \def\@elt##1{%
829             \hyxmp@add@to@xml{%

```

```

830 -----<rdf:li>##1</rdf:li>^^J%
831     }%
832   }%
833   \hyxmp@list
834   \egroup
835   \hyxmp@add@to@xml{%
836 -----</rdf:#3>^^J%
837 -----</dc:#2>^^J%
838   }%
839   \fi
840 }

```

`\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.

```

841 \newcommand*{\hyxmp@dc@schema}{%
842   \hyxmp@add@to@xml{%
843 -----<rdf:Description rdf:about=""^^J%
844 -----xmlns:dc="http://purl.org/dc/elements/1.1/"^^J%
845 -----<dc:format>application/pdf</dc:format>^^J%
846   }%
847   \hyxmp@rdf@dc[\ifHy@pdfa]{title}{\@pdftitle}%
848   \hyxmp@rdf@dc[\ifHy@pdfa]{description}{\@pdfsubject}%
849   \hyxmp@rdf@dc{rights}{\@pdfcopyright}%
850   \hyxmp@list@to@xml[\ifHy@pdfa]{creator}{Seq}{\hyxmp@pdfauthor}%
851   \hyxmp@list@to@xml{subject}{Bag}{\hyxmp@pdfkeywords}%
852   \hyxmp@list@to@xml{date}{Seq}{\hyxmp@today}%
853   \hyxmp@list@to@xml{language}{Bag}{\@pdflang}%
854   \hyxmp@list@to@xml{type}{Bag}{\@pdftype}%
855   \ifx\@pdfsource\@empty
856   \else
857     \hyxmp@add@simple{dc:source}{\@pdfsource}%
858   \fi
859   \hyxmp@add@to@xml{%
860 -----</rdf:Description>^^J%
861   }%
862 }

```

### 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.

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

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

```
864 \let\hyxmp@rights=\@empty
865 \ifx\@pdflicenseurl\@empty
866 \else
867 \def\hyxmp@rights{YES}%
868 \fi
869 \ifx\@pdfcopyright\@empty
870 \else
871 \def\hyxmp@rights{YES}%
872 \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.

```
873 \ifx\hyxmp@rights\@empty
874 \else
```

Header

```
875 \hyxmp@add@to@xml{%
876 -----<rdf:Description rdf:about="^^J%
877 -----xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/">^^J%
878 }%
```

Copyright indication

```
879 \ifx\@pdfcopyright\@empty
880 \else
881 \hyxmp@add@to@xml{%
882 -----<xmpRights:Marked>True</xmpRights:Marked>^^J%
883 }%
884 \fi
```

License URL

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

Trailer

```
886 \hyxmp@add@to@xml{%
887 -----</rdf:Description>^^J%
888 }%
889 \fi
890 }
```

### 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.

```

891 \gdef\hyxmp@mm@schema{%
892   \hyxmp@def@DocumentID
893   \hyxmp@def@InstanceID
894   \hyxmp@add@to@xml{%
895     <rdf:Description rdf:about=""^^J%
896     _____xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/">^^J%
897     <xmpMM:DocumentID>\hyxmp@DocumentID</xmpMM:DocumentID>^^J%
898     <xmpMM:InstanceID>\hyxmp@InstanceID</xmpMM:InstanceID>^^J%
899     </rdf:Description>^^J%
900   }%
901 }

```

### 3.5.6 The XMP Basic schema

`\hyxmp@define@createdate` Define `\hyxmp@createdate` as the document's creation date but in XMP date format, not PDF date format. We use `\hyxmp@createdate` for the `CreateDate`, `ModifyDate`, and `MetadataDate` fields.

```

902 \newcommand*{\hyxmp@define@createdate}{%
903   \@ifundefined{pdffeedback}{%
904     \@ifundefined{pdfcreationdate}{%

```

Case 1: Neither `\pdffeedback` nor `\pdfcreationdate` is defined (X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X and regular L<sup>A</sup>T<sub>E</sub>X).

```

905     \hyxmp@today@define\hyxmp@createdate
906   }{%

```

Case 2: `\pdfcreationdate` is defined (pdfL<sup>A</sup>T<sub>E</sub>X and pre-0.85 LuaL<sup>A</sup>T<sub>E</sub>X).

```

907     \edef\hyxmp@createdate{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
908   }%
909 }{%

```

Case 3: `\pdffeedback` is defined (LuaL<sup>A</sup>T<sub>E</sub>X 0.85+).

```

910     \edef\hyxmp@createdate{\expandafter\hyxmp@pdf@to@xmp@date\pdffeedback creationdate}%
911   }%
912 }

```

`\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`.

```

913 \newcommand*{\hyxmp@xmp@basic@schema}{%
914   \hyxmp@add@to@xml{%
915     <rdf:Description rdf:about=""^^J%
916     _____xmlns:xmp="http://ns.adobe.com/xap/1.0/">^^J%
917   }%
918   \hyxmp@define@createdate

```

For the document's creation date, use the user-specified `\@pdfcreationdate` if defined and non-empty. Otherwise use our fabricated `\hyxmp@createdate`.

```

919   \@ifundefined{@pdfcreationdate}{%
920     \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@createdate}%

```

```

921 }{%
922   \ifx\@pdfcreationdate\@empty
923     \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@createdate}%
924   \else
925     \hyxmp@add@simple{xmp:CreateDate}{%
926       \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfcreationdate}}%
927   \fi
928 }%

```

For the document's modification date, use the user-specified \@pdfmoddate if defined and non-empty. Otherwise use our fabricated \hyxmp@createdate.

```

929 \@ifundefined{@pdfmoddate}{%
930   \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@createdate}%
931 }{%
932   \ifx\@pdfmoddate\@empty
933     \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@createdate}%
934   \else
935     \hyxmp@add@simple{xmp:ModifyDate}{%
936       \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfmoddate}}%
937   \fi
938 }%

```

For the document's metadata date, use the user-specified \@pdfmetadatettime if defined and non-empty. Otherwise use our fabricated \hyxmp@createdate.

```

939 \ifx\@pdfmetadatettime\@empty
940   \hyxmp@add@simple{xmp:MetadataDate}{\hyxmp@createdate}%
941 \else
942   \hyxmp@add@simple{xmp:MetadataDate}{\@pdfmetadatettime}%
943 \fi

```

Define the creation tool and the base URL.

```

944 \hyxmp@add@simple{xmp:CreatorTool}{\@pdfcreator}%
945 \hyxmp@add@simple{xmp:BaseURL}{\@baseurl}%
946 \hyxmp@add@to@xml{%
947 -----</rdf:Description>^^J%
948 }%
949 }

```

### 3.5.7 The Photoshop schema

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

```

950 \gdef\hyxmp@photoshop@schema{%
951   \edef\hyxmp@photoshop@data{\@pdfauthortitle\@pdfcaptionwriter}%
952   \ifx\hyxmp@photoshop@data\@empty
953     \else
954       \hyxmp@add@to@xml{%
955 -----<rdf:Description rdf:about=""^^J%
956 -----_xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/">^^J%

```

```

957     }%
958   \fi
959   \hyxmp@add@simple{photoshop:AuthorsPosition}{\@pdfauthortitle}%
960   \hyxmp@add@simple{photoshop:CaptionWriter}{\@pdfcaptionwriter}%
961   \ifx\hyxmp@photoshop@data\@empty
962   \else
963     \hyxmp@add@to@xml{%
964     -----</rdf:Description>^^J%
965     }%
966   \fi
967 }

```

### 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.

```

968 \begingroup
969   \catcode'\&=12
970   \catcode'\#=12
971   \gdef\xmplinesep{&#xA;}
972 \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.

```

973 \newcommand*{\hyxmp@list@to@lines}[2]{%
974   \ifnotmtargexp{#2}{%
975     \bgroup
976     \hyxmp@add@to@xml{%
977     -----<#1>%
978     }%

```

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

```

979     \def\@elt@first##1{%
980       \hyxmp@add@to@xml{##1}%
981       \let\@elt=\@elt@rest
982     }%

```

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

```

983     \def\@elt@rest##1{%
984       \hyxmp@add@to@xml{\xmplinesep##1}%
985     }%

```

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

```

986     \let\@elt=\@elt@first
987     \hyxmp@xmllify{#2}%

```

```

988     \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmllified}%
989     \hyxmp@list
990     \hyxmp@add@to+xml{</#1>^^J}%
991   \egroup
992 }%
993 }

\hyxmp@photometa@schema Add properties defined by the IPTC Photo Metadata schema [7] to the
\hyxmp@photometa@data \hyxmp+xml macro. We currently support only the contact-information
                        details structure, viz. the Iptc4xmpCore:CreatorContactInfo/CiAdrExtadr,
                        Iptc4xmpCore:CreatorContactInfo/CiAdrCity, Iptc4xmpCore:CreatorContactInfo/
                        CiAdrRegion, Iptc4xmpCore:CreatorContactInfo/CiAdrPcode,
                        Iptc4xmpCore:CreatorContactInfo/CiAdrCtry, Iptc4xmpCore:CreatorContactInfo/
                        CiTelWork, Iptc4xmpCore:CreatorContactInfo/CiEmailWork, and
                        Iptc4xmpCore:CreatorContactInfo/CiUrlWork properties.
994 \gdef\hyxmp@photometa@schema{%
995   \edef\hyxmp@photometa@data{%
996     \@pdfcontactaddress
997     \@pdfcontactcity
998     \@pdfcontactregion
999     \@pdfcontactpostcode
1000    \@pdfcontactcountry
1001    \@pdfcontactphone
1002    \@pdfcontactemail
1003    \@pdfcontacturl
1004  }%
1005  \ifx\hyxmp@photometa@data\@empty
1006  \else
1007    \hyxmp@iptc@extensions
1008    \hyxmp@add@to+xml{%
1009      -----<rdf:Description rdf:about="^^J%
1010      -----xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/">^^J%
1011      -----<Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">^^J%
1012        }%
1013    \fi
1014    \hyxmp@list@to@lines{Iptc4xmpCore:CiAdrExtadr}{\@pdfcontactaddress}%
1015    \hyxmp@add@simple{Iptc4xmpCore:CiAdrCity}{\@pdfcontactcity}%
1016    \hyxmp@add@simple{Iptc4xmpCore:CiAdrRegion}{\@pdfcontactregion}%
1017    \hyxmp@add@simple{Iptc4xmpCore:CiAdrPcode}{\@pdfcontactpostcode}%
1018    \hyxmp@add@simple{Iptc4xmpCore:CiAdrCtry}{\@pdfcontactcountry}%

```

\xmplinesep The IPTC standard states that sets of telephone numbers, email addresses, and URLs for the contact person or institution, “[m]ay have to be separated by a comma in the user interface” [7]. 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.

```

1019 \bgroup
1020   \def\xmplinesep{,}%
1021   \hyxmp@list@to@lines{Iptc4xmpCore: CiTelWork}{\@pdfcontactphone}%
1022   \hyxmp@list@to@lines{Iptc4xmpCore: CiEmailWork}{\@pdfcontactemail}%
1023   \hyxmp@list@to@lines{Iptc4xmpCore: CiUrlWork}{\@pdfcontacturl}%
1024 \egroup
1025 \ifx\hyxmp@photometa@data\@empty
1026 \else
1027   \hyxmp@add@to@xml{%
1028 -----</Iptc4xmpCore:CreatorContactInfo>^^J%
1029 -----</rdf:Description>^^J%
1030   }%
1031 \fi
1032 }

```

\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 [11], 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.

```

1033 \newcommand*{\hyxmp@iptc@extensions}{%
1034   \hyxmp@add@to@xml{%
1035 -----<rdf:Description rdf:about=""^^J%
1036 -----xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"^^J%
1037 -----xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema\hyxmp@hash"^^J%
1038 -----xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property\hyxmp@hash"^^J%
1039 -----xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type\hyxmp@hash"^^J%
1040 -----xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field\hyxmp@hash">^^J%
1041 -----<pdfaExtension:schemas>^^J%
1042 -----<rdf:Bag>^^J%
1043 -----<rdf:li rdf:parseType="Resource">^^J%
1044 -----<pdfaSchema:schema>IPTC Core Schema</pdfaSchema:schema>^^J%
1045 -----<pdfaSchema:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaSchema:
1046 -----<pdfaSchema:prefix>Iptc4xmpCore</pdfaSchema:prefix>^^J%
1047 -----<pdfaSchema:property>^^J%
1048 -----<rdf:Seq>^^J%
1049 -----<rdf:li rdf:parseType="Resource">^^J%
1050 -----<pdfaProperty:name>CreatorContactInfo</pdfaProperty:name>^^J%
1051 -----<pdfaProperty:valueType>contactinfo</pdfaProperty:valueType>^^J%
1052 -----<pdfaProperty:category>external</pdfaProperty:category>^^J%
1053 -----<pdfaProperty:description>contact information for the document's creator</p
1054 -----</rdf:li>^^J%
1055 -----</rdf:Seq>^^J%
1056 -----</pdfaSchema:property>^^J%
1057 -----<pdfaSchema:valueType>^^J%
1058 -----<rdf:Seq>^^J%
1059 -----<rdf:li rdf:parseType="Resource">^^J%

```



```

1060 -----<pdfaType:type>contactinfo</pdfaType:type>^^J%
1061 -----<pdfaType:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaType:
1062 -----<pdfaType:prefix>Iptc4xmpCore</pdfaType:prefix>^^J%
1063 -----<pdfaType:description>contact information</pdfaType:description>^^J%
1064 -----<pdfaType:field>^^J%
1065 -----<rdf:Seq>^^J%
1066 }%

1067 \hyxmp@text@resource{CiAdrExtadr}{contact address}%
1068 \hyxmp@text@resource{CiAdrCity}{contact city}%
1069 \hyxmp@text@resource{CiAdrRegion}{contact region}%
1070 \hyxmp@text@resource{CiAdrPcode}{contact postal code}%
1071 \hyxmp@text@resource{CiAdrCtry}{contact country}%
1072 \hyxmp@text@resource{CiTelWork}{contact telephone number}%
1073 \hyxmp@text@resource{CiEmailWork}{contact email address}%
1074 \hyxmp@text@resource{CiUrlWork}{contact url}%

1075 \hyxmp@add@to@xml{%
1076 -----</rdf:Seq>^^J%
1077 -----</pdfaType:field>^^J%
1078 -----</rdf:li>^^J%
1079 -----</rdf:Seq>^^J%
1080 -----</pdfaSchema:valueType>^^J%
1081 -----</rdf:li>^^J%
1082 -----</rdf:Bag>^^J%
1083 -----</pdfaExtension:schemas>^^J%
1084 -----</rdf:Description>^^J%
1085 }%
1086 }

```

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

```

1087 \newcommand*{\hyxmp@text@resource}[2]{%
1088   \hyxmp@add@to@xml{%
1089 -----<rdf:li rdf:parseType="Resource">^^J%
1090 -----<pdfaField:name>#1</pdfaField:name>^^J%
1091 -----<pdfaField:valueType>Text</pdfaField:valueType>^^J%
1092 -----<pdfaField:description>#2</pdfaField:description>^^J%
1093 -----</rdf:li>^^J%
1094   }
1095 }

```

### 3.5.9 The PDF/A Identification schema

`\hyxmp@pdfa@id@schema` Add properties defined by the PDF/A Identification schema [10] to the `\hyxmp@xml` macro. These properties identify a document as conforming to a particular PDF/A standard. We default to PDF/A-1b if any PDF/A compliance is detected but let the author override the “1” with `pdfapart` and the “B” with `pdfaconformance`.

```

1096 \newcommand*{\hyxmp@pdfa@id@schema}{%
1097   \ifHy@pdfa
1098     \hyxmp@add@to@xml{%

```

```

1099 -----<rdf:Description rdf:about=""^^J%
1100 -----xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/">^^J%
1101     }%
1102     \hyxmp@add@simple{pdfaid:part}{\@pdfapart}%
1103     \hyxmp@add@simple{pdfaid:conformance}{\@pdfaconformance}%
1104     \hyxmp@add@to@xml{%
1105 -----</rdf:Description>^^J%
1106     }%
1107     \fi
1108 }

```

### 3.5.10 Combining schemata into an XMP packet

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

```

1109 \begingroup
1110   \ifhyxmp@unicodetex
1111     \lccode'\!="FEFF %
1112     \lowercase{%
1113       \gdef\hyxmp@bom{!}
1114     }%
1115   \else
1116     \catcode'\^^ef=12
1117     \catcode'\^^bb=12
1118     \catcode'\^^bf=12
1119     \gdef\hyxmp@bom{^^ef^^bb^^bf}%
1120   \fi
1121 \endgroup

```

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

```

1122 \def\hyxmp@construct@packet{%
1123   \gdef\hyxmp@xml{%
1124     \hyxmp@add@to@xml{<?xpacket begin="\hyxmp@bom" %
1125 id="W5M0MpCehiHzreSzNTczkc9d"?>^^J%
1126 <x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="3.1-702">^^J%
1127 ___<rdf:RDF
1128 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\hyxmp@hash">^^J%
1129   }%
1130   \hyxmp@pdf@schema
1131   \hyxmp@xmpRights@schema
1132   \hyxmp@dc@schema
1133   \hyxmp@photoshop@schema
1134   \hyxmp@photometa@schema
1135   \hyxmp@xmp@basic@schema
1136   \hyxmp@pdfa@id@schema
1137   \hyxmp@mm@schema
1138   \hyxmp@add@to@xml{%
1139 ___</rdf:RDF>^^J%
1140 </x:xmpmeta>^^J%

```

```

1141 \hyxmp@padding
1142 <?xpacket end="w"?>^^J%
1143 }%
1144 }

```

### 3.6 Embedding the XMP packet

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

```

\hyxmp@embed@packet Determine which hyperref driver is in use and invoke the appropriate embedding
\hyxmp@driver function.
1145 \newcommand*{\hyxmp@embed@packet}{%
1146   \hyxmp@construct@packet
1147   \def\hyxmp@driver{hpdftex}%
1148   \ifx\hyxmp@driver\Hy@driver
1149     \hyxmp@embed@packet@pdftex
1150   \else
1151     \def\hyxmp@driver{hluatex}%
1152     \ifx\hyxmp@driver\Hy@driver
1153       \hyxmp@embed@packet@luatex
1154     \else
1155       \def\hyxmp@driver{hdvipdfm}%
1156       \ifx\hyxmp@driver\Hy@driver
1157         \hyxmp@embed@packet@dvipdfm
1158       \else
1159         \def\hyxmp@driver{hxetex}%
1160         \ifx\hyxmp@driver\Hy@driver
1161           \hyxmp@embed@packet@xetex
1162         \else
1163           \@ifundefined{pdfmark}{%
1164             \PackageWarningNoLine{hyperxmp}{%
1165               Unrecognized hyperref driver ‘\Hy@driver’. \MessageBreak
1166               \jobname.tex’s XMP metadata will *not* be \MessageBreak
1167               embedded in the resulting file}%
1168           }{%
1169             \hyxmp@embed@packet@pdfmark
1170           }%
1171         \fi
1172       \fi
1173     \fi
1174   \fi
1175 }

```

#### 3.6.1 Embedding using pdfTeX

Up to version 0.85, LuaTeX supported the pdfTeX primitives, and hyperref didn’t distinguish the two backends. However, from hyperxmp’s perspective there is one

key difference: the effect of `\pdfcompresslevel` is local to a group in pdfTeX but is global in LuaTeX.

The PDF object representing the XMP packet is supposed to include an uncompressed stream so it can be read by non-PDF-aware tools. However, we don't want to unnecessarily uncompress *every* PDF stream. The solution, provided by Hans Hagen on the `luatex` mailing list (thread: "Leaving a single PDF object uncompressed", 6 JUL 2016), is to provide the `uncompressed` flag to `\pdfobj`. Our definition of `\hyxmp@embed@packet@pdftex` uses the `ifluatex` package to distinguish the pdfTeX case from the pre-0.85 LuaTeX case.

```
1176 \RequirePackage{ifluatex}
```

`\hyxmp@embed@packet@pdftex` Embed the XMP packet using pdfTeX primitives, which are supported by both pdfTeX and pre-0.85 LuaTeX. The only difference is that in the former case we locally specify `\pdfcompresslevel=0` to leave the PDF object uncompressed while in the latter case we pass the `uncompressed` flag to `\pdfobj` to achieve the same effect.

```
1177 \newcommand*{\hyxmp@embed@packet@pdftex}{%
1178   \bgroup
1179   \ifluatex
1180   \else
1181     \pdfcompresslevel=0
1182   \fi
1183   \immediate\pdfobj \ifluatex uncompressed\fi stream attr {%
1184     /Type /Metadata
1185     /Subtype /XML
1186   }{\hyxmp@xml}%
1187   \pdfcatalog {/Metadata \the\pdflastobj\space 0 R}%
1188   \egroup
1189 }
```

### 3.6.2 Embedding using LuaTeX 0.85+

`\hyxmp@embed@packet@luatex` Embed the XMP packet using LuaTeX 0.85+ primitives.

```
1190 \newcommand*{\hyxmp@embed@packet@luatex}{%
1191   \immediate\pdfextension obj uncompressed stream attr {%
1192     /Type /Metadata
1193     /Subtype /XML
1194   }{\hyxmp@xml}%
1195   \pdfextension catalog {/Metadata \the\numexpr\pdffeedback lastobj\relax\space 0 R}%
1196 }
```

### 3.6.3 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.

```
1197 \newcommand*{\hyxmp@embed@packet@pdfmark}{%
```

```

1198 \pdfmark{%
1199   pdfmark=/NamespacePush
1200 }%
1201 \pdfmark{%
1202   pdfmark=/OBJ,
1203   Raw={/_objdef \string{hyxmp@Metadata\string} /type /stream}%
1204 }%
1205 \pdfmark{%
1206   pdfmark=/PUT,
1207   Raw={\string{hyxmp@Metadata\string}
1208     2 dict begin
1209       /Type /Metadata def
1210       /Subtype /XML def
1211       currentdict
1212     end
1213   }%
1214 }%
1215 \pdfmark{%
1216   pdfmark=/PUT,
1217   Raw={\string{hyxmp@Metadata\string} (\hyxmp@xml)}%
1218 }%
1219 \pdfmark{%
1220   pdfmark=/Metadata,
1221   Raw={\string{Catalog\string} \string{hyxmp@Metadata\string}}%
1222 }%
1223 \pdfmark{%
1224   pdfmark=/NamespacePop
1225 }%
1226 }

```

### 3.6.4 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.

```

1227 \newcommand*{\hyxmp@embed@packet@dvipdfm}{%
1228   \hyxmp@stringlen{\hyxmp@xml}%
1229   \special{pdf: object @hyxmp@Metadata
1230     <<
1231       /Type /Metadata
1232       /Subtype /XML
1233       /Length \the\@tempcnta
1234     >>
1235     stream^^J\hyxmp@xml endstream%
1236   }%
1237   \special{pdf: docview
1238     <<
1239       /Metadata @hyxmp@Metadata
1240     >>

```

```

1241 }%
1242 }

```

`\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.

```

1243 \newcommand*{\hyxmp@string@len}[1]{%
1244   \@tempcnta=0
1245   \expandafter\hyxmp@count@spaces#1 {} %
1246   \expandafter\hyxmp@count@non@spaces#1{}%
1247 }

```

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

```

1248 \def\hyxmp@count@spaces#1 {%
1249   \def\hyxmp@one@token{#1}%
1250   \ifx\hyxmp@one@token\empty
1251     \advance\@tempcnta by -1
1252   \else
1253     \advance\@tempcnta by 1
1254     \expandafter\hyxmp@count@spaces
1255   \fi
1256 }

```

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

```

1257 \newcommand*{\hyxmp@count@non@spaces}[1]{%
1258   \def\hyxmp@one@token{#1}%
1259   \ifx\hyxmp@one@token\empty
1260   \else
1261     \advance\@tempcnta by 1
1262     \expandafter\hyxmp@count@non@spaces
1263   \fi
1264 }

```

### 3.6.5 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.

```

1265 \newcommand*{\hyxmp@embed@packet@xetex}{%
1266   \special{pdf:stream @hyxmp@Metadata (\hyxmp+xml)
1267     <<
1268     /Type /Metadata

```

```

1269      /Subtype /XML
1270    >>
1271  }%
1272  \special{pdf:put @catalog
1273    <<
1274      /Metadata @hyxmp@Metadata
1275    >>
1276  }%
1277 }

```

### 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.

```

1278 \catcode'\="=\hyxmp@dq@code

```

## 4 Future Work

**Help wanted** Ideally, `\xmpquote` should automatically replace all commas with `\xmpcomma`. Unfortunately, my  $\text{\TeX}$  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  $\text{\TeX}$  engines (pdf $\text{\TeX}$ , Lua $\text{\TeX}$ , Xe $\text{\TeX}$ , 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  $\text{\LaTeX}$  document presented on pages 6–7. 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:li xml:lang="de">
        Über einen die Erzeugung und Verwandlung des Lichtes
        betreffenden heuristischen Gesichtspunkt
      </rdf:li>
    </rdf:Alt>
  </dc:title>
  <dc:description>
    <rdf:Alt>
      <rdf:li xml:lang="en">photoelectric effect</rdf:li>
      <rdf:li xml:lang="x-default">photoelectric effect</rdf:li>
    </rdf:Alt>
  </dc:description>
  <dc:rights>
    <rdf:Alt>
      <rdf:li xml:lang="en">
        Copyright (C) 1905, Albert Einstein
      </rdf:li>
      <rdf:li xml:lang="x-default">
        Copyright (C) 1905, Albert Einstein
      </rdf:li>
    </rdf:Alt>
  </dc:rights>
  <dc:creator>
    <rdf:Seq>
      <rdf:li>Albert Einstein</rdf:li>
    </rdf:Seq>
  </dc:creator>
  <dc:subject>
    <rdf:Bag>

```



```

        <rdf:li>energy quanta</rdf:li>
        <rdf:li>Hertz effect</rdf:li>
        <rdf:li>quantum physics</rdf:li>
    </rdf:Bag>
</dc:subject>
<dc:date>
    <rdf:Seq>
        <rdf:li>1905-03-17</rdf:li>
    </rdf:Seq>
</dc:date>
<dc:language>
    <rdf:Bag>
        <rdf:li>en</rdf:li>
    </rdf:Bag>
</dc:language>
<dc:type>
    <rdf:Bag>
        <rdf:li>Text</rdf:li>
    </rdf:Bag>
</dc:type>
<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:prefix>Iptc4xmpCore</pdfaType:prefix>
            <pdfaType:description>contact information</pdfaType:description>
            <pdfaType:field>
                <rdf:Seq>
                    <rdf:li rdf:parseType="Resource">
                        <pdfaField:name>CiAdrExtadr</pdfaField:name>
                        <pdfaField:valueType>Text</pdfaField:valueType>
                        <pdfaField:description>contact address</pdfaField:description>
                    </rdf:li>
                    <rdf:li rdf:parseType="Resource">
                        <pdfaField:name>CiAdrCity</pdfaField:name>
                        <pdfaField:valueType>Text</pdfaField:valueType>
                        <pdfaField:description>contact city</pdfaField:description>
                    </rdf:li>
                    <rdf:li rdf:parseType="Resource">
                        <pdfaField:name>CiAdrRegion</pdfaField:name>
                        <pdfaField:valueType>Text</pdfaField:valueType>
                        <pdfaField:description>contact region</pdfaField:description>
                    </rdf:li>
                    <rdf:li rdf:parseType="Resource">
                        <pdfaField:name>CiAdrPcode</pdfaField:name>
                        <pdfaField:valueType>Text</pdfaField:valueType>
                        <pdfaField:description>contact postal code</pdfaField:description>
                    </rdf:li>
                    <rdf:li rdf:parseType="Resource">
                        <pdfaField:name>CiAdrCtry</pdfaField:name>
                        <pdfaField:valueType>Text</pdfaField:valueType>
                        <pdfaField:description>contact country</pdfaField:description>
                    </rdf:li>
                    <rdf:li rdf:parseType="Resource">
                        <pdfaField:name>CiTelWork</pdfaField:name>
                        <pdfaField:valueType>Text</pdfaField:valueType>
                        <pdfaField:description>contact telephone number</pdfaField:description>
                    </rdf:li>
                    <rdf:li rdf:parseType="Resource">
                        <pdfaField:name>CiEmailWork</pdfaField:name>

```

```

        <pdfaField:valueType>Text</pdfaField:valueType>
        <pdfaField:description>contact email address</pdfaField:description>
      </rdf:li>
      <rdf:li rdf:parseType="Resource">
        <pdfaField:name>CiUrlWork</pdfaField:name>
        <pdfaField:valueType>Text</pdfaField:valueType>
        <pdfaField:description>contact url</pdfaField:description>
      </rdf:li>
    </rdf:Seq>
  </pdfaType:field>
</rdf:li>
</rdf:Seq>
</pdfaSchema:valueType>
</rdf:li>
</rdf:Bag>
</pdfaExtension:schemas>
</rdf:Description>
<rdf:Description rdf:about=""
  xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/">
  <Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">
    <Iptc4xmpCore:CiAdrExtadr>Kramgasse 49</Iptc4xmpCore:CiAdrExtadr>
    <Iptc4xmpCore:CiAdrCity>Bern</Iptc4xmpCore:CiAdrCity>
    <Iptc4xmpCore:CiAdrPcode>3011</Iptc4xmpCore:CiAdrPcode>
    <Iptc4xmpCore:CiAdrCtry>Switzerland</Iptc4xmpCore:CiAdrCtry>
    <Iptc4xmpCore:CiTelWork>031 312 00 91</Iptc4xmpCore:CiTelWork>
    <Iptc4xmpCore:CiEmailWork>aeinstein@ipi.ch</Iptc4xmpCore:CiEmailWork>
    <Iptc4xmpCore:CiUrlWork>
      <a href="http://einstein.biz/">http://einstein.biz/</a>,
      <a href="https://www.facebook.com/AlbertEinstein">https://www.facebook.com/AlbertEinstein</a>
    </Iptc4xmpCore:CiUrlWork>
  </Iptc4xmpCore:CreatorContactInfo>
</rdf:Description>
<rdf:Description rdf:about=""
  xmlns:xmp="http://ns.adobe.com/xap/1.0/">
  <xmp:CreateDate>2017-11-26T22:54:45-07:00</xmp:CreateDate>
  <xmp:ModifyDate>2017-11-26T22:54:45-07:00</xmp:ModifyDate>
  <xmp:MetadataDate>2017-11-26T22:54:45-07:00</xmp:MetadataDate>
  <xmp:CreatorTool>LaTeX with hyperref package</xmp:CreatorTool>
  <xmp:BaseURL>
    <a href="http://mirror.ctan.org/macros/latex/contrib/hyperxmp/">http://mirror.ctan.org/macros/latex/contrib/hyperxmp/</a>
  </xmp:BaseURL>
</rdf:Description>
<rdf:Description rdf:about=""
  xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/">
  <xmpMM:DocumentID>
    <a href="http://www.adobe.com/uuid:0595fdce-41dc-e4c4-6c418dc4ce46">uuid:0595fdce-41dc-e4c4-6c418dc4ce46</a>
  </xmpMM:DocumentID>

```

```

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

```

## References

- [1] Adobe Systems, Inc., San Jose, California. *Adobe Acrobat X SDK Help, pdfmark Reference*. Available from <http://www.adobe.com/devnet/acrobat/documentation.html>.
- [2] Adobe Systems, Inc. *PostScript Language Reference Manual*. Addison-Wesley, 2nd edition, January 1996, ISBN: 0-201-18127-4.
- [3] Adobe Systems, Inc., San Jose, California. *Document Management—Portable Document Format—Part 1: PDF 1.7*, July 2008. ISO 32000-1 standard document. Available from [http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000\\_2008.pdf](http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000_2008.pdf).
- [4] Adobe Systems, Inc., San Jose, California. *XMP Specification Part 1: Data model, Serialization, and Core Properties*, April 2012. Available from <http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/xmp/pdfs/cc-201306/XMPSpecificationPart1.pdf>.
- [5] DCMI Usage Board *DCMI Metadata Terms*, June 14, 2012. Available from <http://dublincore.org/documents/dcmi-terms/>.
- [6] 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>.
- [7] 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).
- [8] Internet Assigned Numbers Authority. Language subtag registry, January 11, 2011. Available from <http://www.iana.org/assignments/language-subtag-registry>.
- [9] Paul J. Leach, Michael Mealling, and Rich Salz. A Universally Unique IDentifier (UUID) URN namespace. Request for Comments 4122, Internet Engineering Task Force, Network Working Group, July 2005. Category: Standards Track. Available from <http://www.ietf.org/rfc/rfc4122.txt>.

- [10] PDF/A Competence Center, Berlin, Germany. *TechNote 0008: Pre-defined XMP Properties in PDF/A-1*, March 20, 2008. Available from [http://www.pdfa.org/wp-content/uploads/2011/08/tn0008\\_predefined\\_xmp\\_properties\\_in\\_pdfa-1\\_2008-03-20.pdf](http://www.pdfa.org/wp-content/uploads/2011/08/tn0008_predefined_xmp_properties_in_pdfa-1_2008-03-20.pdf).
- [11] 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).
- [12] Misha Wolf and Charles Wicksteed. Date and time formats. Note NOTE-datetime, World Wide Web Consortium (W3C), September 15, 1997. Available from <http://www.w3.org/TR/NOTE-datetime>.

## Change History

v1.0		wrote the document's metadata	19
	General: Initial version	.....	1 v1.4
v1.1	<code>\hyxmp@construct@packet:</code>		
	Explicitly set the category		
	codes of characters $\langle EF \rangle$ , $\langle BB \rangle$ ,	<code>\hyxmp@rdf@dc:</code> Included metadata	
	and $\langle BF \rangle$ to "letter". Thanks	in the <code>x-default</code> language	
	to Daniel Schömer for the bug	regardless of the specified	
	report	metadata language	40
	.....		50
v1.2	<code>\hyxmp@xmpRights@schema:</code>		
	General: Added support for the	Renamed the <code>xapRights</code>	
	<code>X<sub>Y</sub>T<sub>E</sub>X</code> backend ( <code>xdvipdfmx</code> )	namespace prefix to <code>xmpRights</code>	43
	.....		
	Added support for the	v1.5	
	Photoshop schema	.....	1
	Made the package compatible	General: Made the XMP inclusion	
	with <code>ngerman</code> . Thanks to	more robust. Thanks to Heiko	
	Tobias Mueller for the bug	Oberdiek for the bug report	
	report.	and suggested modifications.	12
	.....		12 v2.0
v1.3	<code>\hyxmp@reencode:</code> Introduced this	<code>\ProcessKeyvalOptions:</code> Added	
	macro to re-encode Unicode	this macro	18
	strings as 8-bit strings before	<code>\XMPTruncateList:</code> Added this	
	manipulating them into XMP	macro	22
	schema. This change addresses	<code>\hyxmp@ProcessKeyvalOptions:</code>	
	a bug reported by Martin	Added this macro	18
	Münch	<code>\hyxmp@SpaceOther:</code> Added by	
	.....	Heiko Oberdiek	32
	28	<code>\hyxmp@add@to+xml:</code> Updated also	
	General: Introduced the	to replace commas	37
	<code>pdfmetalang</code> package option,	<code>\hyxmp@bom:</code> Added by Heiko	
	which enables an author to	Oberdiek	50
	specify the language in which he		

\hyxmp@comma: Added this macro .	21	\hyxmp@zero: Added by Heiko Oberdiek . . . . .	33
\hyxmp@construct@packet: Modified by Heiko Oberdiek to use an appropriate BOM representation via \hyxmp@bom	50	\ifhyxmp@unicodetex: Added by Heiko Oberdiek . . . . .	28
\hyxmp@crap@convert: Added by Heiko Oberdiek . . . . .	32	\xmpcomma: Added this macro . . . .	21
\hyxmp@crap@test: Added by Heiko Oberdiek . . . . .	32	\xmpquote: Added this macro . . .	22
\hyxmp@dc@schema: Added support for dc:language and dc:source .	42	General: Added support for the XMP Basic schema and miscellaneous other bits of metadata . . . . .	1
\hyxmp@is@unicode: Added by Heiko Oberdiek . . . . .	29	Heiko Oberdiek's major rewrite of the code to better support native-Unicode T <sub>E</sub> X implementations (X <sub>Y</sub> T <sub>E</sub> X and LuaT <sub>E</sub> X) . . . . .	1
\hyxmp@list@to@xml: Modified by Heiko Oberdiek to use the new Unicode-processing macros . .	41	New \AtBeginDocument code from Heiko Oberdiek to properly encode \@pdfmetalang . . . . .	19
\hyxmp@photoshop@schema: Simplified using \hyxmp@add@simple . . . . .	45	v2.1	
\hyxmp@reencode: Replaced with an empty macro by Heiko Oberdiek . . . . .	28	\hypersetup: Added this macro .	18
\hyxmp@skiptorelax: Added by Heiko Oberdiek . . . . .	32	\hyxmp@hypersetup: Added this macro . . . . .	18
\hyxmp@skipzeros: Added by Heiko Oberdiek . . . . .	31	\hyxmp@redefine@Hyp: Added this macro . . . . .	16
\hyxmp@string: Added this macro	39	General: Enabled hyperxmp and hyperref to be loaded in either order. This addresses a bug report by Yuri Donskoy . . . .	16
\hyxmp@toxml: Added by Heiko Oberdiek . . . . .	30	v2.2	
Escaped parentheses written with pdfmarks to prevent dvips from line-wrapping the XMP packet . . . . .	30	\hyxmp@iptc@extensions: Added this macro to support PDF/A generation . . . . .	48
\hyxmp@toxml@unicodetex: Added by Heiko Oberdiek . . . . .	30	\hyxmp@list@to@lines: Added this macro . . . . .	46
\hyxmp@xetex@crap: Added by Heiko Oberdiek . . . . .	31	\hyxmp@photometa@schema: Added this macro . . . . .	47
\hyxmp@xmlify: Completely rewritten by Heiko Oberdiek to better support Unicode-enabled T <sub>E</sub> X programs . . . . .	28	\hyxmp@text@resource: Added this macro . . . . .	49
\hyxmp@xmp@basic@schema: Added this macro . . . . .	44	\xmpcomma: Changed the default from \relax to an ordinary comma . . . . .	21
\hyxmp@xmpRights@schema: Modified to include xmpRights:Marked only when pdfcopyright is specified and xmpRights:WebStatement only when pdflicenseurl is specified .	43	\xmplinesep: Added this macro .	46
		General: Added support for the IPTC Photo Metadata schema .	1
		v2.3	
		\hyxmp@iptc@extensions: Gave the lptc4xmpCore:CreatorContactInfo fields a unique pdfaType:prefix	

to better support conversion of the document to PDF/A . . . . .	48	\xmptilde: Added this macro . . .	22
v2.3a		General: Added support for the PDF/A Identification schema, as requested by Florian Breitwieser . . . . .	1
General: Bug fix: Redefine \@pdflang as \@empty when hyperref has set it to \relax . .	19	v2.5	
v2.3b		\hyxmp@add@to@xml: Updated also to replace underscores and to modify only the text being added, not the already-modified text . . . . .	37
\XMPTruncateList: Made all definitions local to avoid spurious Too many unprocessed floats errors when running with memoir . .	22	\hyxmp@textunderscore: Added this macro . . . . .	13
v2.4		\hyxmp@uscore: Added this macro	21
\hyxmp@add@simple@var: Added this macro . . . . .	40	General: Enabled “\_” to work within email addresses, as requested by Leonid Sinev . . . .	1
\hyxmp@create@uuid: Modified this macro to produce a proper version 4 (random or pseudorandom) UUID . . . . .	36	v2.6	
\hyxmp@dc@schema: Made dc:language a Bag instead of an individual item so as to conform to the latest XMP specifications, a detail identified by Florian Breitwieser . . . . .	42	General: Added support for a new pdfdate key to explicitly specify the document date (and optionally time) . . . . .	1
\hyxmp@parse@time: Added this macro . . . . .	23	v2.7	
\hyxmp@parse@tz: Added this macro . . . . .	23	General: Automatically use \title and \author if pdftitle and pdfauthor are left unspecified. Thanks to Maciej Radziejewski for the suggestion . . . . .	20
\hyxmp@parse@tz@char: Added this macro . . . . .	23	v2.8	
\hyxmp@pdf@schema: Made \hyxmp@pdf@schema <i>always</i> include the Adobe PDF schema, even when empty. Florian Breitwieser noted that this is necessary for PDF/A-1b compliance . . . . .	39	\hyxmp@add@to@xml: Corrected inadvertent lowercasing of non-Latin characters when run under Xe <sub>La</sub> TeX or Lua <sub>La</sub> TeX (bug reported by Leonid Sinev)	37
\hyxmp@pdf@to@xmp@date: Added this macro . . . . .	23	v2.9	
\hyxmp@pdfa@id@schema: Added this macro . . . . .	49	\hyxmp@pdfa@id@schema: Let the author specify the PDF/A part and conformance IDs, as requested by Leonid Sinev . . .	49
\hyxmp@today: Modified the code to parse the time and timezone from \pdfcreationdate, as proposed by Florian Breitwieser	26	\hyxmp@photometa@schema: Use lptc4xmpCore instead of lptc4ContInfo as the contact-information metadata prefix. Leonid Sinev reports that Acrobat’s PDF/A validator seems to prefer lptc4xmpCore .	47
\hyxmp@today@define: Added this macro . . . . .	25	General: Force inclusion of dc:creator, dc:title, and dc:description—even if empty—when hyperref is loaded	
\hyxmp@xmp@to@pdf@date: Added this macro . . . . .	24		

	with the <code>pdfa</code> option (suggested by Leonid Sinev) . . . . .	1		<code>\hyxmp@today@define</code> : Modified to include hours and minutes . . .	25
	Introduced the <code>pdftype</code> package option, which enables an author to specify the type of document being produced . . . . .	1		<code>\hyxmp@xmp@basic@schema</code> : Honor <code>hyperref</code> 's <code>pdfcreationdate</code> and <code>pdfmoddate</code> options plus a new <code>pdfmetadate</code> option. Leonid Sinev requested this additional control and helped test the resulting <code>hyperxmp</code> code . . . .	44
v3.0	<code>\hyxmp@embed@packet@luatex</code> : Added this macro . . . . .	52	v3.3	<code>\@pdfsource</code> : Added this macro and the corresponding <code>pdfsource</code> option, at Niklas Beisert's request . . . . .	15
	<code>\hyxmp@today@define</code> : Modified to accept the name of a macro to define . . . . .	25		<code>\XMPLangAlt</code> : Added this macro based on a request—and some code—by Niklas Beisert to support metadata expressed in multiple languages . . . . .	34
	<code>\hyxmp@xmp@basic@schema</code> : Made the XMP <code>CreateDate</code> , <code>ModifyDate</code> , and <code>MetadataDate</code> match the PDF <code>CreationDate</code> .	44		<code>\hyxmp@rdf@dc</code> : Bug fix: Output the metadata language as correct XML even when <code>hyperref</code> is loaded with the <code>unicode</code> option . . . . .	40
	General: Made the code compatible with LuaTeX 0.85. Thanks to Robert Schlicht, Leonid Sinev, and David Carlisle for bug reports and to Leonid Sinev for helping test the new <code>hyperxmp</code> code . . . . .	1		General: Don't overwrite an existing <code>pdfmetalang</code> with <code>pdflang</code> or <code>x-default</code> . This addresses a bug report by Niklas Beisert . . . . .	19
v3.1	<code>\hyxmp@embed@packet@luatex</code> : Updated to use <code>\pdfextension</code> <code>obj</code> <code>uncompressed</code> as suggested by Hans Hagen . . . . .	52	v3.4	<code>\hyxmp@seed@string</code> : Correctly handle an author field of all spaces. Bug reported by Gaëtan Leurent . . . . .	37
	<code>\hyxmp@embed@packet@pdftex</code> : Leave the XMP packet—and only the XMP packet—uncompressed in both pdfTeX and pre-0.85 LuaTeX .	52		General: Use <code>ifmtarg</code> to test for empty arguments, including non-empty but all spaces . . . .	1
v3.2	<code>\hyxmp@as@pdf@date</code> : Added this macro . . . . .	24			
	<code>\hyxmp@as@xmp@date</code> : Added this macro . . . . .	22			



# 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		
\# . . . . .	726, 970	\@pdfcontactregion . . . . . 74, 150, 998, 1016
\& . . . . .	459, 491, 969	\@pdfcontacturl . . . . . 84, 151, 1003, 1023
\, . . . . .	719	\@pdfcopyright . . . . . 46, 152, 849, 869, 879
\@author . . . . .	196, 197	\@pdfcreationdate . . . . . 153, 922, 926
\@baseurl . . . . .	140, 945	\@pdfcreator . . . . . 944
\@elt . . . . .	243, 826, 981, 986	\@pdfdatetime . . . . . 24, 154, 185, 187
\@elt@first . . . . .	979	\@pdfkeywords . . . . . 116, 155
\@elt@rest . . . . .	981, 983	\@pdflang . . . . . 156, 174, 175, 178, 181, 853
\@firstofone . . . . .	538	\@pdflicenseurl . . . . . 50, 157, 865, 885
\@firstoftwo . . . . .	450	\@pdfmetadatetitle . . . . . 35, 158, 939, 942
\@gobble . . . . .	248, 535	\@pdfmetalang . . . . . 56, 177, 179, 181, 184, 792, 794
\@gobbletwo . . . . .	322, 337, 339	\@pdfmoddate . . . . . 159, 932, 936
\@hyxmp@gobbletwo . . . . .	322, 335	\@pdfsource . . . . . 62, 855, 857
\@ifmtarg . . . . .	16	\@pdfsubject . . . . . 160, 848
\@ifmtargexp . . . . .	16, 189, 195, 781, 816	\@pdftitle . . . . . 161, 189, 694, 700, 847
\@ifnotmtarg . . . . .	17	\@pdftype . . . . . 48, 162, 854
\@ifnotmtargexp . . . . .	16, 190, 196, 974	\@secondoftwo . . . . . 452
\@pdfaformance . . . . .	60, 1103	\@tempswafalse . . . . . 781, 816
\@pdfapart . . . . .	58, 1102	\@tempswatrue . . . . . 781, 783, 816, 818
\@pdfauthor . . . . .	95, 141, 195, 694, 700	\@title . . . . . 190, 191
\@pdfauthortitle . . . . .	52, 142, 951, 959	\^ . . . . . 231, 235, 395, 719, 720, 1116–1118
\@pdfcaptionwriter . . . . .	54, 143, 951, 960	\_ . . . . . 718, 720
\@pdfcontactaddress . . . . .	64, 144, 996, 1014	\~ . . . . . 240, 529, 530
\@pdfcontactcity . . . . .	72, 145, 997, 1015	
\@pdfcontactcountry . . . . .	78, 146, 1000, 1018	
\@pdfcontactemail . . . . .	82, 147, 1002, 1022	
\@pdfcontactphone . . . . .	80, 148, 1001, 1021	
\@pdfcontactpostcode . . . . .	76, 149, 999, 1017	
		<b>A</b>
		\and . . . . . 95
		ASCII . . . . . 13, 29
		\AtBeginDocument . . . . . 172
		\AtEndDocument . . . . . 5
		\AtEndDvi . . . . . 8
		atenddvi . . . . . 12
		Author . . . . . 7, 8, 22
		<b>B</b>
		Bag . . . . . 63
		baseurl (option) . . . . . 3, 11, 13, 44
		BOM . . . . . 50, 62
		<b>C</b>
		CiAdrCity . . . . . 2, 47
		CiAdrCtry . . . . . 2, 47
		CiAdrExtadr . . . . . 2, 47
		CiAdrPcode . . . . . 2, 47
		CiAdrRegion . . . . . 2, 47
		CiEmailWork . . . . . 2, 47
		CiTelWork . . . . . 2, 47
		CiUrlWork . . . . . 2, 47
		CreateDate . . . . . 44, 64
		CreationDate . . . . . 64
		<b>D</b>
		Date . . . . . 26
		\day . . . . . 355, 356, 358
		dc:creator . . . . . 2, 8, 42, 63
		dc:date . . . . . 2, 42
		dc:description . . . . . 3, 33, 42, 63
		dc:format . . . . . 2
		dc:language . . . . . 2, 42, 62, 63
		dc:rights . . . . . 2, 33, 42
		dc:source . . . . . 2, 42, 62
		dc:subject . . . . . 2, 42
		dc:title . . . . . 3, 33, 42, 63
		dc:type . . . . . 2
		\define@key . . . . . 25, 36, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 73, 75, 77, 79, 81, 83, 85, 95, 116, 597
		\do . . . . . 599, 606, 802
		dvipdf (option) . . . . . 52
		dvipdfm . . . . . 53

dvips (option) . . . . .	52	1075, 1088, 1098,	\hyxmp@dc@schema . . .
dvips . . . . .	7, 30, 62	1104, 1124, 1138	. . . . . <u>841</u> , 1132
dvipsone (option) . . . .	52	\hyxmp@alt@description	\hyxmp@def@DocumentID
dviwindo (option) . . . .	52	. . . . . <u>593</u> , 603	. . . . . <u>693</u> , 892
		\hyxmp@alt@rights . .	\hyxmp@def@InstanceID
		. . . . . <u>593</u> , 604	. . . . . <u>699</u> , 893
<b>E</b>		\hyxmp@alt@title <u>593</u> , 602	\hyxmp@define@createdate
\EdefEscapeHex . . . . .	418, 431	\hyxmp@and . . . . .	. . . . . <u>902</u> , 918
\EdefUnescapeHex . . . .	435	\hyxmp@append@hex . .	\hyxmp@DocumentID . .
\EdefUnescapeString . . .	405	. . . . . <u>642</u> , 661–663, 667	. . . . . <u>693</u> , 897
ETX . . . . .	21, 28	\hyxmp@append@hex@iii	\hyxmp@dq@code . . <u>1</u> , 1278
		. . . . . <u>660</u> , 666, 676, 687	\hyxmp@driver . . <u>3</u> , <u>1145</u>
<b>G</b>		\hyxmp@append@hex@iv	\hyxmp@embed@packet
Ghostscript . . . . .	7	. . . . . <u>665</u> , 671,	. . . . . <u>203</u> , <u>1145</u>
		672, 674, 689–691	\hyxmp@embed@packet@dvipdfm
<b>H</b>		\hyxmp@as@pdf@date . <u>282</u>	. . . . . <u>1157</u> , <u>1227</u>
\Hy@driver . . . . .		\hyxmp@as@xmp@date .	\hyxmp@embed@packet@luatex
. . . . .	4, 1148, 1152,	. . . . .	. . . . . <u>1153</u> , <u>1190</u>
	1156, 1160, 1165	30, 41, <u>254</u> , 926, 936	\hyxmp@embed@packet@pdfmark
\Hy@unicodefalse . . . .	27, 38	\hyxmp@at@end . . . .	. . . . . <u>1169</u> , <u>1197</u>
hyperref . . . . .	1, 3–6,	\hyxmp@big@prime . . .	\hyxmp@embed@packet@pdftex
	8, 11, 13, 16, 18–	. . . . . <u>616</u> , 619, 629, 639	. . . . . <u>1149</u> , <u>1177</u>
	20, 39, 51, 52, 62–64	\hyxmp@big@prime@ii	\hyxmp@embed@packet@xetex
\hypersetup <u>133</u> , 191, 197		. . . . . <u>616</u> , 638	. . . . . <u>1161</u> , <u>1265</u>
hyperxmp . . . . .	1–6,	\hyxmp@bom . . . . .	\hyxmp@find@metadata
	8–16, 19–22, 28,	\hyxmp@comma . . . . .	. . . . . <u>138</u> , 202
	35, 39, 51, 55, 62, 64	. . . . . <u>66</u> , <u>96</u> , <u>117</u> , <u>230</u>	\hyxmp@first@char . . <u>252</u>
\hyxmp@@is@unicode . . <u>439</u>		\hyxmp@commas@to@list	\hyxmp@first@char@i
\hyxmp@add@simple . . . .		. . . . . <u>214</u> , <u>245</u> , <u>827</u> , <u>988</u>	. . . . . <u>252</u> , <u>255</u> , <u>283</u>
. . . . .	752, 755, <u>761</u> ,	\hyxmp@commas@to@list@i	\hyxmp@gobbletwo . . . <u>322</u>
	857, 885, 920,	. . . . . <u>216</u> , <u>218</u>	\hyxmp@hash . . . . . <u>725</u> ,
	923, 925, 930,	\hyxmp@concat@metadata	1037–1040, 1128
	933, 935, 940,	. . . . . <u>138</u>	\hyxmp@Hyp@pdfauthor <u>89</u>
	942, 944, 945,	\hyxmp@construct@packet	\hyxmp@Hyp@pdfkeywords
	959, 960, 1015–	. . . . . <u>1122</u> , <u>1146</u>	. . . . . <u>110</u>
	1018, 1102, 1103	\hyxmp@count@non@spaces	\hyxmp@hypersetup . . <u>133</u>
\hyxmp@add@simple@var		. . . . . <u>1246</u> , <u>1257</u>	\hyxmp@InstanceID . .
. . . . .	747, 748, <u>771</u>	\hyxmp@count@spaces	. . . . . <u>699</u> , 898
\hyxmp@add@to+xml . . .		. . . . . <u>1245</u> , <u>1248</u>	\hyxmp@iptc@extensions
. . . . .	<u>705</u> ,	\hyxmp@crap@convert	. . . . . <u>1007</u> , <u>1033</u>
	731, 743, 757,	. . . . . <u>521</u> , <u>555</u>	\hyxmp@is@unicode . .
	766, 775, 788,	\hyxmp@crap@result . .	. . . . . <u>407</u> , <u>424</u> , <u>439</u>
	795, 799, 804,	. . . . . <u>511</u> , <u>547</u>	\hyxmp@LA@accept . . .
	809, 821, 829,	\hyxmp@crap@test <u>518</u> , <u>543</u>	. . . . . <u>596</u> , 602–604
	835, 842, 859,	\hyxmp@create@uuid . .	\hyxmp@legal . . . . . <u>864</u>
	875, 881, 886,	. . . . . <u>669</u> , 697, 703	\hyxmp@list . . . . .
	894, 914, 946,	\hyxmp@createdate . .	. . . . . <u>827</u> , <u>833</u> , <u>988</u> , <u>989</u>
	954, 963, 976,	. . . . . <u>902</u> , 920,	\hyxmp@list@to@lines
	980, 984, 990,	. . . . . <u>923</u> , <u>930</u> , <u>933</u> , <u>940</u>	. . . . . <u>973</u> ,
	1008, 1027, 1034,	\hyxmp@cur@lang <u>599</u> , <u>607</u>	1014, 1021–1023

\hyxmp@list@to@xml .	\hyxmp@redefine@Hyp	\hyxmp@x@default . . .
. . . . . 815, 850–854	. . . . . 88, 130, 135	. 179, 741, 792, 800
\hyxmp@mm@schema . . .	\hyxmp@reencode . . . 400	\hyxmp@xetex@crap . .
. . . . . 891, 1137	\hyxmp@rights . . . . .	. . . . . 412, 511
\hyxmp@modulo@a 610,	. 864, 867, 871, 873	\hyxmp+xml . . . . .
629, 639, 645, 680	\hyxmp@seed@rng . . .	722, 729, 1122,
\hyxmp@new@xml 721, 722	. . . . . 618, 695, 701	1186, 1194, 1217,
\hyxmp@num . . . . . 555	\hyxmp@seed@rng@i . .	1228, 1235, 1266
\hyxmp@one@token . . .	. . . . . 620, 622	\hyxmp+xmlified 403,
618, 622, 1249,	\hyxmp@seed@string .	767, 776, 787,
1250, 1258, 1259	. . . . . 693, 699	796, 805, 827, 988
\hyxmp@padding 729, 1141	\hyxmp@set@rand@num	\hyxmp+xmlify . . . . .
\hyxmp@parse@time . .	. . . . . 635, 643, 678	. . . . . 184, 403,
. . . . . 263, 265	\hyxmp@skiptorelax .	765, 774, 786,
\hyxmp@parse@tz . . .	. . . . . 548, 554	794, 803, 826, 987
. . . . . 272, 275, 279	\hyxmp@skipzeros . . . 506	\hyxmp@xmp@basic@schema
\hyxmp@parse@tz@char	\hyxmp@SpaceOther . .	. . . . . 913, 1135
. . . . . 267, 269	. . . . . 515, 528	\hyxmp@xmp@to@pdf@date
\hyxmp@pdf@schema . .	\hyxmp@string . . . . . 761	. . . . . 286, 289
. . . . . 742, 1130	\hyxmp@string@len . .	\hyxmp@xmp@to@pdf@date@i
\hyxmp@pdf@to@xmp@date	. . . . . 1228, 1243	. . . . . 290, 292
. . . . . 256, 261,	\hyxmp@sublist . . . .	\hyxmp@xmp@to@pdf@date@ii
379, 382, 907, 910	. 219, 220, 223, 224	. . . . . 295, 298
\hyxmp@pdfa@id@schema	\hyxmp@temp@list . . . 243	\hyxmp@xmp@to@pdf@date@iii
. . . . . 1096, 1136	\hyxmp@temp@str . . . 243	. . . . . 301, 304
\hyxmp@pdfauthor . . .	\hyxmp@text . . . . .	\hyxmp@xmp@to@pdf@date@iv
. . . . . 86, 95, 850	. 403, 481, 511, 555	. . . . . 307, 310
\hyxmp@pdfkeywords .	\hyxmp@text@resource	\hyxmp@xmp@to@pdf@date@v
. . . . . 86, 116, 851	. . 1067–1074, 1087	. . . . . 313, 316
\hyxmp@pdfstringdef	\hyxmp@textunderscore	\hyxmp@xmp@to@pdf@date@vi
. . . . . 18,	. . . . . 18	. . . . . 319, 323
29, 40, 47, 49, 51,	\hyxmp@today . . . . .	\hyxmp@xmp@to@pdf@date@vii
53, 55, 57, 59, 61,	. 187, 375, 700, 852	. . . . . 326, 329, 339
63, 68, 73, 75, 77,	\hyxmp@today@define	\hyxmp@xmp@to@pdf@date@viii
79, 81, 83, 85, 598	. . . . . 348, 377, 905	. . . . . 342, 345
\hyxmp@photometa@data	\hyxmp@toxml . . 433, 456	\hyxmp@xmpRights@schema
. . . . . 994	\hyxmp@toxml@unicodetex	. . . . . 863, 1131
\hyxmp@photometa@schema	. . . . . 421, 481	\hyxmp@zero . . . . . 564,
. . . . . 994, 1134	\hyxmp@trimb . . 388, 391	571, 578, 584, 589
\hyxmp@photoshop@data	\hyxmp@trimc . . 391, 392	
. . . . . 950	\hyxmp@trimspaces . .	<b>I</b>
\hyxmp@photoshop@schema	. . . . . 223, 384	IETF . . . . . 5
. . . . . 950, 1133	\hyxmp@try . . . . . 511	\if@tempswa . . . 785, 820
\hyxmp@ProcessKeyvalOptions	\hyxmp@unicodetexfalse	\iffalse . . . . . 780, 815
. . . . . 128	. . . . . 394	\ifHy@pdfa . . . . .
\hyxmp@rand@num 635,	\hyxmp@unicodetextrue	847, 848, 850, 1097
644, 679, 696, 702	. . . . . 394	\ifhyxmp@unicodetex
\hyxmp@rdfsdc . . . . .	\hyxmp@uscore . . 20, 234	394, 406, 708, 1110
. . . . . 780, 847–849	\hyxmp@value . . 598, 787	ifluatex . . . . . 52
		\ifluatex . . . 1179, 1183

ifmtarg ..... 13, 64  
 ifxetex ..... 13  
 \ifxetex ..... 411  
 Info ..... 7  
 intcalc ..... 13  
 \intcalcDiv 560, 567, 574  
 \intcalcMod 562, 569, 576  
 IPTC . 9, 15, 37, 47, 48, 62  
 lptc4xmpCore:CreatorContactInfo  
     ..... 2, 47, 62  
 ISO ..... 12, 15, 33

## J

\jobname ... 62, 166,  
           207, 694, 700, 1166

## K

Keywords ..... 7, 39  
 \KV@Hyp@pdfauthor .. 95  
 \KV@Hyp@pdfkeywords 116  
 kvoptions ..... 13, 18

## L

LF ..... 46  
 Lua<sup>L</sup>TeX ..... 7,  
           10, 26, 27, 39, 44, 63  
 LuaTeX ..... 28,  
           31, 51, 52, 55, 62, 64

## M

memoir ..... 63  
 Metadata ..... 7, 51, 54  
 MetadataDate .... 44, 64  
 ModifyDate ..... 44, 64  
 \month .... 350, 351, 353

## N

NAK ..... 13, 21, 28  
 nativepdf (option) ... 52  
 \newif ..... 394  
 \next ... 28, 39, 218, 622  
 ngerman ..... 12, 61  
 \number 558, 560, 562,  
           567, 569, 574, 576  
 \numexpr ..... 1195

## O

options  
   baseurl . 3, 11, 13, 44  
   dvipdf ..... 52

dvips ..... 52  
 dvipsone ..... 52  
 dviwindo ..... 52  
 nativepdf ..... 52  
 pdfa ..... 5, 64  
 pdfaconformance .  
     ..... 4, 5, 49  
 pdfapart .... 4, 5, 49  
 pdfauthor 3, 10, 11,  
           13, 16, 17, 20, 42, 63  
 pdfauthortitle . 4, 5, 11  
 pdfcaptionwriter . 4, 5  
 pdfcontactaddress .  
     ..... 4, 5, 9, 10  
 pdfcontactcity .. 4, 5  
 pdfcontactcountry 4, 5  
 pdfcontactemail . 4, 5  
 pdfcontactphone . 4, 5  
 pdfcontactpostcode  
     ..... 4, 5  
 pdfcontactregion . 4, 5  
 pdfcontacturl . 4, 5, 11  
 pdfcopyright .....  
     ... 4, 5, 42, 43, 62  
 pdfcreationdate 3, 5, 64  
 pdfdate 4, 5, 11, 14, 63  
 pdfkeywords .....  
     ... 3, 10, 13, 16, 42  
 pdflang .... 3, 6,  
           11, 13, 19, 33, 42, 64  
 pdflicenseurl .....  
     ... 4, 5, 11, 43, 62  
 pdfmark ..... 52  
 pdfmetadate 4, 5, 14, 64  
 pdfmetalang .....  
     4–6, 11, 33, 61, 64  
 pdfmoddate .. 4, 5, 64  
 pdfproducer .... 4, 13  
 pdfsource .... 4, 6, 64  
 pdfsubject .. 4, 13, 42  
 pdftitle ..... 4,  
           11, 13, 20, 42, 63  
 pdftype ..... 4, 6, 64  
 ps2pdf ..... 52  
 textures ..... 52  
 unicode ..... 11, 64  
 vtexpdfmark ..... 52

## P

\PackageWarningNoLine  
     ... 165, 206, 1164  
 PDF 1–3, 5, 7–10, 13, 14,  
     20–24, 30, 37, 39,  
     44, 50–52, 54, 63, 64  
 PDF/A ..... 3, 5, 15,  
     37, 39, 48, 49, 62, 63  
 pdf:Keywords ..... 2, 39  
 pdf:PDFVersion .... 3, 39  
 pdf:Producer ..... 3, 39  
 pdfa (option) ..... 5, 64  
 pdfaconformance (op-  
     tion) ... 4, 5, 49  
 pdfaid:conformance ... 3  
 pdfaid:part ..... 3  
 pdfapart (option) . 4, 5, 49  
 pdfaType:prefix ..... 62  
 pdfauthor (option) ...  
     ..... 3, 10, 11,  
     13, 16, 17, 20, 42, 63  
 pdfauthortitle (option)  
     ..... 4, 5, 11  
 pdfcaptionwriter (op-  
     tion) ..... 4, 5  
 \pdfcatalog ..... 1187  
 \pdfcompresslevel . 1181  
 pdfcontactaddress (op-  
     tion) . 4, 5, 9, 10  
 pdfcontactcity (option)  
     ..... 4, 5  
 pdfcontactcountry (op-  
     tion) ..... 4, 5  
 pdfcontactemail (op-  
     tion) ..... 4, 5  
 pdfcontactphone (op-  
     tion) ..... 4, 5  
 pdfcontactpostcode (op-  
     tion) ..... 4, 5  
 pdfcontactregion (op-  
     tion) ..... 4, 5  
 pdfcontacturl (option)  
     ..... 4, 5, 11  
 pdfcopyright (option) .  
     ... 4, 5, 42, 43, 62  
 pdfcreationdate (option)  
     ..... 3, 5, 64  
 \pdfcreationdate 379, 907  
 pdfdate (option) ....  
     ... 4, 5, 11, 14, 63

PDFDocEncoding . . .	Producer . . . . . 39	UUID . . . . . 34, 36, 37, 63
. . . . . 16, 28, 29	ps2pdf (option) . . . . . 52	
pdfescape . . . . . 13		<b>V</b>
\pdfextension 1191, 1195	<b>Q</b>	\vfuZZ . . . . . 392
\pdffeedback . . . . .	\Q . . . . . 384, 393	vtexpdfmark (option) . 52
. . . . . 382, 910, 1195		
pdfkeywords (option) .	<b>R</b>	<b>X</b>
. . . . . 3, 10, 13, 16, 42	RDF . . . . . 41	\x . . . . . <u>511</u>
pdflang (option) 3, 6,	rdf:li . . . . . 2	xdvipdfmx . . . . . 10, 54
11, 13, 19, 33, 42, 64	rdf:Seq . . . . . 2	X <sub>q</sub> L <sup>A</sup> T <sub>E</sub> X . . . . . 7,
\pdflastobj . . . . . 1187	\renewcommand . . . . . 129	10, 26, 39, 44, 63
pdfL <sup>A</sup> T <sub>E</sub> X . . . . .	\RequirePackage . . . . .	X <sub>q</sub> L <sup>A</sup> T <sub>E</sub> X . . . . . 13,
3, 7, 10, 26, 39, 44	. . . . . 7, 10–15, 1176	28, 31, 54, 55, 61, 62
pdflicenseurl (option) .		XML . . . . . 1, 2,
. . . . . 4, 5, 11, 43, 62	<b>S</b>	9, 20, 28–31, 37,
pdfmark (option) . . . . . 52	\SE->pdfdoc@03 . . . . . <u>401</u>	39–41, 46, 47, 50, 64
\pdfmark . . . . . 1198,	\SE->pdfdoc@15 . . . . . <u>402</u>	XMP . . . . . 1–3, 5–11,
1201, 1205,	\setkeys . . . . . 608	13, 14, 19–24, 26,
1215, 1219, 1223	\special . . . . . 1229,	27, 30, 33, 34, 37–
pdfmetadate (option) .	1237, 1266, 1272	44, 48, 52–55, 61–64
. . . . . 4, 5, 14, 64	stringenc . . . . . 13	xmp:BaseURL . . . . . 2
pdfmetalang (option) .	\StringEncodingConvert	xmp:CreateDate . . . . . 2
4–6, 11, 33, 61, 64	. . . . . 408,	xmp:CreatorTool . . . . . 3
\pdfminorversion . . . 752	414, 425, 428, 523	xmp:MetadataDate . . . . . 2
pdfmoddate (option) .	Subject . . . . . 7	xmp:ModifyDate . . . . . 2
. . . . . 4, 5, 64		\xmpcomma . . . . .
\pdfobj . . . . . 1183	<b>T</b>	66, 69, <u>95</u> , <u>116</u> , <u>229</u>
pdfproducer (option) 4, 13	T <sub>E</sub> X . . . . . 28, 30, 31,	xmpincl . . . . . 3
pdfsource (option) 4, 6, 64	34, 35, 43, 54, 55, 62	\XMPLangAlt . . . . . <u>605</u>
\pdfstringdef . . . . . 21	Text . . . . . 49	\xmplineSep 968, 984, <u>1019</u>
pdfsubject (option) . .	\textunderscore . . . . .	xmpMM:DocumentID . .
. . . . . 4, 13, 42	. . . . . 19, 20, 22	. . . . . 2, 34, 43
pdfT <sub>E</sub> X . . . . . 12,	textures (option) . . . . . 52	xmpMM:InstanceID . .
30, 51, 52, 55, 64	\time . . . . . 360, 368	. . . . . 2, 34, 43
pdftitle (option) . . 4,	Title . . . . . 7	\xmpquote . . . . .
11, 13, 20, 42, 63		67, 70, <u>95</u> , <u>116</u> , <u>238</u>
pdftype (option) . 4, 6, 64	<b>U</b>	xmpRights:Marked 2, 42, 62
\pdfvariable . . . . . 755	Unicode . . . . . 11, 13, 28–	xmpRights:WebStatement
photoshop:AuthorsPosition	32, 41, 46, 50, 55, 62	. . . . . 2, 42, 62
. . . . . 3, 45	unicode (option) . . . . . 11, 64	\xmptilde . . . . . <u>239</u>
photoshop:CaptionWriter	URL . . . . . 2, 5,	\XMPTruncateList . . . <u>243</u>
. . . . . 3, 45	11, 14, 16, 43–45, 47	
PI . . . . . 37	UTF-16BE . . . . . 29	<b>Y</b>
\ProcessKeyvalOptions	UTF-32BE . . . . . 28, 29	\year . . . . . 349
. . . . . <u>128</u>	UTF-8 . . . . . 29	