

The hyperxmp package^{*}

Scott Pakin
scott+hyxmp@pakin.org

July 25, 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>
```

^{*}This document corresponds to `hyperxmp` v3.3, dated 2017/07/25.

```

    </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^AT_EX 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 `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`.¹ 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,

¹Although 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 \LaTeX 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 \LaTeX 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 \LaTeX 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:

- pdfL^AT_EX
- LuaL^AT_EX
- L^AT_EX + Dvipdfm
- L^AT_EX + Dvips + Adobe Acrobat Distiller
- X_YL^AT_EX

Unfortunately, the L^AT_EX + 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.

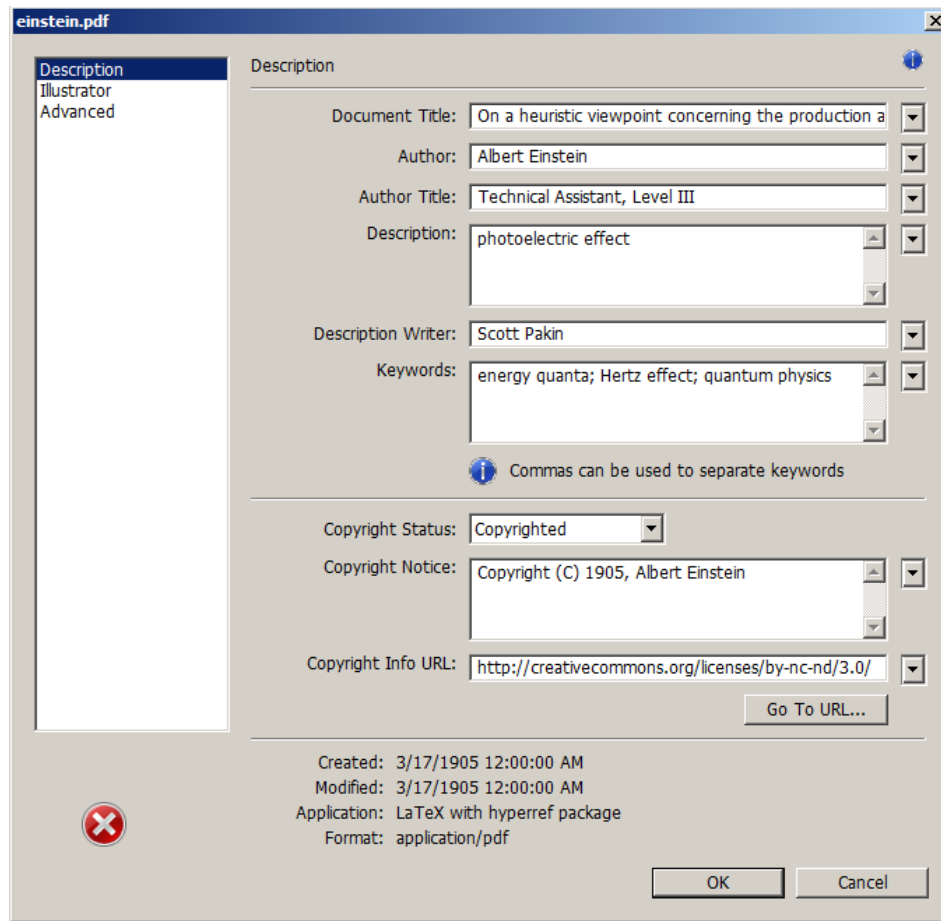


Figure 1: XMP metadata as it appears in Adobe Acrobat

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,

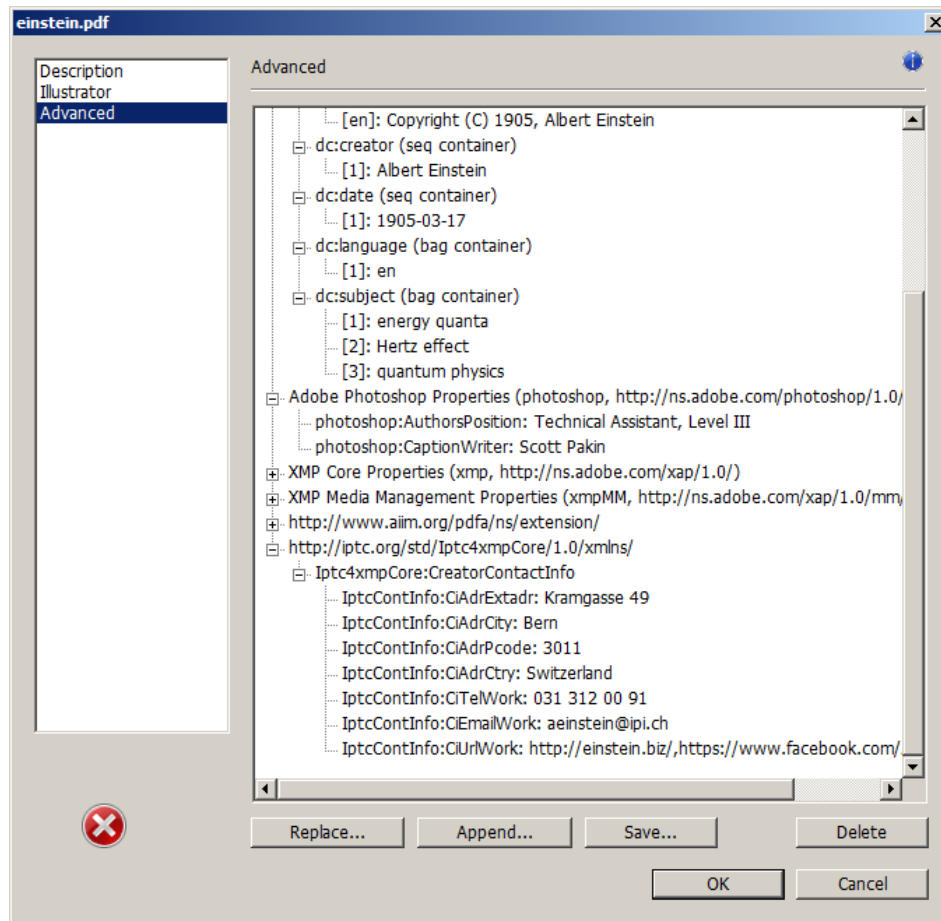


Figure 2: Additional XMP metadata as it appears in Adobe Acrobat

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

3.1 Initial preparation

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

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

`\hyxmp@at@end` The `\hyxmp@at@end` macro includes code at the end of the document. For pdfT_EX, the standard `\AtEndDocument` works well enough. For all the other backends we use `\AtEndDvi` from the `atenddvi` package, which is more robust but requires an addition L^AT_EX run.

`\hyxmp@driver`

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

```

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

```

3.2 Integration with hyperref

An important design decision underlying `hyperxmp` is that the package should integrate seamlessly with `hyperref`. To that end, `hyperxmp` takes its XMP metadata from `hyperref`'s `baseurl`, `pdfauthor`, `pdfkeywords`, `pdflang`, `pdfproducer`, `pdfsubject`, and `pdftitle` options. It also introduces a number of new options, which are listed on pages 4–5. For consistency with `hyperref`'s document-metadata naming conventions (which are in turn based on L^AT_EX's document-metadata naming conventions), we do not prefix metadata-related macro names with our package-specific `\hyxmp@` prefix. That is, we use names like `\@pdfcopyright` instead of `\hyxmp@pdfcopyright`.

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

```

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

```

```

\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.

```

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

```

```
\@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.

```

21 \def\@pdfdatetime{}
22 \define@key{Hyp}{pdfdate}{%
23 \begingroup
24 \Hy@unicodefalse

```

```
\next
```

Expand `pdfdate`'s argument and convert it to XMP format.

```

25     \edef\next{%
26         \noexpand\hyxmp@pdfstringdef\noexpand\@pdfdatetime{%
27             \noexpand\hyxmp@as@xmp@date{#1}}%
28     }%
29     \next
30 \endgroup
31 }

\@pdfmetadatetime 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.
32 \def\@pdfmetadatetime{}
33 \define@key{Hyp}{pdfmetadate}{%
34     \begingroup
35         \Hy@unicodetofalse

\next Expand pdfmetadate's argument and convert it to XMP format.
36     \edef\next{%
37         \noexpand\hyxmp@pdfstringdef\noexpand\@pdfmetadatetime{%
38             \noexpand\hyxmp@as@xmp@date{#1}}%
39     }%
40     \next
41 \endgroup
42 }

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

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

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

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

\@pdfcaptionwriter Prepare to store the name of the person who inserted the hyperxmp metadata.
51 \def\@pdfcaptionwriter{}
52 \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.
53 \def\@pdfmetalang{}
54 \define@key{Hyp}{pdfmetalang}{\hyxmp@pdfstringdef\@pdfmetalang{#1}}

```

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

```
55 \def\@pdfapart{1}
56 \define@key{Hyp}{pdfapart}{\hyxmp@pdfstringdef\@pdfapart{#1}}
```

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

```
57 \def\@pdfaconformance{B}
58 \define@key{Hyp}{pdfaconformance}{\hyxmp@pdfstringdef\@pdfaconformance{#1}}
```

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

```
59 \edef\@pdfsource{\jobname.tex}
60 \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.

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

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

```
69 \def\@pdfcontactcity{}
70 \define@key{Hyp}{pdfcontactcity}{\hyxmp@pdfstringdef\@pdfcontactcity{#1}}
```

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

```
71 \def\@pdfcontactregion{}
72 \define@key{Hyp}{pdfcontactregion}{\hyxmp@pdfstringdef\@pdfcontactregion{#1}}
```

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

```
73 \def\@pdfcontactpostcode{}
74 \define@key{Hyp}{pdfcontactpostcode}{\hyxmp@pdfstringdef\@pdfcontactpostcode{#1}}
```

<code>\@pdfcontactcountry</code>	<p>Prepare to store the country of the document's contact person/institution.</p> <pre> 75 \def\@pdfcontactcountry{} 76 \define@key{Hyp}{pdfcontactcountry}{\hyxmp@pdfstringdef\@pdfcontactcountry{#1}} </pre>
<code>\@pdfcontactphone</code>	<p>Prepare to store the telephone number of the document's contact person/institution.</p> <pre> 77 \def\@pdfcontactphone{} 78 \define@key{Hyp}{pdfcontactphone}{\hyxmp@pdfstringdef\@pdfcontactphone{#1}} </pre>
<code>\@pdfcontactemail</code>	<p>Prepare to store the email address of the document's contact person/institution.</p> <pre> 79 \def\@pdfcontactemail{} 80 \define@key{Hyp}{pdfcontactemail}{\hyxmp@pdfstringdef\@pdfcontactemail{#1}} </pre>
<code>\@pdfcontacturl</code>	<p>Prepare to store the URL of the document's contact person/institution.</p> <pre> 81 \def\@pdfcontacturl{} 82 \define@key{Hyp}{pdfcontacturl}{\hyxmp@pdfstringdef\@pdfcontacturl{#1}} </pre>

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>	<p>Prepare to store the name of the author and a list of keywords.</p> <pre> 83 \def\hyxmp@pdfauthor{} 84 \def\hyxmp@pdfkeywords{} </pre>
<code>\hyxmp@redefine@Hyp</code>	<p>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>.</p> <pre> 85 \newcommand*{\hyxmp@redefine@Hyp}{% </pre>
<code>\hyxmp@Hyp@pdfauthor</code>	<p>Store the old definition of <code>\KV@Hyp@pdfauthor</code> in <code>\hyxmp@Hyp@pdfauthor</code>, but only if we see that <code>\KV@Hyp@pdfauthor</code> is defined and <code>\hyxmp@Hyp@pdfauthor</code> isn't. Otherwise, we'd be defining <code>\hyxmp@Hyp@pdfauthor</code> in terms of itself and creating an infinite loop.</p> <pre> 86 \ifundefined{KV@Hyp@pdfauthor}{}{% 87 \ifundefined{hyxmp@Hyp@pdfauthor}{% 88 \expandafter\let\expandafter\hyxmp@Hyp@pdfauthor 89 \csname KV@Hyp@pdfauthor\endcsname 90 }{}% 91 }% </pre>

`\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.

```

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

`\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`.

```

107  \ifundefined{KV@Hyp@pdfkeywords}{}%
108    \ifundefined{hyxmp@Hyp@pdfkeywords}{%
109      \expandafter\let\expandafter\hyxmp@Hyp@pdfkeywords
110        \csname KV@Hyp@pdfkeywords\endcsname
111    }{}%
112 }%
```

`\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).

```

113  \define@key{Hyp}{pdfkeywords}{%
114    \let\xmpcomma=\hyxmp@comma
115    \def\xmpquote####1{####1}%
116    \hyxmp@Hyp@pdfkeywords{##1}%
117    \global\let\hyxmp@pdfkeywords=\@pdfkeywords
```

```

118 \def\xmpcomma{,}%
119 \def\xmpquote####1{"####1"}%
120 \hyxmp@Hyp@pdfkeywords{##1}%
121 \def\xmpcomma{,}%
122 \let\xmpquote=\relax
123 }%
124 }

```

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

```

125 \let\hyxmp@ProcessKeyvalOptions=\ProcessKeyvalOptions
126 \renewcommand*{\ProcessKeyvalOptions}{%
127 \hyxmp@redefine@Hyp
128 \hyxmp@ProcessKeyvalOptions
129 }

```

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

```

130 \let\hyxmp@hypersetup=\hypersetup
131 \def\hypersetup{%
132 \hyxmp@redefine@Hyp
133 \hyxmp@hypersetup
134 }

```

`\hyxmp@find@metadata` Issue a warning message if the author failed to specify any metadata at all. This
`\hyxmp@concat@metadata` excludes metadata that is included automatically such as the current timestamp.
Note that we don't consider `\@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.

```

135 \newcommand*{\hyxmp@find@metadata}{%
136 \edef\hyxmp@concat@metadata{%
137 \@baseurl
138 \@pdfauthor
139 \@pdfauthortitle
140 \@pdfcaptionwriter
141 \@pdfcontactaddress
142 \@pdfcontactcity
143 \@pdfcontactcountry
144 \@pdfcontactemail
145 \@pdfcontactphone
146 \@pdfcontactpostcode
147 \@pdfcontactregion
148 \@pdfcontacturl
149 \@pdfcopyright
150 \@pdfcreationdate
151 \@pdfdatetime
152 \@pdfkeywords
153 \@pdflang
154 \@pdflicenseurl

```

```

155     \@pdfmetadatetime
156     \@pdfmoddate
157     \@pdfsubject
158     \@pdftitle
159     \@pdftype
160 }%
161 \ifx\hyxmp@concat@metadata\@empty
162     \PackageWarningNoLine{hyperxmp}{%
163 \jobname.tex did not specify any metadata to\MessageBreak
164 include in the XMP packet.\space\space Please see the\MessageBreak
165 hyperxmp documentation for instructions on how to\MessageBreak
166 provide metadata values to hyperxmp}%
167 \fi
168 }

```

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.

```

169 \AtBeginDocument{%
170     \@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`.

```

171     \ifx\@pdflang\relax
172         \let\@pdflang=\@empty
173     \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.

```

174     \ifx\@pdfmetalang\@empty
175         \ifx\@pdflang\@empty
176             \let\@pdfmetalang=\hyxmp@x@default
177         \else
178             \edef\@pdfmetalang{\@pdflang}%
179         \fi
180     \fi
181     \hyxmp@xmlify\@pdfmetalang

```

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

```

182     \ifx\@pdfdatetime\@empty
183     \else
184         \edef\hyxmp@today{\@pdfdatetime}%
185     \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`.

```

186 \ifx\@pdftitle\@empty
187 \ifx\@title\@empty
188 \else
189 \hypersetup{pdftitle={\@title}}%
190 \fi
191 \fi
192 \ifx\@pdfauthor\@empty
193 \ifx\@author\@empty
194 \else
195 \hypersetup{pdfauthor={\@author}}%
196 \fi
197 \fi

```

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`.

```

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

```

3.3 Manipulating author-supplied data

The author provides metadata information to `hyperxmp` via package options to `hyperref` or via `hyperref`'s `\hypersetup` command. The functions in this section convert author-supplied lists (e.g., `pdfkeywords={foo, bar, baz}`) into L^AT_EX lists (e.g., `\@elt {foo} \@elt {bar} \@elt {baz}`) that can be more easily manipulated (Section 3.3.1); trim spaces off the ends of strings (Section 3.3.3); and, in Section 3.3.4, convert text to XML (e.g., from `<scott+hyxmp@pakin.org>` to `<scott+hyxmp@pakin.org>`).

3.3.1 List manipulation

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

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

```

211 \newcommand*{\hyxmp@commas@to@list}[2]{%
212   \gdef#1{%
213     \expandafter\hyxmp@commas@to@list@i\expandafter#1#2,,%
214   }

```

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

```

\next 215 \def\hyxmp@commas@to@list@i#1#2,{%
216   \gdef\hyxmp@sublist{#2}%
217   \ifx\hyxmp@sublist\@empty
218     \let\next=\relax
219   \else
220     \hyxmp@trimspaces\hyxmp@sublist
221     \@cons{#1}{\hyxmp@sublist}%
222     \def\next{\hyxmp@commas@to@list@i{#1}}%
223   \fi
224   \next
225 }

```

`\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.

```

226 \def\xmpcomma{,}%

```

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

```

227 \bgroup
228   \catcode'\^^C=11
229   \gdef\hyxmp@comma{^^C}
230 \egroup

```

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

```

231 \bgroup
232   \catcode'\^^U=11
233   \gdef\hyxmp@uscore{^^U}
234 \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.

```
235 \let\xmpquote=\relax
```

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

```
236 \bgroup
237 \catcode'\~ =12%
238 \gdef\xmptilde{~}%
239 \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 `\hyxmp@temp@list` Adobe Acrobat display the author list correctly. It's sad that this is necessary, though.

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

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:20170725220856-06'00') [3], while XMP timestamps are of the form “YYYY-MM-DDThh:mm:ss+TT:tt” (e.g., 2017-07-25T22:08:56-06:00) [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 249 \def\hyxmp@first@char#1{\hyxmp@first@char@i#1\relax}
250 \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.

```
251 \def\hyxmp@as@xmp@date#1{%
252 \expandafter\ifx\hyxmp@first@char@i#1\relax D%
```

```

253     \hyxmp@pdf@to@xmp@date{#1}%
254   \else
255     #1%
256   \fi
257 }

```

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

```

258 \def\hyxmp@pdf@to@xmp@date#1:#2#3#4#5#6#7#8#9{%
259   #2#3#4#5-#6#7-#8#9%
260   \hyxmp@parse@time
261 }

```

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

```

262 \def\hyxmp@parse@time#1#2#3#4#5#6{%
263   T#1#2:#3#4:#5#6%
264   \hyxmp@parse@tz@char
265 }

```

`\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).

```

266 \def\hyxmp@parse@tz@char#1{%
267   #1%
268   \ifx#1-%
269     \expandafter\hyxmp@parse@tz
270   \else
271     \ifx#1+%
272       \expandafter\hyxmp@parse@tz
273     \fi
274   \fi
275 }

```

`\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.

```

276 \def\hyxmp@parse@tz#1'#2'{%
277   #1:#2%
278 }

```

`\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.

```

279 \def\hyxmp@as@pdf@date#1{%
280   \expandafter\ifx\hyxmp@first@char@i#1\relax D%
281     #1%
282   \else
283     \hyxmp@xmp@to@pdf@date{#1}%
284   \fi
285 }

```

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

```

286 \def\hyxmp@xmp@to@pdf@date#1{%
287   D:\hyxmp@xmp@to@pdf@date@i#1\relax\relax
288 }

```

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

```

289 \def\hyxmp@xmp@to@pdf@date@i#1#2#3#4#5#6{%
290   #1#2#3#4%
291   \ifx#5-%
292     \expandafter\hyxmp@xmp@to@pdf@date@ii\expandafter#6%
293   \fi
294 }

```

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

```

295 \def\hyxmp@xmp@to@pdf@date@ii#1#2#3#4{%
296   #1#2%
297   \ifx#3-%
298     \expandafter\hyxmp@xmp@to@pdf@date@iii\expandafter#4%
299   \fi
300 }

```

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

```

301 \def\hyxmp@xmp@to@pdf@date@iii#1#2#3#4{%
302   #1#2%
303   \ifx#3T%
304     \expandafter\hyxmp@xmp@to@pdf@date@iv\expandafter#4%
305   \fi
306 }

```

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

```

307 \def\hyxmp@xmp@to@pdf@date@iv#1#2#3#4{%
308   #1#2%
309   \ifx#3:%
310     \expandafter\hyxmp@xmp@to@pdf@date@v\expandafter#4%
311   \fi
312 }

```


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

```

313 \def\hyxmp@xmp@to@pdf@date@v#1#2#3#4{%
314   #1#2%
315   \ifx#3:%
316     \expandafter\hyxmp@xmp@to@pdf@date@vi\expandafter#4%
317   \fi
318 }

```

`\hyxmp@gobbletwo` This is exactly the same as L^AT_EX 2_ε's `\@gobbletwo` but needs to be a different literal for `\hyxmp@xmp@to@pdf@date@vii`'s pattern-matching to work.

```

319 \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).

```

320 \def\hyxmp@xmp@to@pdf@date@vi#1#2#3#4{%
321   #1#2%
322   \ifx#3+%
323     +\expandafter\hyxmp@xmp@to@pdf@date@vii
324   \fi
325   \ifx#3-%
326     -\expandafter\hyxmp@xmp@to@pdf@date@vii
327   \fi
328   \ifx#3Z%
329     Z%
330   \fi
331   \ifx#3\relax
332     \expandafter\@hyxmp@gobbletwo
333   \fi
334   \@gobbletwo #4%
335 }

```

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

```

336 \def\hyxmp@xmp@to@pdf@date@vii#1\@gobbletwo#2#3#4#5{%
337   #2#3%
338   \ifx#4:%
339     \expandafter\hyxmp@xmp@to@pdf@date@viii\expandafter#5%
340   \fi
341 }

```

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

```

342 \def\hyxmp@xmp@to@pdf@date@viii#1#2#3#4{%
343   '#1#2'%
344 }

```

`\hyxmp@today@define` Use T_EX primitives to define a given macro as today's date in YYYY-MM-DDThh:mm format.

```

345 \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.
346 \xdef#1{\the\year}%
347 \ifnum\month<10
348 \xdef#1{#1-0\the\month}%
349 \else
350 \xdef#1{#1-\the\month}%
351 \fi
352 \ifnum\day<10
353 \xdef#1{#1-0\the\day}%
354 \else
355 \xdef#1{#1-\the\day}%
356 \fi

```

T_EX 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_EX to query the number of seconds since midnight or the timezone so we omit those fields when defining macro #1.

```

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

```

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

```

372 \@ifundefined{pdffeedback}{%
373 \@ifundefined{pdfcreationdate}{%

```

Case 1: Neither `\pdffeedback` nor `\pdfcreationdate` is defined (X_qL^AT_EX and regular L^AT_EX).

```

374 \hyxmp@today@define\hyxmp@today
375 }{%

```

Case 2: `\pdfcreationdate` is defined (pdfL^AT_EX and pre-0.85 LuaL^AT_EX).

```

376 \edef\hyxmp@today{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
377 }%
378 }{%

```

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

```
379 \edef\hyxmp@today{\expandafter\hyxmp@pdf@to@xmp@date\pdffeedback creationdate}%
380 }
```

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.

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

`\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.

```
388 \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.

```
389 \def\hyxmp@trimc#1Q#2{\afterassignment\endgroup \vfuzz\the\vfuzz#1}
390 \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 XeqTeX 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.
391 \newif\ifhyxmp@unicodetex
392 \ifnum64='^^^^0040\relax
393   \hyxmp@unicodetexttrue
394 \else
395   \hyxmp@unicodetextfalse
396 \fi

\hyxmp@reencode This is now a placeholder macro needed only for \@pdfmetalang in the
\begin{document}.
397 \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.
398 \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.
399 \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;.
400 \newcommand*{\hyxmp@xmlify}[1]{%
401   \gdef\hyxmp@xmlified{%
Escaped PDF string → PDFDocEncoding/Unicode
402   \EdefUnescapeString\hyxmp@text{#1}%
403   \ifhyxmp@unicodetex
PDFDocEncoding/Unicode → UTF-32BE
404     \hyxmp@is@unicode\hyxmp@text{%
405       \StringEncodingConvert
406       \hyxmp@text\hyxmp@text{utf16be}{utf32be}%
407     }{%

```

```

408     \ifxetex
409         \hyxmp@xetex@crap
410     \else
411         \StringEncodingConvert
412         \hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
413     \fi
414 }%

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

UTF-32BE → XML in ASCII
416     \edef\hyxmp@text{%
417         \expandafter
418     }\expandafter\hyxmp@toxml@unicodetex\hyxmp@text
419     \relax\relax\relax\relax\relax\relax\relax\relax
420 \else

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

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

UTF-8 as hex string → XML in UTF-8 as hex string
429     \edef\hyxmp@text{%
430         \expandafter\hyxmp@toxml\hyxmp@text\@empty\@empty
431     }%

XML in UTF-8 as hex string → XML in UTF-8
432     \EdefUnescapeHex\hyxmp@text{\hyxmp@text}%
433 \fi
434 \global\let\hyxmp@xmlified\hyxmp@text
435 }

```

`\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.

```

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

```

```

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

```

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

```

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

```

463   \@ifundefined{pdfmark}{%
464     #1#2%
465   }{%
466     \ifnum"#1#2='\( %
467       5C28% \(
468     \else\ifnum"#1#2='\) %
469       5C29% \)
470     \else
471       #1#2%
472     \fi\fi
473   }%
474   \fi\fi\fi
475   \expandafter\hyxmp@toxml
476 \fi
477 }

```

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

```

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

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

```

503 \def\hyxmp@skipzeros#1{%
504   \ifx#10%
505     \expandafter\hyxmp@skipzeros
506   \fi
507 }
```

`\x` In the case of `XYTEX`, the strings defined by `\pdfstringdef` can contain big

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

```

\hyxmp@try 508 \begingroup
\hyxmp@crap@result 509 \def\x#1{\endgroup
\hyxmp@text 510 \def\hyxmp@xetex@crap{%
511   \edef\hyxmp@try{%
512     \expandafter\hyxmp@SpaceOther\hyxmp@text#1@nil
513   }%
514   \let\hyxmp@crap@result=N%
515   \expandafter\hyxmp@crap@test\hyxmp@try\relax
516   \ifx\hyxmp@crap@result Y%
517     \let\hyxmp@text\@empty
518     \expandafter\hyxmp@crap@convert\hyxmp@try\relax
519   \else
```

```

520     \StringEncodingConvert\hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
521     \fi
522 }%
523 }
524 \x{ }

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

\hyxmp@num 552 \def\hyxmp@crap@convert#1{%
\hyxmp@text 553     \ifx#1\relax
554     \else
555     \edef\hyxmp@num{\number'#1}%
556     \ifnum\hyxmp@num>"FFFFFF %
557     \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"1000000}\relax
558     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
559     \edef\hyxmp@num{\intcalMod{\hyxmp@num}{\number"1000000}}%
560     \else

```



```

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

```

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

```

586 \begingroup
587   \catcode0=12 %
588   \gdef\hyxmp@zero{^^00}%
589 \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}`”.

```

590 \def\hyxmp@alt@title{}
591 \def\hyxmp@alt@description{}
592 \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.`

```

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

```

`\hyxmp@value` As Niklas Beisert observed, if the option passed to the current key contains L^AT_EX code, this code will be included in the XMP packet, which is undesirable. Hence, we first clean up the string using `\hyxmp@pdfstringdef`.

```

595   \hyxmp@pdfstringdef\hyxmp@value{##1}%
596   \xdef#2{#2\noexpand\do {\hyxmp@cur@lang} {\hyxmp@value}}%
597 }
598 }

```

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

```

599 \hyxmp@LA@accept{pdftitle}{\hyxmp@alt@title}
600 \hyxmp@LA@accept{pdfsubject}{\hyxmp@alt@description}
601 \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.

```

602 \newcommand{\XMPLangAlt}[2]{%
603   \let\do=\relax

```

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

```

604   \edef\hyxmp@cur@lang{#1}%
605   \setkeys{hyxmp@LA}{#2}%
606 }

```

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_EX and is good enough for producing the XMP `xmpMM:DocumentID` and `xmpMM:InstanceID` fields.

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

```

607 \def\hyxmp@modulo@a#1{%
608   \@tempcntb=\@tempcnta
609   \divide\@tempcntb by #1
610   \multiply\@tempcntb by #1
611   \advance\@tempcnta by -\@tempcntb
612 }

```

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

```

\hyxmp@big@prime@ii 613 \def\hyxmp@big@prime{536870923}
614 \def\hyxmp@big@prime@ii{536870027}

```

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

```

\hyxmp@one@token 615 \def\hyxmp@seed@rng#1{%
616 \@tempcnta=\hyxmp@big@prime
617 \futurelet\hyxmp@one@token\hyxmp@seed@rng@i#1\@empty
618 }

```

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

```

\hyxmp@one@token \next 619 \def\hyxmp@seed@rng@i{%
620 \ifx\hyxmp@one@token\@empty
621 \let\next=\relax
622 \else
623 \def\next##1{%
624 \multiply\@tempcnta by 3
625 \advance\@tempcnta by '##1
626 \hyxmp@modulo@a{\hyxmp@big@prime}%
627 \futurelet\hyxmp@one@token\hyxmp@seed@rng@i
628 }%
629 \fi
630 \next
631 }

```

`\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 632 \def\hyxmp@set@rand@num{%
633 \@tempcnta=\hyxmp@rand@num
634 \multiply\@tempcnta by 3
635 \advance\@tempcnta by \hyxmp@big@prime@ii
636 \hyxmp@modulo@a{\hyxmp@big@prime}%
637 \xdef\hyxmp@rand@num{\the\@tempcnta}%
638 }

```

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

```

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

```

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

```

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

```

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

```

657 \def\hyxmp@append@hex@iii#1{%
658     \hyxmp@append@hex#1%
659     \hyxmp@append@hex#1%
660     \hyxmp@append@hex#1%
661 }

```

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

```

662 \def\hyxmp@append@hex@iv#1{%
663     \hyxmp@append@hex@iii#1%
664     \hyxmp@append@hex#1%
665 }

```

`\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`.

```

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

```

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

```

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

```

```

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

```

`\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.

```

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

```

`\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.

```

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

```

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.

```

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

```

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

```

722 \bgroup
723 \catcode'\#=11
724 \gdef\hyxmp@hash{#}
725 \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.

```

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

```

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

```

738 \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.

```

739 \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.

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

```

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

```

```

765     }%
766   \fi
767 }

```

`\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.

```

768 \newcommand*{\hyxmp@add@simple@var}[2]{%
769   \expandafter\ifx\csname#2\endcsname\relax
770   \else
771     \hyxmp@xmllify{\csname#2\endcsname}%
772     \hyxmp@add@to@xml{%
773       -----<#1>\hyxmp@xmllified</#1>^^J%
774     }%
775   \fi
776 }

```

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.

```

777 \newcommand*{\hyxmp@rdf@dc}[3][\iffalse]{%

```

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

```

778   \@tempswafalse
779   \ifx#3\@empty
780   \else
781     \@tempswatrue
782   \fi
783   #1
784   \@tempswatrue
785   \fi

```

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

```

786   \if@tempswa
787     \hyxmp@xmllify{#3}%

```

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

```

788   \let\hyxmp@value=\hyxmp@xmllified
789   \hyxmp@add@to@xml{%
790     -----<dc:#2>^^J%
791     -----<rdf:Alt>^^J%
792   }%

```



```

793 \ifx\@pdfmetalang\hyxmp@x@default
794 \else
795 \hyxmp@xmlify{\@pdfmetalang}%
796 \hyxmp@add@to@xml{%
797 -----<rdf:li xml:lang="\hyxmp@xmlified">\hyxmp@value</rdf:li>^^J%
798 }%
799 \fi
800 \hyxmp@add@to@xml{%
801 -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@value</rdf:li>^^J%
802 }%

```

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

```

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

```

Complete this XMP element.

```

810 \hyxmp@add@to@xml{%
811 -----</rdf:Alt>^^J%
812 -----</dc:#2>^^J%
813 }%
814 \fi
815 }%

```

\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.

```

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

```

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

```

817 \@tempwafalse
818 \ifx#4\@empty
819 \else
820 \@tempwattrue
821 \fi
822 #1
823 \@tempwattrue
824 \fi

```

Append the corresponding XML only if \@tempwattrue.

```

825 \if@tempwa
826 \hyxmp@add@to@xml{%
827 -----<dc:#2>^^J%
828 -----<rdf:#3>^^J%
829 }%

```

```

830      \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.

831      \hyxmp@xmlify{#4}%
832      \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
833      \def\@elt##1{%
834          \hyxmp@add@to@xml{%
835      -----<rdf:li>##1</rdf:li>^^J%
836          }%
837      }%
838      \hyxmp@list
839      \egroup
840      \hyxmp@add@to@xml{%
841      -----</rdf:#3>^^J%
842      -----</dc:#2>^^J%
843          }%
844      \fi
845  }

\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 LATEX and the dc:source
property using the base name of the source file with .tex appended.

846 \newcommand*{\hyxmp@dc@schema}{%
847     \hyxmp@add@to@xml{%
848     -----<rdf:Description rdf:about=""^^J%
849     -----_xmlns:dc="http://purl.org/dc/elements/1.1/">^^J%
850     -----<dc:format>application/pdf</dc:format>^^J%
851     }%
852     \hyxmp@rdf@dc[\ifHy@pdfa]{title}{\@pdftitle}%
853     \hyxmp@rdf@dc[\ifHy@pdfa]{description}{\@pdfsubject}%
854     \hyxmp@rdf@dc{rights}{\@pdfcopyright}%
855     \hyxmp@list@to@xml[\ifHy@pdfa]{creator}{Seq}{\hyxmp@pdfauthor}%
856     \hyxmp@list@to@xml{subject}{Bag}{\hyxmp@pdfkeywords}%
857     \hyxmp@list@to@xml{date}{Seq}{\hyxmp@today}%
858     \hyxmp@list@to@xml{language}{Bag}{\@pdflang}%
859     \hyxmp@list@to@xml{type}{Bag}{\@pdftype}%
860     \ifx\@pdfsource\@empty
861     \else
862         \hyxmp@add@simple{dc:source}{\@pdfsource}%
863     \fi
864     \hyxmp@add@to@xml{%
865     -----</rdf:Description>^^J%
866     }%

```

867 }

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.

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

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

```
869 \let\hyxmp@rights=\@empty
870 \ifx\@pdflicenseurl\@empty
871 \else
872 \def\hyxmp@rights{YES}%
873 \fi
874 \ifx\@pdfcopyright\@empty
875 \else
876 \def\hyxmp@rights{YES}%
877 \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.

```
878 \ifx\hyxmp@rights\@empty
879 \else
```

Header

```
880 \hyxmp@add@to@xml{%
881 -----<rdf:Description rdf:about=""^^J%
882 -----xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/">^^J%
883 }%
```

Copyright indication

```
884 \ifx\@pdfcopyright\@empty
885 \else
886 \hyxmp@add@to@xml{%
887 -----<xmpRights:Marked>True</xmpRights:Marked>^^J%
888 }%
889 \fi
```

License URL

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

Trailer

```
891 \hyxmp@add@to@xml{%
892 -----</rdf:Description>^^J%
893 }%
894 \fi
895 }
```

3.5.5 The XMP Media Management schema

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

```

896 \gdef\hyxmp@mm@schema{%
897   \hyxmp@def@DocumentID
898   \hyxmp@def@InstanceID
899   \hyxmp@add@to@xml{%
900     <rdf:Description rdf:about=""^^J%
901     _____xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/">^^J%
902     <xmpMM:DocumentID>\hyxmp@DocumentID</xmpMM:DocumentID>^^J%
903     <xmpMM:InstanceID>\hyxmp@InstanceID</xmpMM:InstanceID>^^J%
904     </rdf:Description>^^J%
905   }%
906 }
```

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.

```

907 \newcommand*{\hyxmp@define@createdate}{%
908   \ifundefined{pdffeedback}{%
909     \ifundefined{pdfcreationdate}{%
```

Case 1: Neither `\pdffeedback` nor `\pdfcreationdate` is defined (\TeX and regular \LaTeX).

```

910     \hyxmp@today@define\hyxmp@createdate
911   }%
```

Case 2: `\pdfcreationdate` is defined (\pdfLaTeX and pre-0.85 \LuaLaTeX).

```

912     \edef\hyxmp@createdate{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
913   }%
914 }
```

Case 3: `\pdffeedback` is defined (\LuaLaTeX 0.85+).

```

915     \edef\hyxmp@createdate{\expandafter\hyxmp@pdf@to@xmp@date\pdffeedback creationdate}%
916   }%
917 }
```

`\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`.

```

918 \newcommand*{\hyxmp@xmp@basic@schema}{%
919   \hyxmp@add@to@xml{%
920     <rdf:Description rdf:about=""^^J%
921     _____xmlns:xmp="http://ns.adobe.com/xap/1.0/">^^J%
```

```

922 }%
923 \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.
924 \@ifundefined{@pdfcreationdate}{%
925     \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@createdate}%
926 }{%
927     \ifx\@pdfcreationdate\@empty
928         \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@createdate}%
929     \else
930         \hyxmp@add@simple{xmp:CreateDate}{%
931             \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfcreationdate}}%
932         \fi
933 }%
    For the document's modification date, use the user-specified \@pdfmoddate if
    defined and non-empty. Otherwise use our fabricated \hyxmp@createdate.
934 \@ifundefined{@pdfmoddate}{%
935     \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@createdate}%
936 }{%
937     \ifx\@pdfmoddate\@empty
938         \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@createdate}%
939     \else
940         \hyxmp@add@simple{xmp:ModifyDate}{%
941             \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfmoddate}}%
942         \fi
943 }%
    For the document's metadata date, use the user-specified \@pdfmetadatetime if
    defined and non-empty. Otherwise use our fabricated \hyxmp@createdate.
944 \ifx\@pdfmetadatetime\@empty
945     \hyxmp@add@simple{xmp:MetadataDate}{\hyxmp@createdate}%
946 \else
947     \hyxmp@add@simple{xmp:MetadataDate}{\@pdfmetadatetime}%
948 \fi
    Define the creation tool and the base URL.
949 \hyxmp@add@simple{xmp:CreatorTool}{\@pdfcreator}%
950 \hyxmp@add@simple{xmp:BaseURL}{\@baseurl}%
951 \hyxmp@add@to@xml{%
952 -----</rdf:Description>^^J%
953 }%
954 }

```

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.

```

955 \gdef\hyxmp@photoshop@schema{%

```

```

956 \edef\hyxmp@photoshop@data{\@pdfauthor\title\@pdfcaptionwriter}%
957 \ifx\hyxmp@photoshop@data\@empty
958 \else
959 \hyxmp@add@to@xml{%
960 -----<rdf:Description rdf:about=""^^J%
961 -----xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/">^^J%
962 }%
963 \fi
964 \hyxmp@add@simple{photoshop:AuthorsPosition}{\@pdfauthor\title}%
965 \hyxmp@add@simple{photoshop:CaptionWriter}{\@pdfcaptionwriter}%
966 \ifx\hyxmp@photoshop@data\@empty
967 \else
968 \hyxmp@add@to@xml{%
969 -----</rdf:Description>^^J%
970 }%
971 \fi
972 }

```

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.

```

973 \begingroup
974 \catcode'\&=12
975 \catcode'\#=12
976 \gdef\xmplinesep{&#xA;}
977 \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.

```

978 \newcommand*{\hyxmp@list@to@lines}[2]{%
979 \ifx#2\@empty
980 \else
981 \bgroup
982 \hyxmp@add@to@xml{%
983 -----<#1>%
984 }%

```

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

```

985 \def\@elt@first##1{%
986 \hyxmp@add@to@xml{##1}%
987 \let\@elt=\@elt@rest
988 }%

```

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

```

989 \def\@elt@rest##1{%
990 \hyxmp@add@to@xml{\xmplinesep##1}%

```

```

991      }%

\@elt  Re-encode the text from Unicode if necessary. Then redefine \@elt to insert a line
       separator between terms.
992      \let\@elt=\@elt@first
993      \hyxmp@xmllify{#2}%
994      \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmllified}%
995      \hyxmp@list
996      \hyxmp@add@to@xml{</#1>^^J}%
997      \egroup
998      \fi
999  }

```

```

\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.
1000 \gdef\hyxmp@photometa@schema{%
1001   \edef\hyxmp@photometa@data{%
1002     \@pdfcontactaddress
1003     \@pdfcontactcity
1004     \@pdfcontactregion
1005     \@pdfcontactpostcode
1006     \@pdfcontactcountry
1007     \@pdfcontactphone
1008     \@pdfcontactemail
1009     \@pdfcontacturl
1010   }%
1011   \ifx\hyxmp@photometa@data\@empty
1012   \else
1013     \hyxmp@iptc@extensions
1014     \hyxmp@add@to@xml{%
1015       <rdf:Description rdf:about="^^J%
1016       _____xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/">^^J%
1017       <Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">^^J%
1018     }%
1019     \fi
1020     \hyxmp@list@to@lines{Iptc4xmpCore:CiAdrExtadr}{\@pdfcontactaddress}%
1021     \hyxmp@add@simple{Iptc4xmpCore:CiAdrCity}{\@pdfcontactcity}%
1022     \hyxmp@add@simple{Iptc4xmpCore:CiAdrRegion}{\@pdfcontactregion}%
1023     \hyxmp@add@simple{Iptc4xmpCore:CiAdrPcode}{\@pdfcontactpostcode}%
1024     \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.

```

1025 \bgroup
1026   \def\xmplinesep{,}%
1027   \hyxmp@list@to@lines{Iptc4xmpCore: CiTelWork}{\@pdfcontactphone}%
1028   \hyxmp@list@to@lines{Iptc4xmpCore: CiEmailWork}{\@pdfcontactemail}%
1029   \hyxmp@list@to@lines{Iptc4xmpCore: CiUrlWork}{\@pdfcontacturl}%
1030 \egroup
1031 \ifx\hyxmp@photometa@data\empty
1032 \else
1033   \hyxmp@add@to@xml{%
1034   _____</Iptc4xmpCore:CreatorContactInfo>^^J%
1035   _____</rdf:Description>^^J%
1036   }%
1037 \fi
1038 }
```

`\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.

```

1039 \newcommand*{\hyxmp@iptc@extensions}{%
1040   \hyxmp@add@to@xml{%
1041   _____<rdf:Description rdf:about=""^^J%
1042   _____xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"^^J%
1043   _____xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema\hyxmp@hash"^^J%
1044   _____xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property\hyxmp@hash"^^J%
1045   _____xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type\hyxmp@hash"^^J%
1046   _____xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field\hyxmp@hash"^^J%
1047   _____<pdfaExtension:schemas>^^J%
1048   _____<rdf:Bag>^^J%
1049   _____<rdf:li rdf:parseType="Resource">^^J%
1050   _____<pdfaSchema:schema>IPTC Core Schema</pdfaSchema:schema>^^J%
1051   _____<pdfaSchema:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaSchema:
1052   _____<pdfaSchema:prefix>Iptc4xmpCore</pdfaSchema:prefix>^^J%
1053   _____<pdfaSchema:property>^^J%
1054   _____<rdf:Seq>^^J%
1055   _____<rdf:li rdf:parseType="Resource">^^J%
1056   _____<pdfaProperty:name>CreatorContactInfo</pdfaProperty:name>^^J%
1057   _____<pdfaProperty:valueType>contactinfo</pdfaProperty:valueType>^^J%
1058   _____<pdfaProperty:category>external</pdfaProperty:category>^^J%
1059   _____<pdfaProperty:description>contact information for the document's creator</p
1060   _____</rdf:li>^^J%
```



```

1061 _____</rdf:Seq>^^J%
1062 _____</pdfaSchema:property>^^J%
1063 _____<pdfaSchema:valueType>^^J%
1064 _____<rdf:Seq>^^J%
1065 _____<rdf:li rdf:parseType="Resource">^^J%
1066 _____<pdfaType:type>contactinfo</pdfaType:type>^^J%
1067 _____<pdfaType:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaType:
1068 _____<pdfaType:prefix>Iptc4xmpCore</pdfaType:prefix>^^J%
1069 _____<pdfaType:description>contact information</pdfaType:description>^^J%
1070 _____<pdfaType:field>^^J%
1071 _____<rdf:Seq>^^J%
1072 }%

1073 \hyxmp@text@resource{CiAdrExtadr}{contact address}%
1074 \hyxmp@text@resource{CiAdrCity}{contact city}%
1075 \hyxmp@text@resource{CiAdrRegion}{contact region}%
1076 \hyxmp@text@resource{CiAdrPcode}{contact postal code}%
1077 \hyxmp@text@resource{CiAdrCtry}{contact country}%
1078 \hyxmp@text@resource{CiTelWork}{contact telephone number}%
1079 \hyxmp@text@resource{CiEmailWork}{contact email address}%
1080 \hyxmp@text@resource{CiUrlWork}{contact url}%

1081 \hyxmp@add@to@xml{%
1082 _____</rdf:Seq>^^J%
1083 _____</pdfaType:field>^^J%
1084 _____</rdf:li>^^J%
1085 _____</rdf:Seq>^^J%
1086 _____</pdfaSchema:valueType>^^J%
1087 _____</rdf:li>^^J%
1088 _____</rdf:Bag>^^J%
1089 _____</pdfaExtension:schemas>^^J%
1090 _____</rdf:Description>^^J%
1091 }%
1092 }

```

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

```

1093 \newcommand*{\hyxmp@text@resource}[2]{%
1094   \hyxmp@add@to@xml{%
1095     _____<rdf:li rdf:parseType="Resource">^^J%
1096     _____<pdfaField:name>#1</pdfaField:name>^^J%
1097     _____<pdfaField:valueType>Text</pdfaField:valueType>^^J%
1098     _____<pdfaField:description>#2</pdfaField:description>^^J%
1099     _____</rdf:li>^^J%
1100   }
1101 }

```

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`.

```

1102 \newcommand*{\hyxmp@pdfa@id@schema}{%
1103   \ifHy@pdfa
1104     \hyxmp@add@to@xml{%
1105       <rdf:Description rdf:about="^^J%
1106         _____xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/">^^J%
1107       }%
1108       \hyxmp@add@simple{pdfaid:part}{\@pdfapart}%
1109       \hyxmp@add@simple{pdfaid:conformance}{\@pdfaconformance}%
1110       \hyxmp@add@to@xml{%
1111         </rdf:Description>^^J%
1112       }%
1113   \fi
1114 }

```

3.5.10 Combining schemata into an XMP packet

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

```

1115 \begingroup
1116   \ifhyxmp@unicodetex
1117     \lccode'\!="FEFF %
1118     \lowercase{%
1119       \gdef\hyxmp@bom{!}
1120     }%
1121   \else
1122     \catcode'\^^ef=12
1123     \catcode'\^^bb=12
1124     \catcode'\^^bf=12
1125     \gdef\hyxmp@bom{^^ef^^bb^^bf}%
1126   \fi
1127 \endgroup

```

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

`\hyxmp+xml`

```

1128 \def\hyxmp@construct@packet{%
1129   \gdef\hyxmp+xml{%
1130     \hyxmp@add@to@xml{<?xpacket begin="\hyxmp@bom" %
1131 id="W5M0MpCehiHzreSzNTczkc9d"?>^^J%
1132 <x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="3.1-702">^^J%
1133 ___<rdf:RDF
1134 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\hyxmp@hash">^^J%
1135   }%
1136   \hyxmp@pdf@schema
1137   \hyxmp@xmpRights@schema
1138   \hyxmp@dc@schema
1139   \hyxmp@photoshop@schema
1140   \hyxmp@photometa@schema
1141   \hyxmp@xmp@basic@schema

```

```

1142 \hyxmp@pdfa@id@schema
1143 \hyxmp@mm@schema
1144 \hyxmp@add@to@xml{%
1145 ___</rdf:RDF>^^J%
1146 </x:xmpmeta>^^J%
1147 \hyxmp@padding
1148 <?xpacket end="w"?>^^J%
1149 }%
1150 }

```

3.6 Embedding the XMP packet

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

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

```

1151 \newcommand*{\hyxmp@embed@packet}{%
1152 \hyxmp@construct@packet
1153 \def\hyxmp@driver{hpdftex}%
1154 \ifx\hyxmp@driver\Hy@driver
1155 \hyxmp@embed@packet@pdftex
1156 \else
1157 \def\hyxmp@driver{hluatex}%
1158 \ifx\hyxmp@driver\Hy@driver
1159 \hyxmp@embed@packet@luatex
1160 \else
1161 \def\hyxmp@driver{hdvipdfm}%
1162 \ifx\hyxmp@driver\Hy@driver
1163 \hyxmp@embed@packet@dvipdfm
1164 \else
1165 \def\hyxmp@driver{hxetex}%
1166 \ifx\hyxmp@driver\Hy@driver
1167 \hyxmp@embed@packet@xetex
1168 \else
1169 \@ifundefined{pdfmark}{%
1170 \PackageWarningNoLine{hyperxmp}{%
1171 Unrecognized hyperref driver ‘\Hy@driver’.\MessageBreak
1172 \jobname.tex’s XMP metadata will *not* be.\MessageBreak
1173 embedded in the resulting file}%
1174 }{%
1175 \hyxmp@embed@packet@pdfmark
1176 }%
1177 \fi
1178 \fi
1179 \fi
1180 \fi
1181 }

```

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.

```

1182 \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.

1183 \newcommand*{\hyxmp@embed@packet@pdftex}{%
1184   \bgroup
1185     \ifluatex
1186     \else
1187       \pdfcompresslevel=0
1188     \fi
1189     \immediate\pdfobj \ifluatex uncompressed\fi stream attr {%
1190       /Type /Metadata
1191       /Subtype /XML
1192     }{\hyxmp@xml}%
1193     \pdfcatalog {/Metadata \the\pdflastobj\space 0 R}%
1194   \egroup
1195 }
```

3.6.2 Embedding using LuaTeX 0.85+

```

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

1196 \newcommand*{\hyxmp@embed@packet@luatex}{%
1197   \immediate\pdfextension obj uncompressed stream attr {%
1198     /Type /Metadata
1199     /Subtype /XML
1200   }{\hyxmp@xml}%
1201   \pdfextension catalog {/Metadata \the\numexpr\pdffeedback lastobj\relax\space 0 R}%
1202 }
```

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.

```

1203 \newcommand*{\hyxmp@embed@packet@pdfmark}{%
1204   \pdfmark{%
1205     pdfmark=/NamespacePush
1206   }%
1207   \pdfmark{%
1208     pdfmark=/OBJ,
1209     Raw={/_objdef \string{hyxmp@Metadata\string} /type /stream}%
1210   }%
1211   \pdfmark{%
1212     pdfmark=/PUT,
1213     Raw={\string{hyxmp@Metadata\string}
1214       2 dict begin
1215         /Type /Metadata def
1216         /Subtype /XML def
1217         currentdict
1218       end
1219     }%
1220   }%
1221   \pdfmark{%
1222     pdfmark=/PUT,
1223     Raw={\string{hyxmp@Metadata\string} (\hyxmp@xml)}%
1224   }%
1225   \pdfmark{%
1226     pdfmark=/Metadata,
1227     Raw={\string{Catalog\string} \string{hyxmp@Metadata\string}}%
1228   }%
1229   \pdfmark{%
1230     pdfmark=/NamespacePop
1231   }%
1232 }
```

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.

```

1233 \newcommand*{\hyxmp@embed@packet@dvipdfm}{%
1234   \hyxmp@string@len{\hyxmp@xml}%
1235   \special{pdf: object @hyxmp@Metadata
1236     <<
1237       /Type /Metadata
1238       /Subtype /XML
1239       /Length \the\@tempcnta
1240     >>
```

```

1241     stream^^J\hyxmp@xml endstream%
1242 }%
1243 \special{pdf: docview
1244     <<
1245     /Metadata @hyxmp@Metadata
1246     >>
1247 }%
1248 }

```

`\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.

```

1249 \newcommand*{\hyxmp@string@len}[1]{%
1250     \@tempcnta=0
1251     \expandafter\hyxmp@count@spaces#1 {} %
1252     \expandafter\hyxmp@count@non@spaces#1{}%
1253 }

```

`\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.

```

1254 \def\hyxmp@count@spaces#1 {%
1255     \def\hyxmp@one@token{#1}%
1256     \ifx\hyxmp@one@token\empty
1257         \advance\@tempcnta by -1
1258     \else
1259         \advance\@tempcnta by 1
1260     \expandafter\hyxmp@count@spaces
1261 \fi
1262 }

```

`\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.

```

1263 \newcommand*{\hyxmp@count@non@spaces}[1]{%
1264     \def\hyxmp@one@token{#1}%
1265     \ifx\hyxmp@one@token\empty
1266     \else
1267         \advance\@tempcnta by 1
1268     \expandafter\hyxmp@count@non@spaces
1269 \fi
1270 }

```

3.6.5 Embedding using \XeTeX

`\hyxmp@embed@packet@xetex` Embed the XMP packet using `xdvipdfmx`-specific `\special` commands. I don't know how to tell `xdvipdfmx` always to leave the Metadata stream uncompressed,

so the XMP metadata is likely to be missed by non-PDF-aware XMP viewers.

```

1271 \newcommand*{\hyxmp@embed@packet@xetex}{%
1272   \special{pdf:stream @hyxmp@Metadata (\hyxmp@xml)
1273     <<
1274       /Type /Metadata
1275       /Subtype /XML
1276     >>
1277   }%
1278   \special{pdf:put @catalog
1279     <<
1280       /Metadata @hyxmp@Metadata
1281     >>
1282   }%
1283 }
```

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.

```

1284 \catcode'\="\hyxmp@dq@code
```

4 Future Work

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

A Sample XMP packet

The following is an example of a complete XMP packet as may be produced by `hyperxmp`. This packet corresponds to the metadata included in the sample \LaTeX document presented on 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>
            </rdf:li>
        </rdf:Bag>
    </pdfaExtension:schemas>

```

```

<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:descripti
          </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:descripti
          </rdf:li>
          <rdf:li rdf:parseType="Resource">
            <pdfaField:name>CiAdrPcode</pdfaField:name>
            <pdfaField:valueType>Text</pdfaField:valueType>
            <pdfaField:description>contact postal code</pdfaField:desc
          </rdf:li>
          <rdf:li rdf:parseType="Resource">
            <pdfaField:name>CiAdrCtry</pdfaField:name>
            <pdfaField:valueType>Text</pdfaField:valueType>
            <pdfaField:description>contact country</pdfaField:descript
          </rdf:li>
          <rdf:li rdf:parseType="Resource">
            <pdfaField:name>CiTelWork</pdfaField:name>

```

```

        <pdfaField:valueType>Text</pdfaField:valueType>
        <pdfaField:description>contact telephone number</pdfaField:
    </rdf:li>
    <rdf:li rdf:parseType="Resource">
        <pdfaField:name>CiEmailWork</pdfaField:name>
        <pdfaField:valueType>Text</pdfaField:valueType>
        <pdfaField:description>contact email address</pdfaField:de
    </rdf:li>
    <rdf:li rdf:parseType="Resource">
        <pdfaField:name>CiUrlWork</pdfaField:name>
        <pdfaField:valueType>Text</pdfaField:valueType>
        <pdfaField:description>contact url</pdfaField:description>
    </rdf:li>
    </rdf:Seq>
    </pdfaType:field>
    </rdf:li>
    </rdf:Seq>
    </pdfaSchema:valueType>
    </rdf:li>
    </rdf:Bag>
    </pdfaExtension:schemas>
    </rdf:Description>
    <rdf:Description rdf:about=""
        xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/">
    <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>
        http://einstein.biz/,
        https://www.facebook.com/AlbertEinstein
    </Iptc4xmpCore:CiUrlWork>
    </Iptc4xmpCore:CreatorContactInfo>
    </rdf:Description>
    <rdf:Description rdf:about=""
        xmlns:xmp="http://ns.adobe.com/xap/1.0/">
    <xmp:CreateDate>2017-07-25T22:08:56-06:00</xmp:CreateDate>
    <xmp:ModifyDate>2017-07-25T22:08:56-06:00</xmp:ModifyDate>
    <xmp:MetadataDate>2017-07-25T22:08:56-06:00</xmp:MetadataDate>
    <xmp:CreatorTool>LaTeX with hyperref package</xmp:CreatorTool>
    <xmp:BaseURL>
        http://mirror.ctan.org/macros/latex/contrib/hyperxmp/
    </xmp:BaseURL>

```

```

</rdf:Description>
<rdf:Description rdf:about=""
  xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/">
  <xmpMM:DocumentID>
    uuid:0595fdce-41dc-e4c4-6c418dc4ce46
  </xmpMM:DocumentID>
  <xmpMM:InstanceID>
    uuid:efd754c4-1d7f-200a-ef754ce413ea
  </xmpMM:InstanceID>
</rdf:Description>
</rdf:RDF>
</x:xmpmeta>
<?xpacket end="w"?>

```

References

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

- [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.
- [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.
- [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			strings as 8-bit strings before manipulating them into XMP
	General: Initial version	1	schema. This change addresses
v1.1			a bug reported by Martin
	\hyxmp@construct@packet:		Münch
	Explicitly set the category		General: Introduced the
	codes of characters $\langle EF \rangle$, $\langle BB \rangle$,		pdfmetalang package option,
	and $\langle BF \rangle$ to “letter”. Thanks		which enables an author to
	to Daniel Schömer for the bug		specify the language in which he
	report	50	wrote the document’s metadata
v1.2			19
	General: Added support for the		
	X _q TeX backend (xdvipdfmx)	1	v1.4
	Added support for the		\hyxmp@mm@schema: Renamed the
	Photoshop schema	1	xapMM namespace prefix to
	Made the package compatible		xmpMM
	with ngerman. Thanks to		44
	Tobias Mueller for the bug		\hyxmp@rdf@dc: Included metadata
	report.	12	in the x-default language
			regardless of the specified
			metadata language
v1.3			40
	\hyxmp@reencode: Introduced this		\hyxmp@xmpRights@schema:
	macro to re-encode Unicode		Renamed the xapRights
			namespace prefix to xmpRights
			43

v1.5	General: Made the XMP inclusion more robust. Thanks to Heiko Oberdiek for the bug report and suggested modifications. .	12	from line-wrapping the XMP packet	30
v2.0	\ProcessKeyvalOptions: Added this macro	18	\hyxmp@toxml@unicodetex: Added by Heiko Oberdiek	30
	\XMPTruncateList: Added this macro	22	\hyxmp@xetex@crap: Added by Heiko Oberdiek	31
	\hyxmp@ProcessKeyvalOptions: Added this macro	18	\hyxmp@xmlify: Completely rewritten by Heiko Oberdiek to better support Unicode-enabled T _E X programs	28
	\hyxmp@SpaceOther: Added by Heiko Oberdiek	32	\hyxmp@xmp@basic@schema: Added this macro	44
	\hyxmp@add@to@xml: Updated also to replace commas	37	\hyxmp@xmpRights@schema: Modified to include xmpRights:Marked only when pdfcopyright is specified and xmpRights:WebStatement only when pdflicenseurl is specified .	43
	\hyxmp@bom: Added by Heiko Oberdiek	50	\hyxmp@zero: Added by Heiko Oberdiek	33
	\hyxmp@comma: Added this macro .	21	\ifhyxmp@unicodetex: Added by Heiko Oberdiek	28
	\hyxmp@construct@packet: Modified by Heiko Oberdiek to use an appropriate BOM representation via \hyxmp@bom	50	\xmpcomma: Added this macro . . .	21
	\hyxmp@crap@convert: Added by Heiko Oberdiek	32	\xmpquote: Added this macro . . .	22
	\hyxmp@crap@test: Added by Heiko Oberdiek	32	General: Added support for the XMP Basic schema and miscellaneous other bits of metadata	1
	\hyxmp@dc@schema: Added support for dc:language and dc:source .	42	Heiko Oberdiek's major rewrite of the code to better support native-Unicode T _E X implementations (X _Y T _E X and LuaT _E X)	1
	\hyxmp@is@unicode: Added by Heiko Oberdiek	29	New \AtBeginDocument code from Heiko Oberdiek to properly encode \@pdfmetalang	19
	\hyxmp@list@to@xml: Modified by Heiko Oberdiek to use the new Unicode-processing macros . .	41	v2.1	
	\hyxmp@photoshop@schema: Simplified using \hyxmp@add@simple	45	\hypersetup: Added this macro .	18
	\hyxmp@reencode: Replaced with an empty macro by Heiko Oberdiek	28	\hyxmp@hypersetup: Added this macro	18
	\hyxmp@skiptorelax: Added by Heiko Oberdiek	32	\hyxmp@redefine@Hyp: Added this macro	16
	\hyxmp@skipzeros: Added by Heiko Oberdiek	31	General: Enabled hyperxmp and hyperref to be loaded in either order. This addresses a bug report by Yuri Donskoy	16
	\hyxmp@string: Added this macro	39	v2.2	
	\hyxmp@toxml: Added by Heiko Oberdiek	30	\hyxmp@iptc@extensions: Added this macro to support PDF/A	
	Escaped parentheses written with pdfmarks to prevent dvips			

generation	48	\hyxmp@pdf@schema: Made	
\hyxmp@list@to@lines: Added		\hyxmp@pdf@schema <i>always</i>	
this macro	46	include the Adobe PDF schema,	
\hyxmp@photometa@schema: Added		even when empty. Florian	
this macro	47	Breitwieser noted that this is	
\hyxmp@text@resource: Added		necessary for PDF/A-1b	
this macro	49	compliance	39
\xmpcomma: Changed the default		\hyxmp@pdf@to@xmp@date: Added	
from \relax to an ordinary		this macro	23
comma	21	\hyxmp@pdfa@id@schema: Added	
\xmplinesep: Added this macro	46	this macro	50
General: Added support for the		\hyxmp@today: Modified the code	
IPTC Photo Metadata schema	1	to parse the time and timezone	
v2.3		from \pdfcreationdate, as	
\hyxmp@iptc@extensions: Gave		proposed by Florian Breitwieser	26
the		\hyxmp@today@define: Added this	
lptc4xmpCore:CreatorContactInfo		macro	25
fields a unique pdfaType:prefix		\hyxmp@xmp@to@pdf@date: Added	
to better support conversion of		this macro	24
the document to PDF/A	48	\xmptilde: Added this macro	22
v2.3a		General: Added support for the	
General: Bug fix: Redefine		PDF/A Identification schema, as	
\@pdflang as \@empty when		requested by Florian	
hyperref has set it to \relax	19	Breitwieser	1
v2.3b		v2.5	
\XMPTruncateList: Made all		\hyxmp@add@to@xml: Updated also	
definitions local to avoid		to replace underscores and to	
spurious Too many		modify only the text being	
unprocessed floats errors		added, not the already-modified	
when running with memoir	22	text	37
v2.4		\hyxmp@textunderscore: Added	
\hyxmp@add@simple@var: Added		this macro	13
this macro	40	\hyxmp@uscore: Added this macro	21
\hyxmp@create@uuid: Modified		General: Enabled “_” to work	
this macro to produce a proper		within email addresses, as	
version 4 (random or		requested by Leonid Sinev	1
pseudorandom) UUID	36	v2.6	
\hyxmp@dc@schema: Made		General: Added support for a new	
dc:language a Bag instead of an		pdfdate key to explicitly specify	
individual item so as to		the document date (and	
conform to the latest XMP		optionally time)	1
specifications, a detail identified		v2.7	
by Florian Breitwieser	42	General: Automatically use \title	
\hyxmp@parse@time: Added this		and \author if pdftitle and	
macro	23	pdfauthor are left unspecified.	
\hyxmp@parse@tz: Added this		Thanks to Maciej Radziejewski	
macro	23	for the suggestion	20
\hyxmp@parse@tz@char: Added		v2.8	
this macro	23	\hyxmp@add@to@xml: Corrected	
		inadvertent lowercasing of	

non-Latin characters when run under X _Y L ^A T _E X or Lua ^A T _E X (bug reported by Leonid Sinev)	37	Updated to use <code>\pdfextension</code> <code>obj uncompressed</code> as suggested by Hans Hagen	52
v2.9		<code>\hyxmp@embed@packet@pdftex</code> : Leave the XMP packet—and only the XMP packet—uncompressed in both pdf _T E _X and pre-0.85 Lua _T E _X	52
<code>\hyxmp@pdfa@id@schema</code> : Let the author specify the PDF/A part and conformance IDs, as requested by Leonid Sinev	50		
<code>\hyxmp@photometa@schema</code> : Use <code>lptc4xmpCore</code> instead of <code>lptc4ContInfo</code> as the contact-information metadata prefix. Leonid Sinev reports that Acrobat's PDF/A validator seems to prefer <code>lptc4xmpCore</code>	47	v3.2	
General: Force inclusion of <code>dc:creator</code> , <code>dc:title</code> , and <code>dc:description</code> —even if empty—when <code>hyperref</code> is loaded with the <code>pdfa</code> option (suggested by Leonid Sinev)	1	<code>\hyxmp@as@pdf@date</code> : Added this macro	24
Introduced the <code>pdftype</code> package option, which enables an author to specify the type of document being produced	1	<code>\hyxmp@as@xmp@date</code> : Added this macro	22
v3.0		<code>\hyxmp@today@define</code> : Modified to include hours and minutes	25
<code>\hyxmp@embed@packet@luatex</code> : Added this macro	52	<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
<code>\hyxmp@today@define</code> : Modified to accept the name of a macro to define	25	v3.3	
<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>\@pdfsource</code> : Added this macro and the corresponding <code>pdfsource</code> option, at Niklas Beisert's request	15
General: Made the code compatible with Lua _T E _X 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	<code>\XMPLangAlt</code> : Added this macro based on a request—and some code—by Niklas Beisert to support metadata expressed in multiple languages	34
v3.1		<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
<code>\hyxmp@embed@packet@luatex</code> :		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

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.

[illegible]

E		
\EdefEscapeHex	415, 428	\hyxmp@alt@title 590, 599
\EdefUnescapeHex	... 432	\hyxmp@and 92
\EdefUnescapeString	402	\hyxmp@append@hex ..
ETX 21, 28	. 639, 658–660, 664
G		
Ghostscript 7	\hyxmp@append@hex@iii
H		
\Hy@driver 657, 663, 673, 684
. 4, 1154, 1158,		\hyxmp@append@hex@iv
1162, 1166, 1171	 662, 668,
\Hy@unicodedefalse	. 24, 35	669, 671, 686–688
hyperref 1, 3–6,	\hyxmp@as@pdf@date . 279
8, 11, 13, 16, 18–		\hyxmp@as@xmp@date .
20, 39, 51–53, 62–64		27, 38, 251, 931, 941
\hypersetup	130, 189, 195	\hyxmp@at@end ... 3, 198
hyperxmp 1–6,	\hyxmp@big@prime ...
8–16, 19–22, 28,		. 613, 616, 626, 636
35, 39, 52, 55, 62, 64		\hyxmp@big@prime@ii
\hyxmp@@is@unicode	. 436 613, 635
\hyxmp@add@simple ..		\hyxmp@bom .. 1115, 1130
. 749, 752, 758,		\hyxmp@comma
862, 890, 925,		. 63, 93, 114, 227
928, 930, 935,		\hyxmp@commas@to@list
938, 940, 945,		. 211, 242, 832, 994
947, 949, 950,		\hyxmp@commas@to@list@i
964, 965, 1021–	 213, 215
1024, 1108, 1109		\hyxmp@concat@metadata
\hyxmp@add@simple@var 744, 745, 768 135
\hyxmp@add@to@xml ..		\hyxmp@construct@packet
..... 702,	 1128, 1152
728, 740, 754,		\hyxmp@count@non@spaces
763, 772, 789,	 1252, 1263
796, 800, 805,		\hyxmp@count@spaces
810, 826, 834,	 1251, 1254
840, 847, 864,		\hyxmp@crap@convert
880, 886, 891,	 518, 552
899, 919, 951,		\hyxmp@crap@result ..
959, 968, 982,	 508, 544
986, 990, 996,		\hyxmp@crap@test 515, 540
1014, 1033, 1040,		\hyxmp@create@uuid .
1081, 1094, 1104,	 666, 694, 700
1110, 1130, 1144		\hyxmp@createdate ..
\hyxmp@alt@description 590, 600 907, 925,
\hyxmp@alt@rights ..		928, 935, 938, 945
..... 590, 601		\hyxmp@cur@lang 596, 604
		\hyxmp@dc@schema ...
	 846, 1138
		\hyxmp@def@DocumentID
	 690, 897
		\hyxmp@def@InstanceID
	 696, 898
		\hyxmp@define@createdate
	 907, 923
		\hyxmp@DocumentID ..
	 690, 902
		\hyxmp@dq@code . 1, 1284
		\hyxmp@driver .. 3, 1151
		\hyxmp@embed@packet
	 200, 1151
		\hyxmp@embed@packet@dvi
	 1163, 1233
		\hyxmp@embed@packet@luatex
	 1159, 1196
		\hyxmp@embed@packet@pdfmark
	 1175, 1203
		\hyxmp@embed@packet@pdftex
	 1155, 1183
		\hyxmp@embed@packet@xetex
	 1167, 1271
		\hyxmp@find@metadata
	 135, 199
		\hyxmp@first@char .. 249
		\hyxmp@first@char@i
	 249, 252, 280
		\hyxmp@gobbletwo ... 319
		\hyxmp@hash 722,
		1043–1046, 1134
		\hyxmp@Hyp@pdfauthor 86
		\hyxmp@Hyp@pdfkeywords
	 107
		\hyxmp@hypersetup .. 130
		\hyxmp@InstanceID ..
	 696, 903
		\hyxmp@iptc@extensions
	 1013, 1039
		\hyxmp@is@unicode ..
	 404, 421, 436
		\hyxmp@LA@accept ...
	 593, 599–601
		\hyxmp@legal 869
		\hyxmp@list
		. 832, 838, 994, 995
		\hyxmp@list@to@lines
	 978,
		1020, 1027–1029
		\hyxmp@list@to@xml .
	 816, 855–859
		\hyxmp@mm@schema ...
	 896, 1143
		\hyxmp@modulo@a 607,
		626, 636, 642, 677

\hyxmp@new@xml	718, 719	\hyxmp@seed@rng@i ..	\hyxmp@xmllified	400,
\hyxmp@num	552 617, 619	764, 773, 788,	
\hyxmp@one@token	615, 619, 1255,	\hyxmp@seed@string .	797, 806, 832, 994	
1256, 1264, 1265		. 691, 692, 697, 698	\hyxmp@xmllify
\hyxmp@padding	726, 1147	\hyxmp@set@rand@num 181, 400,	
\hyxmp@parse@time	260, 262 632, 640, 675	762, 771, 787,	
..... 260, 262		\hyxmp@skiptorelax .	795, 804, 831, 993	
\hyxmp@parse@tz	269, 272, 276 545, 551	\hyxmp@xmp@basic@schema 918, 1141
..... 269, 272, 276		\hyxmp@skipzeros ... 503	\hyxmp@xmp@to@pdf@date 283, 286
\hyxmp@parse@tz@char	264, 266 512, 525	\hyxmp@xmp@to@pdf@date@i 287, 289
..... 264, 266		\hyxmp@string 287, 289	
\hyxmp@pdf@schema	739, 1136	\hyxmp@string@len ..	\hyxmp@xmp@to@pdf@date@i 292, 295
..... 739, 1136	 1234, 1249	\hyxmp@xmp@to@pdf@date@i 298, 301
\hyxmp@pdf@to@xmp@date	253, 258,	\hyxmp@sublist 298, 301	
376, 379, 912, 915		. 216, 217, 220, 221	\hyxmp@xmp@to@pdf@date@iv 304, 307
\hyxmp@pdfa@id@schema	1102, 1142	\hyxmp@temp@list ... 240	\hyxmp@xmp@to@pdf@date@v 310, 313
..... 1102, 1142		\hyxmp@temp@str ... 240	\hyxmp@xmp@to@pdf@date@vi 316, 320
\hyxmp@pdfauthor	83, 92, 855	\hyxmp@text	\hyxmp@xmp@to@pdf@date@vii 323, 326, 336
..... 83, 92, 855		. 400, 478, 508, 552	\hyxmp@xmp@to@pdf@date@viii 339, 342
\hyxmp@pdfkeywords	83, 113, 856	\hyxmp@text@resource	\hyxmp@xmpRights@schema 868, 1137
..... 83, 113, 856		.. 1073–1080, 1093	\hyxmp@zero	561,
\hyxmp@pdfstringdef	15,	\hyxmp@textunderscore	568, 575, 581, 586	
..... 15,	 15		
26, 37, 44, 46, 48,		\hyxmp@today		
50, 52, 54, 56, 58,	 184, 372, 697, 857		
60, 65, 70, 72, 74,		\hyxmp@today@define		
76, 78, 80, 82, 595	 345, 374, 910		
\hyxmp@photometa@data	1000	\hyxmp@toxml		
..... 1000		.. 430, 453		
\hyxmp@photometa@schema	1000, 1140	\hyxmp@toxml@unicodetex		
..... 1000, 1140	 418, 478		
\hyxmp@photoshop@data	955	\hyxmp@trimb		
..... 955		.. 385, 388		
\hyxmp@photoshop@schema	955, 1139	\hyxmp@trimc		
..... 955, 1139		.. 388, 389		
\hyxmp@ProcessKeyvalOptions	125	\hyxmp@trimspaces		
..... 125	 220, 381		
\hyxmp@rand@num	632,	\hyxmp@try		
641, 676, 693, 699	 508		
\hyxmp@rdf@dc	777, 852–854	\hyxmp@unicodetexfalse		
..... 777, 852–854	 391		
\hyxmp@redefine@Hyp	85, 127, 132	\hyxmp@unicodetexttrue		
..... 85, 127, 132	 391		
\hyxmp@reencode	397	\hyxmp@uscore		
\hyxmp@rights	869, 872, 876, 878	.. 17, 231		
..... 869, 872, 876, 878		\hyxmp@value		
\hyxmp@seed@rng	615, 692, 698	.. 595, 788		
..... 615, 692, 698		\hyxmp@x@default		
		. 176, 738, 793, 801		
		\hyxmp@xetex@crap		
	 409, 508		
		\hyxmp+xml		
		719, 726, 1128,		
		1192, 1200, 1223,		
		1234, 1241, 1272		

I

IETF	5
\if@tempwa	786, 825
\iffalse	777, 816
\ifHy@pdfa	852, 853, 855, 1103
\ifhyxmp@unicodetex	391, 403, 705, 1116
ifluatex	52
\ifluatex	1185, 1189
ifxetex	13
\ifxetex	408
Info	7
intcalc	13
\intcalcDiv	557, 564, 571
\intcalcMod	559, 566, 573
IPTC	9, 15, 37, 47, 48, 63

lptc4xmpCore:CreatorContactInfo 2, 47, 63
 ISO 12, 14, 33

J

\jobname . . . 59, 163,
 204, 691, 697, 1172

K

Keywords 7, 39
 \KV@Hyp@pdfauthor . . 92
 \KV@Hyp@pdfkeywords 113
 kvoptions 13, 18

L

LF 46
 Lua^LTeX 7,
 10, 26, 27, 39, 44, 64
 LuaTeX 28,
 31, 52, 55, 62, 64

M

memoir 63
 Metadata 7, 51, 54
 MetadataDate 44, 64
 ModifyDate 44, 64
 \month 347, 348, 350

N

NAK 13, 21, 28
 nativepdf (option) . . . 53
 \newif 391
 \next . . . 25, 36, 215, 619
 ngerman 12, 61
 \number 555, 557, 559,
 564, 566, 571, 573
 \numexpr 1201

O

options
 baseurl . . 3, 11, 13, 44
 dvipdf 53
 dvips 53
 dvipsone 53
 diviwindo 53
 nativepdf 53
 pdfa 5, 64
 pdfaconformance 4, 5, 50
 pdfapart 4, 5, 50

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, 13, 63
 pdfkeywords 3, 10, 13, 16, 42
 pdflang 3, 6,
 11, 13, 19, 33, 42, 64
 pdflicenseurl 4, 5, 11, 43, 62
 pdfmark 53
 pdfmetadata 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 53
 textures 53
 unicode 11, 64
 vtexpdfmark 53

P

\PackageWarningNoLine 162, 203, 1170
 PDF 1–3, 5,
 7–10, 13, 14, 20,
 22–24, 30, 37, 39,
 44, 50–52, 55, 63, 64
 PDF/A 3, 5, 15,
 37, 39, 48–50, 62–64
 pdf:Keywords 2, 39
 pdf:PDFVersion 3, 39
 pdf:Producer 3, 39
 pdfa (option) 5, 64
 pdfaconformance (op-
 tion) 4, 5, 50
 pdfaid:conformance . . . 3
 pdfaid:part 3
 pdfapart (option) . . 4, 5, 50
 pdfaType:prefix 63
 pdfauthor (option) 3, 10, 11,
 13, 16, 17, 20, 42, 63
 pdfauthortitle (option)
 4, 5, 11
 pdfcaptionwriter (op-
 tion) 4, 5
 \pdfcatalog 1193
 \pdfcompresslevel . . 1187
 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 376, 912
 pdfdate (option) 4, 5, 11, 13, 63
 PDFDocEncoding 16, 28, 29
 pdfescape 13
 \pdfextension 1197, 1201
 \pdffeedback 379, 915, 1201
 pdfkeywords (option) 3, 10, 13, 16, 42

pdflang (option) 3, 6, 11, 13, 19, 33, 42, 64			
\pdflastobj 1193			
pdfL ^A T _E X			
3, 7, 10, 26, 39, 44			
pdflicenseurl (option) .			
. . . 4, 5, 11, 43, 62			
pdfmark (option) 53			
\pdfmark 1204,			
1207, 1211,			
1221, 1225, 1229			
pdfmetadata (option) .			
. 4, 5, 14, 64			
pdfmetalang (option) .			
4–6, 11, 33, 61, 64			
\pdfminorversion . . . 749			
pdfmoddate (option) .			
. 4, 5, 64			
\pdfobj 1189			
pdfproducer (option) 4, 13			
pdfsource (option) 4, 6, 64			
\pdfstringdef 18			
pdfsubject (option) . .			
. 4, 13, 42			
pdfT _E X 12, 30, 52, 55, 64			
pdftitle (option) . . 4,			
11, 13, 20, 42, 63			
pdftype (option) . 4, 6, 64			
\pdfvariable 752			
photoshop:AuthorsPosition			
. 3, 45			
photoshop:CaptionWriter			
. 3, 45			
PI 37			
\ProcessKeyvalOptions			
. <u>125</u>			
Producer 39			
ps2pdf (option) 53			
Q			
\Q 381, 390			
	R		
	RDF 41		
	rdf:li 2		
	rdf:Seq 2		
	\renewcommand 126		
	\RequirePackage . . .		
	. . . 7, 10–14, 1182		
	S		
	\SE->pdfdoc@03 398		
	\SE->pdfdoc@15 399		
	\setkeys 605		
	\special 1235,		
	1243, 1272, 1278		
	stringenc 13		
	\StringEncodingConvert		
 405,		
	411, 422, 425, 520		
	Subject 7		
	T		
	T _E X 28, 30, 31,		
	34, 35, 44, 54, 55, 62		
	Text 49		
	\textunderscore . . .		
 16, 17, 19		
	textures (option) 53		
	\time 357, 365		
	Title 7		
	U		
	Unicode . . 11, 13, 28–		
	32, 42, 47, 50, 55, 62		
	unicode (option) . . 11, 64		
	URL 2, 5,		
	11, 14, 16, 43–45, 47		
	UTF-16BE 29		
	UTF-32BE 28, 29		
	UTF-8 29		
	UUID 34, 36, 37, 63		
		V	
		\vfuzz 389	
		vtexpdfmark (option) . 53	
		X	
		\x <u>508</u>	
		xdvipdfmx 10, 54	
		X _L ^A T _E X 7,	
		10, 26, 39, 44, 64	
		X _F ^A T _E X 13,	
		28, 31, 54, 55, 61, 62	
		XML 1, 2,	
		9, 20, 28–31, 37,	
		39–42, 46, 48, 50, 64	
		xmp 1–3, 5–11, 13, 14,	
		19–24, 26, 27, 30,	
		33, 34, 37–41, 43,	
		44, 48, 52–55, 61–64	
		xmp:BaseURL 2	
		xmp:CreateDate 2	
		xmp:CreatorTool 3	
		xmp:MetadataDate 2	
		xmp:ModifyDate 2	
		\xmpcomma	
		63, 66, <u>92</u> , <u>113</u> , <u>226</u>	
		xmpincl 3	
		\XMLLangAlt <u>602</u>	
		\xmplinesep <u>973</u> , 990, <u>1025</u>	
		xmpMM:DocumentID .	
	 2, 34, 44	
		xmpMM:InstanceId . .	
	 2, 34, 44	
		\xmpquote	
		64, 67, <u>92</u> , <u>113</u> , <u>235</u>	
		xmpRights:Marked 2, 43, 62	
		xmpRights:WebStatement	
	 2, 43, 62	
		\xmptilde <u>236</u>	
		\XMPTruncateList . . . <u>240</u>	
		Y	
		\year 346	