

The hyperxmp package^{*}

Scott Pakin
scott+hyxmp@pakin.org

February 25, 2019

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` v3.5a, dated 2019/02/25.

```

    </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
- pdfdocumentid
- pdfinstanceid
- 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.

XMP can include a universally unique identifier (UUID) for each document and for each instance of a given document. By default, `hyperxmp` assigns a version 4 (i.e., pseudorandom) UUID [9] for each of these. However, a document can alternatively specify a particular document iden-

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

tifier using `pdfdocumentid` and (not normally recommended) a particular instance identifier using `pdfinstanceid`. These should be of the form `uuid:xxxxxxxx-xxx-xxx-xxx-xxxxxxxxxxxx`, where “x” is a lowercase hexadecimal number. For example, `uuid:53ab7f19-a48c-5177-8bb2-403ad907f632` is a valid argument to `pdfdocumentid` (or `pdfinstanceid`). See Leach, Mealling, and Salz’s UUID specification document for details on how to produce the various forms of UUIDs [9].

`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, and so forth. If `pdfmetalang` is not specified, `hyperxmp` assumes the metadata language is the same as the document language (`hyperref`’s `pdflang` option). If neither `pdfmetalang` nor `pdflang` is specified, `hyperxmp` uses only “x-default” as the metadata language. Note that “x-default” metadata are always included in addition to the specified metadata language, as the user reading the document may not have specified a language preference.

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

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

It is usually more convenient to provide values for those options using `hyperref`’s `\hypersetup` command than on the `\usepackage` command line. See the `hyperref` manual for more information. The following is a sample L^AT_EX 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
},
pdfdocumentid={uuid:6d1ac9ec-4ff2-515a-954b-648eeb4853b0},
pdflang={en},
pdfmetalang={en},
baseurl={http://mirror.ctan.org/macros/latex/contrib/hyperxmp/}
}
\XMLLangAlt{de}{pdftitle={Über einen die Erzeugung und Verwandlung
    des Lichtes betreffenden heuristischen Gesichtspunkt}}
\begin{document}
\maketitle
A profound formal difference exists between the theoretical
concepts that physicists have formed about gases and other
ponderable bodies, and Maxwell's theory of electromagnetic
processes in so-called empty space\dots
\end{document}

```

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

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

Unfortunately, the \LaTeX + Dvips + Ghostscript path doesn't work. Ghostscript bug report #690066, closed with "WONTFIX" status on 2012-05-28, explains that Ghostscript doesn't honor the Metadata tag needed to inject a custom XMP packet.

Instead, Ghostscript fabricates an XMP packet of its own based on the metadata it finds in the PDF file’s Info dictionary (Author, Title, Subject, and Keywords).

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

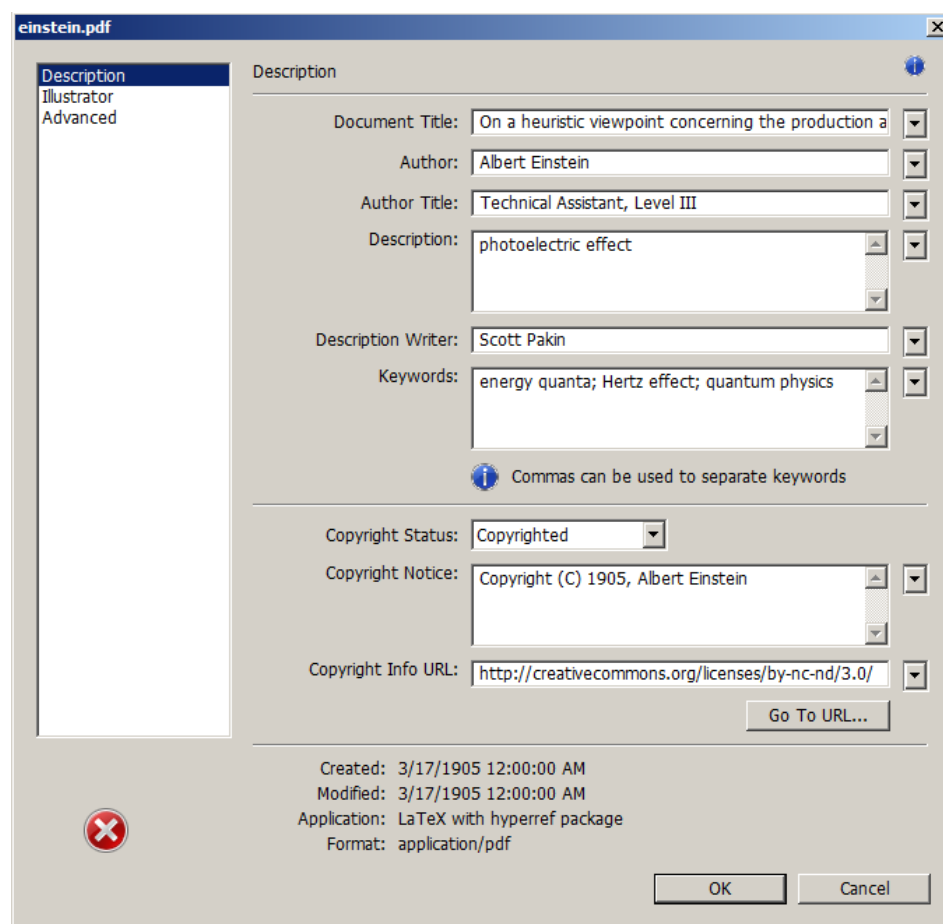


Figure 1: XMP metadata as it appears in Adobe Acrobat

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)

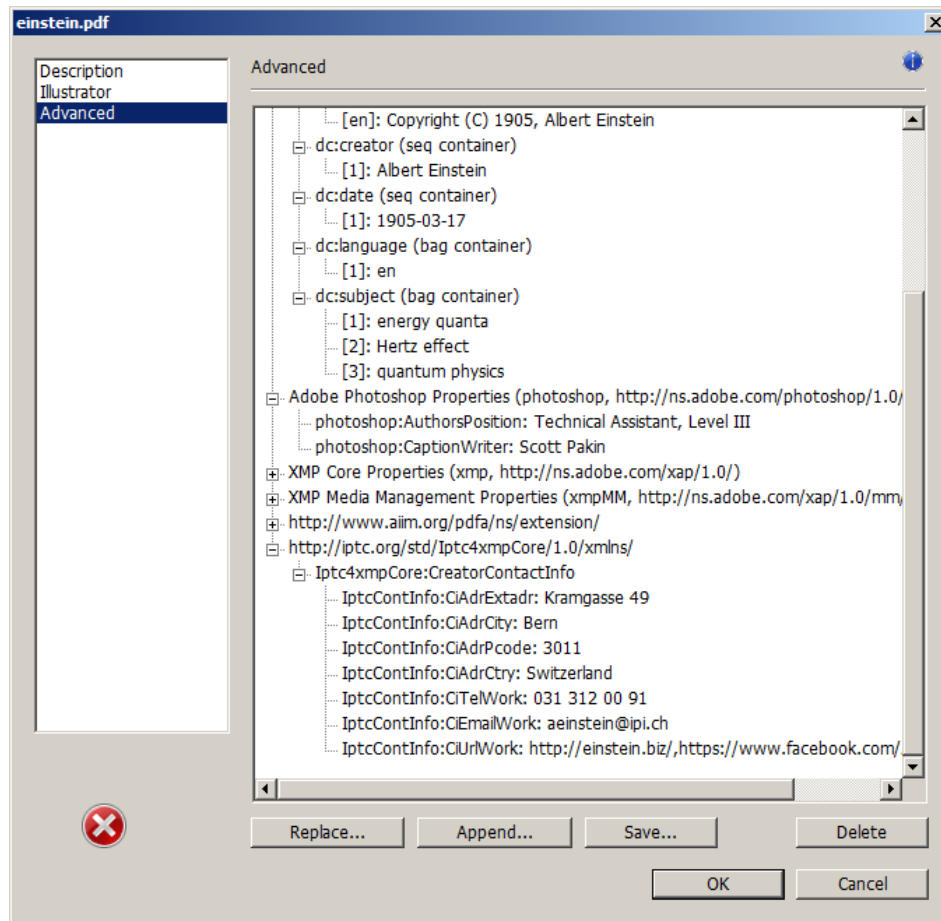


Figure 2: Additional XMP metadata as it appears in Adobe Acrobat

while the remaining authors are displayed from the XMP data (`dc:creator`). For example, suppose that a document’s authors are Jack Napier, Edward Nigma, and Harvey Dent. When displaying the document properties, Adobe Acrobat replaces “Jack Napier” with a single author named “Jack Napier, Edward Nigma, Harvey Dent” and leaves “Edward Nigma” and “Harvey Dent” as the second and third authors, respectively.

`\XMPTruncateList`

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

```
\XMPTruncateList{pdfauthor}
```

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

Note 2: Acrobat multiline-field bug The IPTC Photo Metadata schema states that “the [contact] address is a multiline field” [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, 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:

`\xmplinesep`

```
\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.

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}}
```

Note 8: Expandable arguments All arguments passed to `hyperxmp` options must be expandable, in T_EX terminology. This implies that any macros that are used in arguments are limited to a relatively small set of operations (such as conditionals and macro expansion) and must produce a string of text. Code (such as macro definitions and arithmetic operations) will be written to XMP as code, not as the result of executing the code.

By way of example, the macros provided by the `texdate` package for typesetting dates are not expandable (at least at the time of this writing). Hence, the `\printfddate{Y}` in the following code snippet is not replaced by the current year, as one might expect:

```
\usepackage{texdate}
\initcurrdate
\hypersetup{%
  pdfcopyright={Copyright \textcopyright\ \printfddate{Y}, Scott Pakin}
}
```

Rather, it generates a `dc:rights` tag of the form “Copyright © =2=0=by-1by=02019, Scott Pakin”. The garbage in that line corresponds to the remnants of the `\printfdate` code after expanding all of the \TeX primitives and certain other control sequences it uses to the empty string. For example, “`\global\advance\texdyr by-1`” expands to “`by-1`”.

It is not possible to determine a priori whether or not a macro is expandable. The best advice is to carefully inspect the XMP package in the output file to ensure that any macros used in arguments to `hyperxmp` options produced the expected output.

3 Implementation

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

3.1 Initial preparation

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

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

`\hyxmp@at@end` The `\hyxmp@at@end` macro includes code at the end of the document. For `pdf \TeX` , the standard `\AtEndDocument` works well enough. For all the other backends we use `\AtEndDvi` from the `atenddvi` package, which is more robust but requires an additional \LaTeX 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 \LaTeX ’s document-metadata naming

conventions), we do not prefix metadata-related macro names with our package-specific `\hyxmp@` prefix. That is, we use names like `\@pdfcopyright` instead of `\hyxmp@pdfcopyright`.

We load a bunch of helper packages: `kvoptions` for package-option processing, `pdfescape` and `stringenc` for re-encoding Unicode strings, `intcalc` for performing integer calculations (division and modulo), `ifxetex` for detecting X_YTeX, and `ifmtarg` for testing if a macro argument is empty or all spaces.

```

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

\ifmtargexp \@ifmtarg and \@ifnotmtarg do not expand their first argument. Define
\ifnotmtargexp \@ifmtargexp and \@ifnotmtargexp as expanding versions of those macros.
16 \def\@ifmtargexp#1{\expandafter\@ifmtarg\expandafter{#1}}
17 \def\@ifnotmtargexp#1{\expandafter\@ifnotmtarg\expandafter{#1}}

\hyxmp@pdfstringdef \hyxmp@textunderscore Because hyperxmp uses underscores to represent hard spaces, we need “\_” to map
initially to something other than an underscore, in particular the ASCII NAK (^~U)
character. To accomplish this, we wrap hyperref’s \pdfstringdef macro with our
own version that temporarily does the proper substitution. Later in the execution,
after underscores have been replaced with spaces, we replace NAK characters with
underscores.
18 \newcommand{\hyxmp@pdfstringdef}[2]{%
19 \let\hyxmp@textunderscore=\textunderscore
20 \let\textunderscore=\hyxmp@uscore
21 \pdfstringdef{#1}{#2}%
22 \let\textunderscore=\hyxmp@textunderscore
23 }

\@pdfdatetime Prepare to store the document’s date and (optionally) time. Whether specified
by the author in XMP format or PDF format (cf. Section 3.3.2) we always store
\@pdfdatetime as an XMP-format string.
24 \def\@pdfdatetime{}
25 \define@key{Hyp}{pdfdate}{%
26 \begingroup
27 \Hy@unicodedefalse

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

`\@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.

```

35 \def\@pdfmetadatetime{}
36 \define@key{Hyp}{pdfmetadate}{%
37   \begingroup
38     \Hy@unicodetofalse

```

`\next` Expand `pdfmetadate`’s argument and convert it to XMP format.

```

39   \edef\next{%
40     \noexpand\hyxmp@pdfstringdef\noexpand\@pdfmetadatetime{%
41       \noexpand\hyxmp@as@xmp@date{#1}}%
42   }%
43   \next
44 \endgroup
45 }

```

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

```

46 \def\@pdfcopyright{}
47 \define@key{Hyp}{pdfcopyright}{\hyxmp@pdfstringdef\@pdfcopyright{#1}}

```

`\@pdftype` Prepare to store the document’s logical type, which defaults to “Text”.

```

48 \def\@pdftype{Text}
49 \define@key{Hyp}{pdftype}{\hyxmp@pdfstringdef\@pdftype{#1}}

```

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

```

50 \def\@pdflicenseurl{}
51 \define@key{Hyp}{pdflicenseurl}{\hyxmp@pdfstringdef\@pdflicenseurl{#1}}

```

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

```

52 \def\@pdfauthortitle{}
53 \define@key{Hyp}{pdfauthortitle}{\hyxmp@pdfstringdef\@pdfauthortitle{#1}}

```

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

```

54 \def\@pdfcaptionwriter{}
55 \define@key{Hyp}{pdfcaptionwriter}{\hyxmp@pdfstringdef\@pdfcaptionwriter{#1}}

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

62 \edef\@pdfsource{\jobname.tex}
63 \define@key{Hyp}{pdfsource}{\hyxmp@pdfstringdef\@pdfsource{#1}}

```

`\hyxmp@DocumentID` Prepare to store a UUID that represents the document.

```

64 \def\hyxmp@DocumentID{}
65 \define@key{Hyp}{pdfdocumentid}{\hyxmp@pdfstringdef\hyxmp@DocumentID{#1}}

```

`\hyxmp@InstanceID` Prepare to store a UUID that represents the current instance of the document.

```

66 \def\hyxmp@InstanceID{}
67 \define@key{Hyp}{pdfinstanceid}{\hyxmp@pdfstringdef\hyxmp@InstanceID{#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.

```

68 \def\@pdfcontactaddress{}
69 \define@key{Hyp}{pdfcontactaddress}{%
70   \let\xmpcomma=\hyxmp@comma
71   \def\xmpquote##1{##1}%
72   \hyxmp@pdfstringdef\@pdfcontactaddress{#1}%
73   \def\xmpcomma{,}%
74   \let\xmpquote=\relax
75 }

```

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

```

76 \def\@pdfcontactcity{}
77 \define@key{Hyp}{pdfcontactcity}{\hyxmp@pdfstringdef\@pdfcontactcity{#1}}

```

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

```

78 \def\@pdfcontactregion{}
79 \define@key{Hyp}{pdfcontactregion}{\hyxmp@pdfstringdef\@pdfcontactregion{#1}}

```

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

```

80 \def\@pdfcontactpostcode{}
81 \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> 82 \def\@pdfcontactcountry{} 83 \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> 84 \def\@pdfcontactphone{} 85 \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> 86 \def\@pdfcontactemail{} 87 \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> 88 \def\@pdfcontacturl{} 89 \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> 90 \def\hyxmp@pdfauthor{} 91 \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> 92 \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> 93 \@ifundefined{KV@Hyp@pdfauthor}{}{% 94 \@ifundefined{hyxmp@Hyp@pdfauthor}{}% 95 \expandafter\let\expandafter\hyxmp@Hyp@pdfauthor 96 \csname KV@Hyp@pdfauthor\endcsname 97 }{}% 98 }% </pre>

\KV@Hyp@pdfauthor \xmpcomma \xmpquote \hyxmp@and \and \hyxmp@pdfauthor \@pdfauthor	<p>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 \xmpcomma is defined as an ordinary comma, and \xmpquote is defined as a macro that puts its argument within double quotes. The result is stored in \@pdfauthor for use in unstructured lists (those in which the entire list appears within a single pair of tags). 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.</p> <pre> 99 \define@key{Hyp}{pdfauthor}{% 100 \let\xmpcomma=\hyxmp@comma 101 \def\xmpquote####1{####1}% 102 \let\hyxmp@and=\and 103 \def\and{,}% 104 \hyxmp@Hyp@pdfauthor{##1}% 105 \global\let\hyxmp@pdfauthor=\@pdfauthor 106 \def\and{and\space}% 107 \def\xmpcomma{,}% 108 \def\xmpquote####1{"####1"% 109 \hyxmp@Hyp@pdfauthor{##1}% 110 \def\xmpcomma{,}% 111 \let\xmpquote=\relax 112 \let\and=\hyxmp@and 113 }% </pre>
\hyxmp@Hyp@pdfkeywords	<p>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.</p> <pre> 114 \@ifundefined{KV@Hyp@pdfkeywords}{}{% 115 \ifundefined{hyxmp@Hyp@pdfkeywords}{% 116 \expandafter\let\expandafter\hyxmp@Hyp@pdfkeywords 117 \csname KV@Hyp@pdfkeywords\endcsname 118 }{}% 119 }% </pre>
\KV@Hyp@pdfkeywords \xmpcomma \xmpquote \hyxmp@pdfkeywords \@pdfkeywords	<p>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).</p> <pre> 120 \define@key{Hyp}{pdfkeywords}{% 121 \let\xmpcomma=\hyxmp@comma 122 \def\xmpquote####1{####1}% 123 \hyxmp@Hyp@pdfkeywords{##1}% 124 \global\let\hyxmp@pdfkeywords=\@pdfkeywords </pre>

```

125 \def\xmpcomma{,}%
126 \def\xmpquote####1{"####1"}%
127 \hyxmp@Hyp@pdfkeywords{##1}%
128 \def\xmpcomma{,}%
129 \let\xmpquote=\relax
130 }%
131 }

```

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

```

132 \let\hyxmp@ProcessKeyvalOptions=\ProcessKeyvalOptions
133 \renewcommand*{\ProcessKeyvalOptions}{%
134 \hyxmp@redefine@Hyp
135 \hyxmp@ProcessKeyvalOptions
136 }

```

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

```

\hypersetup
137 \let\hyxmp@hypersetup=\hypersetup
138 \def\hypersetup{%
139 \hyxmp@redefine@Hyp
140 \hyxmp@hypersetup
141 }

```

`\hyxmp@find@metadata` Issue a warning message if the author failed to specify any metadata at all. This excludes metadata that is included automatically such as the current timestamp. 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.

`\hyxmp@concat@metadata`

```

142 \newcommand*{\hyxmp@find@metadata}{%
143 \edef\hyxmp@concat@metadata{%
144 \@baseurl
145 \@pdfauthor
146 \@pdfauthortitle
147 \@pdfcaptionwriter
148 \@pdfcontactaddress
149 \@pdfcontactcity
150 \@pdfcontactcountry
151 \@pdfcontactemail
152 \@pdfcontactphone
153 \@pdfcontactpostcode
154 \@pdfcontactregion
155 \@pdfcontacturl
156 \@pdfcopyright
157 \@pdfcreationdate
158 \@pdfdatetime
159 \@pdfkeywords
160 \@pdflang
161 \@pdflicenseurl

```

```

162     \@pdfmetadatetime
163     \@pdfmoddate
164     \@pdfsubject
165     \@pdftitle
166     \@pdftype
167 }%
168 \ifx\hyxmp@concat@metadata\@empty
169     \PackageWarningNoLine{hyperxmp}{%
170 \jobname.tex did not specify any metadata to\MessageBreak
171 include in the XMP packet.\space\space Please see the\MessageBreak
172 hyperxmp documentation for instructions on how to\MessageBreak
173 provide metadata values to hyperxmp}%
174 \fi
175 }

```

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.

```

176 \AtBeginDocument{%
177     \@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`.

```

178     \ifx\@pdflang\relax
179         \let\@pdflang=\@empty
180     \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.

```

181     \ifx\@pdfmetalang\@empty
182         \ifx\@pdflang\@empty
183             \let\@pdfmetalang=\hyxmp@x@default
184         \else
185             \edef\@pdfmetalang{\@pdflang}%
186         \fi
187     \fi
188     \hyxmp@xmlify\@pdfmetalang

```

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

```

189     \ifx\@pdfdatetime\@empty
190     \else
191         \edef\hyxmp@today{\@pdfdatetime}%
192     \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`.

```

193 \ifmtargexp{\@pdftitle}{%
194 \ifnotmtargexp{\@title}{%
195 \hypersetup{pdftitle={\@title}}}%
196 }%
197 }%
198 {}%
199 \ifmtargexp{\@pdfauthor}{%
200 \ifnotmtargexp{\@author}{%
201 \hypersetup{pdfauthor={\@author}}}%
202 }%
203 }%
204 {}%
```

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`.

```

205 \hyxmp@at@end{%
206 \hyxmp@find@metadata
207 \hyxmp@embed@packet
208 }%
209 }{%
210 \PackageWarningNoLine{hyperxmp}{%
211 \jobname.tex failed to include a\MessageBreak
212 \string\usepackage\string{hyperref\string}
213 in the preamble.\MessageBreak
214 Consequently, all hyperxmp functionality will be\MessageBreak
215 disabled}%
216 }%
217 }
```

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.)

```

218 \newcommand*{\hyxmp@commas@to@list}[2]{%
219   \gdef#1{%
220     \expandafter\hyxmp@commas@to@list@i\expandafter#1#2,,%
221   }

```

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

```

\next 222 \def\hyxmp@commas@to@list@i#1#2,{%
223   \gdef\hyxmp@sublist{#2}%
224   \ifx\hyxmp@sublist\@empty
225     \let\next=\relax
226   \else
227     \hyxmp@trimspaces\hyxmp@sublist
228     \@cons{#1}{\hyxmp@sublist}%
229     \def\next{\hyxmp@commas@to@list@i{#1}}%
230   \fi
231   \next
232 }

```

`\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.

```

233 \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.

```

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

```

238 \bgroup
239   \catcode'\^^U=11
240   \gdef\hyxmp@uscore{^^U}
241 \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.

```
242 \let\xmpquote=\relax
```

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

```
243 \bgroup
244 \catcode'\~ =12%
245 \gdef\xmptilde{~}%
246 \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.

```
247 \newcommand{\XMPTruncateList}[1]{%
248 \edef\hyxmp@temp@str{\csname hyxmp@#1\endcsname}%
249 \hyxmp@commas@to@list{\hyxmp@temp@list}{\hyxmp@temp@str}%
250 \def\@elt##1{%
251 \expandafter\gdef\csname @#1\endcsname{##1}%
252 \let\@elt=\@gobble
253 }
254 \hyxmp@temp@list
255 }}
```

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:20190225213624-07'00') [3], while XMP timestamps are of the form “YYYY-MM-DDThh:mm:ss+TT:tt” (e.g., 2019-02-25T21:36:24-07:00) [4]. The `\hyxmp@as@pdf@date` and `\hyxmp@as@xmp@date` macros defined in this section facilitate timestamp conversions to PDF and XMP formats, respectively.

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

```
\hyxmp@first@char@i 256 \def\hyxmp@first@char#1{\hyxmp@first@char@i#1\relax}
257 \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.

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

```

260     \hyxmp@pdf@to@xmp@date{#1}%
261   \else
262     #1%
263   \fi
264 }

```

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

```

265 \def\hyxmp@pdf@to@xmp@date#1:#2#3#4#5#6#7#8#9{%
266   #2#3#4#5-#6#7-#8#9%
267   \hyxmp@parse@time
268 }

```

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

```

269 \def\hyxmp@parse@time#1#2#3#4#5#6{%
270   T#1#2:#3#4:#5#6%
271   \hyxmp@parse@tz@char
272 }

```

`\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.

```

273 \def\hyxmp@parse@tz@char#1{%
274   #1%
275   \ifx#1-%
276     \expandafter\hyxmp@parse@tz
277   \else
278     \ifx#1+%
279       \expandafter\hyxmp@parse@tz
280     \fi
281   \fi
282 }

```

`\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.

```

283 \def\hyxmp@parse@tz#1'#2'{%
284   #1:#2%
285 }

```


`\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.

```

286 \def\hyxmp@as@pdf@date#1{%
287   \expandafter\ifx\hyxmp@first@char@i#1\relax D%
288     #1%
289   \else
290     \hyxmp@xmp@to@pdf@date{#1}%
291   \fi
292 }

```

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

```

293 \def\hyxmp@xmp@to@pdf@date#1{%
294   D:\hyxmp@xmp@to@pdf@date@i#1\relax\relax
295 }

```

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

```

296 \def\hyxmp@xmp@to@pdf@date@i#1#2#3#4#5#6{%
297   #1#2#3#4%
298   \ifx#5-%
299     \expandafter\hyxmp@xmp@to@pdf@date@ii\expandafter#6%
300   \fi
301 }

```

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

```

302 \def\hyxmp@xmp@to@pdf@date@ii#1#2#3#4{%
303   #1#2%
304   \ifx#3-%
305     \expandafter\hyxmp@xmp@to@pdf@date@iii\expandafter#4%
306   \fi
307 }

```

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

```

308 \def\hyxmp@xmp@to@pdf@date@iii#1#2#3#4{%
309   #1#2%
310   \ifx#3T%
311     \expandafter\hyxmp@xmp@to@pdf@date@iv\expandafter#4%
312   \fi
313 }

```

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

```

314 \def\hyxmp@xmp@to@pdf@date@iv#1#2#3#4{%
315   #1#2%
316   \ifx#3:%
317     \expandafter\hyxmp@xmp@to@pdf@date@v\expandafter#4%
318   \fi
319 }

```

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

```

320 \def\hyxmp@xmp@to@pdf@date@v#1#2#3#4{%
321   #1#2%
322   \ifx#3:%
323     \expandafter\hyxmp@xmp@to@pdf@date@vi\expandafter#4%
324   \fi
325 }
```

`\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.

```

326 \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).

```

327 \def\hyxmp@xmp@to@pdf@date@vi#1#2#3#4{%
328   #1#2%
329   \ifx#3+%
330     +\expandafter\hyxmp@xmp@to@pdf@date@vii
331   \fi
332   \ifx#3-%
333     -\expandafter\hyxmp@xmp@to@pdf@date@vii
334   \fi
335   \ifx#3Z%
336     Z%
337   \fi
338   \ifx#3\relax
339     \expandafter\@hyxmp@gobbletwo
340   \fi
341   \@gobbletwo #4%
342 }
```

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

```

343 \def\hyxmp@xmp@to@pdf@date@vii#1\@gobbletwo#2#3#4#5{%
344   #2#3%
345   \ifx#4:%
346     \expandafter\hyxmp@xmp@to@pdf@date@viii\expandafter#5%
347   \fi
348 }
```

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

```

349 \def\hyxmp@xmp@to@pdf@date@viii#1#2#3#4{%
350   '#1#2'%
351 }
```

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

```

352 \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.
353 \xdef#1{\the\year}%
354 \ifnum\month<10
355     \xdef#1{#1-0\the\month}%
356 \else
357     \xdef#1{#1-\the\month}%
358 \fi
359 \ifnum\day<10
360     \xdef#1{#1-0\the\day}%
361 \else
362     \xdef#1{#1-\the\day}%
363 \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.

```

364 \@tempcnta=\time
365 \divide\@tempcnta by 60%
366 \ifnum\@tempcnta<10%
367     \xdef#1{#1T0\the\@tempcnta}%
368 \else
369     \xdef#1{#1T\the\@tempcnta}%
370 \fi
371 \multiply\@tempcnta by -60%
372 \advance\@tempcnta by \time
373 \ifnum\@tempcnta<10%
374     \xdef#1{#1:0\the\@tempcnta}%
375 \else
376     \xdef#1{#1:\the\@tempcnta}%
377 \fi
378 }

```

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

```

379 \@ifundefined{pdffeedback}{%
380     \@ifundefined{pdfcreationdate}{%

```

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

```

381     \hyxmp@today@define\hyxmp@today
382 }{%

```

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

```

383     \edef\hyxmp@today{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
384 }%
385 }{%

```

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

```
386 \edef\hyxmp@today{\expandafter\hyxmp@pdf@to@xmp@date\pdffeedback creationdate}%
387 }
```

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.

```
388 \catcode'\Q=3
\hyxmp@trimspaces\x redefines \x to have the same replacement text sans leading
and trailing space tokens.
389 \newcommand{\hyxmp@trimspaces}[1]{%
  Use grouping to emulate a multi-token afterassignment queue.
390 \begingroup
  Put “\toks 0 {” into the afterassignment queue.
391 \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.
392 \expandafter\hyxmp@trimb\expandafter\noexpand#1Q Q}%
  Transfer the trimmed text back into #1.
393 \edef#1{\the\toks0}%
394 }
```

`\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.

```
395 \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.

```
396 \def\hyxmp@trimc#1Q#2{\afterassignment\endgroup \vfuzz\the\vfuzz#1}
397 \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 XeLaTeX 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.

398 \newif\ifhyxmp@unicodetex
399 \ifnum64='\^^^0040\relax
400   \hyxmp@unicodetexttrue
401 \else
402   \hyxmp@unicodetextfalse
403 \fi

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

404 \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.

405 \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.

406 \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;.

407 \newcommand*{\hyxmp@xmlify}[1]{%
408   \gdef\hyxmp@xmlified{%
Escaped PDF string → PDFDocEncoding/Unicode
409   \EdefUnescapeString\hyxmp@text{#1}%
410   \ifhyxmp@unicodetex
PDFDocEncoding/Unicode → UTF-32BE
411     \hyxmp@is@unicode\hyxmp@text{%
412       \StringEncodingConvert
413       \hyxmp@text\hyxmp@text{utf16be}{utf32be}%
414     }{%

```

```

415     \ifxetex
416       \hyxmp@xetex@crap
417     \else
418       \StringEncodingConvert
419       \hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
420     \fi
421   }%

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

UTF-32BE → XML in ASCII
423   \edef\hyxmp@text{%
424     \expandafter
425   }\expandafter\hyxmp@toxml@unicodetex\hyxmp@text
426   \relax\relax\relax\relax\relax\relax\relax\relax
427   \else

PDFDocEncoding/Unicode → UTF-8
428   \hyxmp@is@unicode\hyxmp@text{%
429     \StringEncodingConvert
430     \hyxmp@text\hyxmp@text{utf16be}{utf8}%
431   }{%
432     \StringEncodingConvert
433     \hyxmp@text\hyxmp@text{pdfdoc}{utf8}%
434   }%

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

UTF-8 as hex string → XML in UTF-8 as hex string
436   \edef\hyxmp@text{%
437     \expandafter\hyxmp@toxml\hyxmp@text\@empty\@empty
438   }%

XML in UTF-8 as hex string → XML in UTF-8
439   \EdefUnescapeHex\hyxmp@text{\hyxmp@text}%
440   \fi
441   \global\let\hyxmp@xmlified\hyxmp@text
442 }

```

`\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.

```

443 \begingroup
444   \lccode'\<=254 %
445   \lccode'\>=255 %
446   \catcode254=12 %
447   \catcode255=12 %
448 \lowercase{\endgroup
449   \def\hyxmp@is@unicode#1{%
450     \expandafter\hyxmp@@is@unicode#1<>\@nil
451   }%

```

```

452 \def\hyxmp@is@unicode#1<>#2\@nil{%
453   \ifx\#1\%
454     \expandafter\@firstoftwo
455   \else
456     \expandafter\@secondoftwo
457   \fi
458 }%
459 }

```

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

```

460 \def\hyxmp@toxml#1#2{%
461   \ifx#1\empty
462   \else
463     \ifnum"#1#2='\& %
464       26616D703B% &amp;
465     \else\ifnum"#1#2='\< %
466       266C743B% &lt;
467     \else\ifnum"#1#2='\> %
468       2667743B% &gt;
469     \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.

```

470   \@ifundefined{pdfmark}{%
471     #1#2%
472   }{%
473     \ifnum"#1#2='\( %
474       5C28% \(
475     \else\ifnum"#1#2='\) %
476       5C29% \)
477     \else
478       #1#2%
479     \fi\fi
480   }%
481   \fi\fi\fi
482   \expandafter\hyxmp@toxml
483 \fi
484 }

```

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

```

485 \def\hyxmp@toxml@unicodetex#1#2#3#4#5#6#7#8{%
486   \ifx#1\relax
487   \else
488     \ifnum"#1#2#3#4#5#6#7#8>127 %
489       \uccode'\*="#1#2#3#4#5#6#7#8\relax
490       \uppercase{%
491         \edef\hyxmp@text{\hyxmp@text *}%
492       }%
493     \else\ifnum"#7#8='< %
494       \edef\hyxmp@text{\hyxmp@text &lt;}%
495     \else\ifnum"#7#8='& %
496       \edef\hyxmp@text{\hyxmp@text &amp;%
497     \else\ifnum"#7#8='> %
498       \edef\hyxmp@text{\hyxmp@text &gt;}%
499     \else\ifnum"#7#8='\ %
500       \edef\hyxmp@text{\hyxmp@text\space}%
501     \else
502       \uccode'\*="#7#8\relax
503       \uppercase{%
504         \edef\hyxmp@text{\hyxmp@text *}%
505       }%
506     \fi\fi\fi\fi\fi
507     \expandafter\hyxmp@toxml@unicodetex
508   \fi
509 }
```

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

```

510 \def\hyxmp@skipzeros#1{%
511   \ifx#10%
512     \expandafter\hyxmp@skipzeros
513   \fi
514 }
```

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



```

527     \StringEncodingConvert\hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
528     \fi
529   }%
530 }
531 \x{ }

```

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

```

532 \begingroup
533   \catcode'\~=12 %
534   \lccode'\~=\' %
535 \lowercase{\endgroup
536   \def\hyxmp@SpaceOther#1 #2\@nil{%
537     #1%
538     \ifx\relax#2\relax
539       \expandafter\@gobble
540     \else
541       ~%
542       \expandafter\@firstofone
543     \fi
544     {\hyxmp@SpaceOther#2\@nil}%
545   }%
546 }

```

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

```

547 \def\hyxmp@crap@test#1{%
548   \ifx#1\relax
549   \else
550     \ifnum'#1>127 %
551       \let\hyxmp@crap@result=Y%
552       \expandafter\expandafter\expandafter\hyxmp@skiptorelax
553     \else
554       \expandafter\expandafter\expandafter\hyxmp@crap@test
555     \fi
556   \fi
557 }

```

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

```

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

```

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

```

\hyxmp@num 559 \def\hyxmp@crap@convert#1{%
\hyxmp@text 560   \ifx#1\relax
561   \else
562     \edef\hyxmp@num{\number'#1}%
563     \ifnum\hyxmp@num>"FFFFFF %
564       \lccode'\!=\intcalcDiv{\hyxmp@num}{\number"1000000}\relax
565       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
566     \edef\hyxmp@num{\intcalcMod{\hyxmp@num}{\number"1000000}}%
567   \else

```

```

568     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
569   \fi
570   \ifnum\hyxmp@num>"FFFF %
571     \lccode'\!=\intcalDiv{\hyxmp@num}{\number"10000}\relax
572     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
573     \edef\hyxmp@num{\intcalMod{\hyxmp@num}{\number"10000}}%
574   \else
575     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
576   \fi
577   \ifnum\hyxmp@num>"FF %
578     \lccode'\!=\intcalDiv{\hyxmp@num}{\number"100}\relax
579     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
580     \edef\hyxmp@num{\intcalMod{\hyxmp@num}{\number"100}}%
581   \else
582     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
583   \fi
584   \ifnum\hyxmp@num>0 %
585     \lccode'\!=\hyxmp@num\relax
586     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
587   \else
588     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
589   \fi
590   \expandafter\hyxmp@crap@convert
591 \fi
592 }

```

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

```

593 \begingroup
594   \catcode0=12 %
595   \gdef\hyxmp@zero{^^00}%
596 \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}`”.

```

597 \def\hyxmp@alt@title{}
598 \def\hyxmp@alt@description{}
599 \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.`

```

600 \newcommand{\hyxmp@LA@accept}[2]{%
601   \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`.

```

602   \hyxmp@pdfstringdef\hyxmp@value{##1}%
603   \xdef#2{#2\noexpand\do {\hyxmp@cur@lang} {\hyxmp@value}}%
604 }
605 }

```

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

```

606 \hyxmp@LA@accept{pdftitle}{\hyxmp@alt@title}
607 \hyxmp@LA@accept{pdfsubject}{\hyxmp@alt@description}
608 \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.

```

609 \newcommand{\XMPLangAlt}[2]{%
610   \let\do=\relax

```

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

```

611   \edef\hyxmp@cur@lang{#1}%
612   \setkeys{hyxmp@LA}{#2}%
613 }

```

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.

```

614 \def\hyxmp@modulo@a#1{%
615   \@tempcntb=\@tempcnta
616   \divide\@tempcntb by #1
617   \multiply\@tempcntb by #1
618   \advance\@tempcnta by -\@tempcntb
619 }

```

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

```

\hyxmp@big@prime@ii 620 \def\hyxmp@big@prime{536870923}
                     621 \def\hyxmp@big@prime@ii{536870027}

```

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

```

\hyxmp@one@token 622 \def\hyxmp@seed@rng#1{%
                  623   \@tempcnta=\hyxmp@big@prime
                  624   \futurelet\hyxmp@one@token\hyxmp@seed@rng@i#1\@empty
                  625 }

```

`\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 626 \def\hyxmp@seed@rng@i{%
                  627   \ifx\hyxmp@one@token\@empty
                  628     \let\next=\relax
                  629   \else
                  630     \def\next##1{%
                  631       \multiply\@tempcnta by 3
                  632       \advance\@tempcnta by '##1
                  633       \hyxmp@modulo@a{\hyxmp@big@prime}%
                  634       \futurelet\hyxmp@one@token\hyxmp@seed@rng@i
                  635     }%
                  636   \fi
                  637   \next
                  638 }

```

`\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 639 \def\hyxmp@set@rand@num{%
                  640   \@tempcnta=\hyxmp@rand@num
                  641   \multiply\@tempcnta by 3
                  642   \advance\@tempcnta by \hyxmp@big@prime@ii
                  643   \hyxmp@modulo@a{\hyxmp@big@prime}%
                  644   \xdef\hyxmp@rand@num{\the\@tempcnta}%
                  645 }

```

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

```

646 \def\hyxmp@append@hex#1{%
647   \hyxmp@set@rand@num
648   \@tempcnta=\hyxmp@rand@num
649   \hyxmp@modulo@a{16}%
650   \ifnum\@tempcnta<10
651     \xdef#1{#1\the\@tempcnta}%
652   \else

```

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

```

653     \advance\@tempcnta by -10
654     \ifcase\@tempcnta
655         \xdef#1{#1a}%
656         \or\xdef#1{#1b}%
657         \or\xdef#1{#1c}%
658         \or\xdef#1{#1d}%
659         \or\xdef#1{#1e}%
660         \or\xdef#1{#1f}%
661     \fi
662 \fi
663 }

```

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

```

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

```

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

```

669 \def\hyxmp@append@hex@iv#1{%
670     \hyxmp@append@hex@iii#1%
671     \hyxmp@append@hex#1%
672 }

```

`\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`.

```

673 \def\hyxmp@create@uuid#1{%
674     \def#1{uuid:}%
675     \hyxmp@append@hex@iv#1%
676     \hyxmp@append@hex@iv#1%
677     \g@addto@macro#1{-}%
678     \hyxmp@append@hex@iv#1%
679     \g@addto@macro#1{-4}%
680     \hyxmp@append@hex@iii#1%
681     \g@addto@macro#1{-}%

```

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

```

682     \hyxmp@set@rand@num
683     \@tempcnta=\hyxmp@rand@num
684     \hyxmp@modulo@a{4}%
685     \ifcase\@tempcnta
686         \g@addto@macro#1{8}%
687         \or\g@addto@macro#1{9}%
688         \or\g@addto@macro#1{a}%

```

```

689     \or\g@addto@macro#1{b}%
690     \fi
691     \hyxmp@append@hex@iii#1%
692     \g@addto@macro#1{-}%
693     \hyxmp@append@hex@iv#1%
694     \hyxmp@append@hex@iv#1%
695     \hyxmp@append@hex@iv#1%
696 }

\hyxmp@def@DocumentID Seed the random-number generator with a function of the current filename, PDF
    \hyxmp@DocumentID document title, and PDF author, then invoke \hyxmp@create@uuid to define
    \hyxmp@seed@string \hyxmp@DocumentID as a random UUID.
697 \newcommand*{\hyxmp@def@DocumentID}{%
698     \edef\hyxmp@seed@string{\jobname:\@pdftitle:\@pdfauthor:}%
699     \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
700     \edef\hyxmp@rand@num{\the\@tempcnta}%
701     \hyxmp@create@uuid\hyxmp@DocumentID
702 }

\hyxmp@def@InstanceID Seed the random-number generator with a function of the current filename,
    \hyxmp@InstanceID PDF document title, PDF author, and the current timestamp, then invoke
    \hyxmp@seed@string \hyxmp@create@uuid to define \hyxmp@InstanceID as a random UUID. For the
    current timestamp, we use both the document-specified timestamp from pdfdate
    and the TEX time. The former can be more precise (to sub-seconds) but may be
    less random (as it depends on manual document modifications) while the latter
    is typically less precise (to minutes) but may be more random (as it is updated
    automatically).
703 \newcommand*{\hyxmp@def@InstanceID}{%
704     \hyxmp@today@define{\hyxmp@seed@string}%
705     \edef\hyxmp@seed@string{%
706         \jobname:\@pdftitle:\@pdfauthor:\hyxmp@today:\hyxmp@seed@string
707     }%
708     \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
709     \edef\hyxmp@rand@num{\the\@tempcnta}%
710     \hyxmp@create@uuid\hyxmp@InstanceID
711 }

```

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.

```

712 \newcommand*{\hyxmp@add@to+xml}[1]{%
713   \bgroup
714   \@tempcnta=0
715   \ifhyxmp@unicodetex
716     \@tempcntb=65536%
717   \else
718     \@tempcntb=256%
719   \fi
720   \loop
721     \lccode\@tempcnta=\@tempcnta
722     \advance\@tempcnta by 1
723     \ifnum\@tempcnta<\@tempcntb
724       \repeat
725       \lccode'\_='\ \relax
726       \lccode'\^C='\,\relax
727       \lccode'\^U='\_\relax
728       \lowercase{\xdef\hyxmp@new+xml{#1}}%
729       \xdef\hyxmp+xml{\hyxmp+xml\hyxmp@new+xml}%
730   \egroup
731 }
```

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

```

732 \bgroup
733 \catcode'\#=11
734 \gdef\hyxmp@hash{#}
735 \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.

```

736 \bgroup
737 \xdef\hyxmp+xml{%
738   \hyxmp@add@to+xml{%
739     -----^^J%
740   }
741   \xdef\hyxmp@padding{\hyxmp+xml}%
742 \egroup
743 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
744 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
745 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
```

```

746 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
747 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}

```

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

```

748 \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.

```

749 \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.

```

750 \hyxmp@add@to@xml{%
751 -----<rdf:Description rdf:about=""^^J%
752 -----xmlns:pdf="http://ns.adobe.com/pdf/1.3/">^^J%
753 }%
754 \hyxmp@add@simple@var{pdf:Keywords}{@pdfkeywords}%
755 \hyxmp@add@simple@var{pdf:Producer}{@pdfproducer}%
756 \@ifundefined{pdfvariable}{%
757   \@ifundefined{pdfminorversion}{%

```

Case 1: Neither `\pdfvariable` nor `\pdfminorversion` is defined (X_YL^AT_EX and regular L^AT_EX).

```

758   }{%

```

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

```

759     \hyxmp@add@simple{pdf:PDFVersion}{1.\the\pdfminorversion}%
760   }%
761 }{%

```

Case 3: `\pdfvariable` is defined (LuaL^AT_EX 0.85+).

```

762   \hyxmp@add@simple{pdf:PDFVersion}{1.\the\pdfvariable minorversion}%
763 }%
764 \hyxmp@add@to@xml{%
765 -----</rdf:Description>^^J%
766 }%
767 }

```

`\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`

```

768 \newcommand*{\hyxmp@add@simple}[2]{%

```



```

769 \edef\hyxmp@string{#2}%
770 \ifx\hyxmp@string\empty
771 \else
772 \hyxmp@xmlify{\hyxmp@string}%
773 \hyxmp@add@to@xml{%
774 -----<#1>\hyxmp@xmlified</#1>^^J%
775 }%
776 \fi
777 }

```

\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.

```

778 \newcommand*{\hyxmp@add@simple@var}[2]{%
779 \expandafter\ifx\csname#2\endcsname\relax
780 \else
781 \hyxmp@xmlify{\csname#2\endcsname}%
782 \hyxmp@add@to@xml{%
783 -----<#1>\hyxmp@xmlified</#1>^^J%
784 }%
785 \fi
786 }

```

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.

```

787 \newcommand*{\hyxmp@rdf@dc}[3][\iffalse]{%
788 \ifmtargexp{#3}{\@tempswafalse}{\@tempswatruetrue}%
789 #1
790 \@tempswatruetrue
791 \fi

```

Append the corresponding XML only if **\@tempswatruetrue**.

```

792 \if@tempswa
793 \hyxmp@xmlify{#3}%

```

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

```

794 \let\hyxmp@value=\hyxmp@xmlified
795 \hyxmp@add@to@xml{%
796 -----<dc:#2>^^J%

```

```

797 -----<rdf:Alt>^^J%
798  }%
799   \ifx\@pdfmetalang\hyxmp@x@default
800   \else
801     \hyxmp@xmlify{\@pdfmetalang}%
802     \hyxmp@add@to@xml{%
803 -----<rdf:li xml:lang="\hyxmp@xmlified">\hyxmp@value</rdf:li>^^J%
804   }%
805   \fi
806   \hyxmp@add@to@xml{%
807 -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@value</rdf:li>^^J%
808   }%

```

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

```

809   \def\do##1##2{
810     \hyxmp@xmlify{##2}%
811     \hyxmp@add@to@xml{%
812 -----<rdf:li xml:lang="##1">\hyxmp@xmlified</rdf:li>^^J%
813   }%
814   }%
815   \csname hyxmp@alt@#2\endcsname

```

Complete this XMP element.

```

816   \hyxmp@add@to@xml{%
817 -----</rdf:Alt>^^J%
818 -----</dc:#2>^^J%
819   }%
820   \fi
821 }%

```

\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.

```

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

```

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

```

823   \@ifmtargexp{#4}{\@tempwafalse}{\@tempwattrue}%
824   #1
825   \@tempwattrue
826   \fi

```

Append the corresponding XML only if \@tempwattrue.

```

827   \if@tempwa
828     \hyxmp@add@to@xml{%
829 -----<dc:#2>^^J%
830 -----<rdf:#3>^^J%
831   }%
832   \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`.

```

833     \hyxmp@xmlify{#4}%
834     \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
835     \def\@elt##1{%
836         \hyxmp@add@to@xml{%
837     -----<rdf:li>##1</rdf:li>^^J%
838         }%
839     }%
840     \hyxmp@list
841     \egroup
842     \hyxmp@add@to@xml{%
843     -----</rdf:#3>^^J%
844     -----</dc:#2>^^J%
845     }%
846     \fi
847 }
```

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

```

848 \newcommand*{\hyxmp@dc@schema}{%
849     \hyxmp@add@to@xml{%
850     -----<rdf:Description rdf:about=""^^J%
851     -----_xmlns:dc="http://purl.org/dc/elements/1.1/">^^J%
852     -----<dc:format>application/pdf</dc:format>^^J%
853     }%
854     \hyxmp@rdf@dc[\ifHy@pdfa]{title}{\@pdftitle}%
855     \hyxmp@rdf@dc[\ifHy@pdfa]{description}{\@pdfsubject}%
856     \hyxmp@rdf@dc{rights}{\@pdfcopyright}%
857     \hyxmp@list@to@xml[\ifHy@pdfa]{creator}{Seq}{\hyxmp@pdfauthor}%
858     \hyxmp@list@to@xml{subject}{Bag}{\hyxmp@pdfkeywords}%
859     \hyxmp@list@to@xml{date}{Seq}{\hyxmp@today}%
860     \hyxmp@list@to@xml{language}{Bag}{\@pdflang}%
861     \hyxmp@list@to@xml{type}{Bag}{\@pdftype}%
862     \ifx\@pdfsource\@empty
863     \else
864         \hyxmp@add@simple{dc:source}{\@pdfsource}%
865     \fi
866     \hyxmp@add@to@xml{%
867     -----</rdf:Description>^^J%
868     }%
869 }
```

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.

```

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

\hyxmp@legal Set \hyxmp@rights to YES if either pdfcopyright or pdflicenseurl was specified.
871 \let\hyxmp@rights=\@empty
872 \ifx\@pdflicenseurl\@empty
873 \else
874 \def\hyxmp@rights{YES}%
875 \fi
876 \ifx\@pdfcopyright\@empty
877 \else
878 \def\hyxmp@rights{YES}%
879 \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.
880 \ifx\hyxmp@rights\@empty
881 \else

Header
882 \hyxmp@add@to@xml{%
883 -----<rdf:Description rdf:about=""^^J%
884 -----xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/">^^J%
885 }%

Copyright indication
886 \ifx\@pdfcopyright\@empty
887 \else
888 \hyxmp@add@to@xml{%
889 -----<xmpRights:Marked>True</xmpRights:Marked>^^J%
890 }%
891 \fi

License URL
892 \hyxmp@add@simple{xmpRights:WebStatement}{\@pdflicenseurl}%

Trailer
893 \hyxmp@add@to@xml{%
894 -----</rdf:Description>^^J%
895 }%
896 \fi
897 }
```

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.

```

898 \gdef\hyxmp@mm@schema{%
899   \ifmtargexp{\hyxmp@DocumentID}{\hyxmp@def@DocumentID}{}%
900   \ifmtargexp{\hyxmp@InstanceID}{\hyxmp@def@InstanceID}{}%
901   \hyxmp@add@to@xml{%
902     <rdf:Description rdf:about=""^^J%
903     -----xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/">^^J%
904   }%
905   \hyxmp@add@simple{xmpMM:DocumentID}{\hyxmp@DocumentID}%
906   \hyxmp@add@simple{xmpMM:InstanceID}{\hyxmp@InstanceID}%
907   \hyxmp@add@to@xml{%
908     </rdf:Description>^^J%
909   }%
910 }
```

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.

```

911 \newcommand*{\hyxmp@define@createdate}{%
912   \ifundefined{pdffeedback}{%
913     \ifundefined{pdfcreationdate}{%
```

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

```

914     \hyxmp@today@define\hyxmp@createdate
915   }{%
```

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

```

916     \edef\hyxmp@createdate{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
917   }%
918 }
```

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

```

919     \edef\hyxmp@createdate{\expandafter\hyxmp@pdf@to@xmp@date\pdffeedback creationdate}%
920   }%
921 }
```

`\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`.

```

922 \newcommand*{\hyxmp@xmp@basic@schema}{%
923   \hyxmp@add@to@xml{%
```

```

924 -----<rdf:Description rdf:about=""^^J%
925 -----xmlns:xmp="http://ns.adobe.com/xap/1.0/">^^J%
926  }%
927   \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.
928   \@ifundefined{@pdfcreationdate}{%
929     \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@createdate}%
930   }{%
931     \ifx\@pdfcreationdate\@empty
932       \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@createdate}%
933     \else
934       \hyxmp@add@simple{xmp:CreateDate}{%
935         \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfcreationdate}}%
936     \fi
937   }%
    For the document's modification date, use the user-specified \@pdfmoddate if
    defined and non-empty. Otherwise use our fabricated \hyxmp@createdate.
938   \@ifundefined{@pdfmoddate}{%
939     \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@createdate}%
940   }{%
941     \ifx\@pdfmoddate\@empty
942       \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@createdate}%
943     \else
944       \hyxmp@add@simple{xmp:ModifyDate}{%
945         \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfmoddate}}%
946     \fi
947   }%
    For the document's metadata date, use the user-specified \@pdfmetadatettime if
    defined and non-empty. Otherwise use our fabricated \hyxmp@createdate.
948   \ifx\@pdfmetadatettime\@empty
949     \hyxmp@add@simple{xmp:MetadataDate}{\hyxmp@createdate}%
950   \else
951     \hyxmp@add@simple{xmp:MetadataDate}{\@pdfmetadatettime}%
952   \fi
    Define the creation tool and the base URL.
953   \hyxmp@add@simple{xmp:CreatorTool}{\@pdfcreator}%
954   \hyxmp@add@simple{xmp:BaseURL}{\@baseurl}%
955   \hyxmp@add@to@xml{%
956 -----</rdf:Description>^^J%
957   }%
958 }

```

3.5.7 The Photoshop schema

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

properties.

```

959 \gdef\hyxmp@photoshop@schema{%
960   \edef\hyxmp@photoshop@data{\@pdfauthortitle\@pdfcaptionwriter}%
961   \ifx\hyxmp@photoshop@data\@empty
962     \else
963       \hyxmp@add@to@xml{%
964         -----<rdf:Description rdf:about=""^^J%
965         -----xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/">^^J%
966       }%
967     \fi
968     \hyxmp@add@simple{photoshop:AuthorsPosition}{\@pdfauthortitle}%
969     \hyxmp@add@simple{photoshop:CaptionWriter}{\@pdfcaptionwriter}%
970     \ifx\hyxmp@photoshop@data\@empty
971       \else
972         \hyxmp@add@to@xml{%
973           -----</rdf:Description>^^J%
974         }%
975       \fi
976 }

```

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 (\textasciitilde J) character but written as an XML character entity for consistency across operating systems.

```

977 \begingroup
978   \catcode'\&=12
979   \catcode'\#=12
980   \gdef\xmplinesep{&\#xA;}
981 \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.

```

982 \newcommand*{\hyxmp@list@to@lines}[2]{%
983   \ifnotmtargexp{#2}{%
984     \bgroup
985       \hyxmp@add@to@xml{%
986         -----<#1>%
987       }%

```

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

```

988   \def\@elt@first##1{%
989     \hyxmp@add@to@xml{##1}%
990     \let\@elt=\@elt@rest
991   }%

```

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

```

992     \def\@elt@rest##1{%
993         \hyxmp@add@to@xml{\xmplinesep##1}%
994     }%

```

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

```

995     \let\@elt=\@elt@first
996     \hyxmp@xmlify{#2}%
997     \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
998     \hyxmp@list
999     \hyxmp@add@to@xml{</#1>^^J}%
1000 \egroup
1001 }%
1002 }

```

\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.

```

1003 \gdef\hyxmp@photometa@schema{%
1004     \edef\hyxmp@photometa@data{%
1005         \@pdfcontactaddress
1006         \@pdfcontactcity
1007         \@pdfcontactregion
1008         \@pdfcontactpostcode
1009         \@pdfcontactcountry
1010         \@pdfcontactphone
1011         \@pdfcontactemail
1012         \@pdfcontacturl
1013     }%
1014     \ifx\hyxmp@photometa@data\@empty
1015     \else
1016         \hyxmp@iptc@extensions
1017         \hyxmp@add@to@xml{%
1018             <rdf:Description rdf:about="^^J%
1019             -----xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/">^^J%
1020             <Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">^^J%
1021             }%
1022         \fi
1023         \hyxmp@list@to@lines{Iptc4xmpCore:CiAdrExtadr}{\@pdfcontactaddress}%
1024         \hyxmp@add@simple{Iptc4xmpCore:CiAdrCity}{\@pdfcontactcity}%
1025         \hyxmp@add@simple{Iptc4xmpCore:CiAdrRegion}{\@pdfcontactregion}%
1026         \hyxmp@add@simple{Iptc4xmpCore:CiAdrPcode}{\@pdfcontactpostcode}%
1027         \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.

```

1028 \bgroup
1029 \def\xmplinesep{,}%
1030 \hyxmp@list@to@lines{Iptc4xmpCore:CTelWork}{\@pdfcontactphone}%
1031 \hyxmp@list@to@lines{Iptc4xmpCore:CIEmailWork}{\@pdfcontactemail}%
1032 \hyxmp@list@to@lines{Iptc4xmpCore:CiUrlWork}{\@pdfcontacturl}%
1033 \egroup
1034 \ifx\hyxmp@photometa@data\empty
1035 \else
1036 \hyxmp@add@to@xml{%
1037 -----</Iptc4xmpCore:CreatorContactInfo>^^J%
1038 -----</rdf:Description>^^J%
1039 }%
1040 \fi
1041 }

```

`\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.

```

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

```

```

1061 -----<pdfaProperty:category>external</pdfaProperty:category>^^J%
1062 -----<pdfaProperty:description>contact information for the document's creator</p
1063 -----</rdf:li>^^J%
1064 -----</rdf:Seq>^^J%
1065 -----</pdfaSchema:property>^^J%
1066 -----<pdfaSchema:valueType>^^J%
1067 -----<rdf:Seq>^^J%
1068 -----<rdf:li rdf:parseType="Resource">^^J%
1069 -----<pdfaType:type>contactinfo</pdfaType:type>^^J%
1070 -----<pdfaType:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaTyp
1071 -----<pdfaType:prefix>Iptc4xmpCore</pdfaType:prefix>^^J%
1072 -----<pdfaType:description>contact information</pdfaType:description>^^J%
1073 -----<pdfaType:field>^^J%
1074 -----<rdf:Seq>^^J%
1075 }%

1076 \hyxmp@text@resource{CiAdrExtadr}{contact address}%
1077 \hyxmp@text@resource{CiAdrCity}{contact city}%
1078 \hyxmp@text@resource{CiAdrRegion}{contact region}%
1079 \hyxmp@text@resource{CiAdrPcode}{contact postal code}%
1080 \hyxmp@text@resource{CiAdrCtry}{contact country}%
1081 \hyxmp@text@resource{CiTelWork}{contact telephone number}%
1082 \hyxmp@text@resource{CiEmailWork}{contact email address}%
1083 \hyxmp@text@resource{CiUrlWork}{contact url}%

1084 \hyxmp@add@to+xml{%
1085 -----</rdf:Seq>^^J%
1086 -----</pdfaType:field>^^J%
1087 -----</rdf:li>^^J%
1088 -----</rdf:Seq>^^J%
1089 -----</pdfaSchema:valueType>^^J%
1090 -----</rdf:li>^^J%
1091 -----</rdf:Bag>^^J%
1092 -----</pdfaExtension:schemas>^^J%
1093 -----</rdf:Description>^^J%
1094 }%
1095 }

```

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

```

1096 \newcommand*{\hyxmp@text@resource}[2]{%
1097   \hyxmp@add@to+xml{%
1098 -----<rdf:li rdf:parseType="Resource">^^J%
1099 -----<pdfaField:name>#1</pdfaField:name>^^J%
1100 -----<pdfaField:valueType>Text</pdfaField:valueType>^^J%
1101 -----<pdfaField:description>#2</pdfaField:description>^^J%
1102 -----</rdf:li>^^J%
1103   }
1104 }

```

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`.

```

1105 \newcommand*{\hyxmp@pdfa@id@schema}{%
1106   \ifHy@pdfa
1107     \hyxmp@add@to@xml{%
1108       <rdf:Description rdf:about=""^^J%
1109       -----xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/">^^J%
1110     }%
1111     \hyxmp@add@simple{pdfaid:part}{\pdfapart}%
1112     \hyxmp@add@simple{pdfaid:conformance}{\pdfaconformance}%
1113     \hyxmp@add@to@xml{%
1114       -----</rdf:Description>^^J%
1115     }%
1116   \fi
1117 }
```

3.5.10 Combining schemata into an XMP packet

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

```

1118 \begingroup
1119   \ifhyxmp@unicodetex
1120     \lccode'\!="FEFF %
1121     \lowercase{%
1122       \gdef\hyxmp@bom{!}
1123     }%
1124   \else
1125     \catcode'\^^ef=12
1126     \catcode'\^^bb=12
1127     \catcode'\^^bf=12
1128     \gdef\hyxmp@bom{^^ef^^bb^^bf}%
1129   \fi
1130 \endgroup
```

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

```

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

```

1141 \hyxmp@dc@schema
1142 \hyxmp@photoshop@schema
1143 \hyxmp@photometa@schema
1144 \hyxmp@xmp@basic@schema
1145 \hyxmp@pdfa@id@schema
1146 \hyxmp@mm@schema
1147 \hyxmp@add@to@xml{%
1148 ___</rdf:RDF>^^J%
1149 </x:xmpmeta>^^J%
1150 \hyxmp@padding
1151 <?xpacket end="w"?>^^J%
1152 }%
1153 }

```

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`

```

1154 \newcommand*{\hyxmp@embed@packet}{%
1155 \hyxmp@construct@packet
1156 \def\hyxmp@driver{hpdfTEX}%
1157 \ifx\hyxmp@driver\Hy@driver
1158 \hyxmp@embed@packet@pdfTEX
1159 \else
1160 \def\hyxmp@driver{hLUAteX}%
1161 \ifx\hyxmp@driver\Hy@driver
1162 \hyxmp@embed@packet@luaTeX
1163 \else
1164 \def\hyxmp@driver{hdvipdfm}%
1165 \ifx\hyxmp@driver\Hy@driver
1166 \hyxmp@embed@packet@dviPDFm
1167 \else
1168 \def\hyxmp@driver{hXeTeX}%
1169 \ifx\hyxmp@driver\Hy@driver
1170 \hyxmp@embed@packet@XeTeX
1171 \else
1172 \@ifundefined{pdfmark}{%
1173 \PackageWarningNoLine{hyperxmp}{%
1174 Unrecognized hyperref driver ‘\Hy@driver’.\MessageBreak
1175 \jobname.tex’s XMP metadata will *not* be\MessageBreak
1176 embedded in the resulting file}%
1177 }{%
1178 \hyxmp@embed@packet@pdfmark
1179 }%
1180 \fi

```

```

1181     \fi
1182   \fi
1183 \fi
1184 }

```

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.

```

1185 \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.

```

1186 \newcommand*{\hyxmp@embed@packet@pdftex}{%
1187   \bgroup
1188   \ifluatex
1189   \else
1190     \pdfcompresslevel=0
1191   \fi
1192   \immediate\pdfobj \ifluatex uncompressed\fi stream attr {%
1193     /Type /Metadata
1194     /Subtype /XML
1195   }{\hyxmp@xml}%
1196   \pdfcatalog {/Metadata \the\pdflastobj\space 0 R}%
1197 \egroup
1198 }

```

3.6.2 Embedding using LuaTeX 0.85+

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

```

1199 \newcommand*{\hyxmp@embed@packet@luatex}{%
1200   \immediate\pdfextension obj uncompressed stream attr {%
1201     /Type /Metadata
1202     /Subtype /XML
1203   }{\hyxmp@xml}%
1204   \pdfextension catalog {/Metadata \the\numexpr\pdffeedback lastobj\relax\space 0 R}%

```

1205 }

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.

```
1206 \newcommand*{\hyxmp@embed@packet@pdfmark}{%
1207   \pdfmark{%
1208     pdfmark=/NamespacePush
1209   }%
1210   \pdfmark{%
1211     pdfmark=/OBJ,
1212     Raw={/_objdef \string{hyxmp@Metadata\string} /type /stream}%
1213   }%
1214   \pdfmark{%
1215     pdfmark=/PUT,
1216     Raw={\string{hyxmp@Metadata\string}
1217       2 dict begin
1218         /Type /Metadata def
1219         /Subtype /XML def
1220         currentdict
1221       end
1222     }%
1223   }%
1224   \pdfmark{%
1225     pdfmark=/PUT,
1226     Raw={\string{hyxmp@Metadata\string} (\hyxmp+xml)}%
1227   }%
1228   \pdfmark{%
1229     pdfmark=/Metadata,
1230     Raw={\string{Catalog\string} \string{hyxmp@Metadata\string}}%
1231   }%
1232   \pdfmark{%
1233     pdfmark=/NamespacePop
1234   }%
1235 }
```

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.

```
1236 \newcommand*{\hyxmp@embed@packet@dvipdfm}{%
1237   \hyxmp@string@len{\hyxmp+xml}%
1238   \special{pdf: object @hyxmp@Metadata
1239     <<
1240       /Type /Metadata
1241       /Subtype /XML
```

```

1242         /Length \the\@tempcnta
1243     >>
1244     stream^^J\hyxmp@xml endstream%
1245 }%
1246 \special{pdf: docview
1247     <<
1248         /Metadata @hyxmp@Metadata
1249     >>
1250 }%
1251 }

```

`\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.

```

1252 \newcommand*{\hyxmp@string@len}[1]{%
1253     \@tempcnta=0
1254     \expandafter\hyxmp@count@spaces#1 {} %
1255     \expandafter\hyxmp@count@non@spaces#1{}%
1256 }

```

`\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.

```

1257 \def\hyxmp@count@spaces#1 {%
1258     \def\hyxmp@one@token{#1}%
1259     \ifx\hyxmp@one@token\empty
1260         \advance\@tempcnta by -1
1261     \else
1262         \advance\@tempcnta by 1
1263     \expandafter\hyxmp@count@spaces
1264     \fi
1265 }

```

`\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.

```

1266 \newcommand*{\hyxmp@count@non@spaces}[1]{%
1267     \def\hyxmp@one@token{#1}%
1268     \ifx\hyxmp@one@token\empty
1269     \else
1270         \advance\@tempcnta by 1
1271     \expandafter\hyxmp@count@non@spaces
1272     \fi
1273 }

```

3.6.5 Embedding using X_YTeX

`\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.

```
1274 \newcommand*{\hyxmp@embed@packet@xetex}{%
1275   \special{pdf:stream @hyxmp@Metadata (\hyxmp@xml)
1276     <<
1277       /Type /Metadata
1278       /Subtype /XML
1279     >>
1280   }%
1281   \special{pdf:put @catalog
1282     <<
1283       /Metadata @hyxmp@Metadata
1284     >>
1285   }%
1286 }
```

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.

```
1287 \catcode'\="=\hyxmp@dq@code
```

4 Future Work

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

A Sample XMP packet

The following is an example of a complete XMP packet as may be produced by `hyperxmp`. This packet corresponds to the metadata included in the sample L^AT_EX 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>
  </rdf:Description>

```

```

</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:namespaceURI>
<pdfaSchema:prefix>Iptc4xmpCore</pdfaSchema:prefix>
<pdfaSchema:property>
  <rdf:Seq>
    <rdf:li rdf:parseType="Resource">
      <pdfaProperty:name>CreatorContactInfo</pdfaProperty:name>
      <pdfaProperty:valueType>contactinfo</pdfaProperty:valueType>
      <pdfaProperty:category>external</pdfaProperty:category>
      <pdfaProperty:description>contact information for the document's creator</pdfaProperty:description>
    </rdf:li>
  </rdf:Seq>
</pdfaSchema:property>
<pdfaSchema:valueType>
  <rdf:Seq>
    <rdf:li rdf:parseType="Resource">
      <pdfaType:type>contactinfo</pdfaType:type>
      <pdfaType:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaType:namespaceURI>
      <pdfaType:prefix>Iptc4xmpCore</pdfaType:prefix>
      <pdfaType:description>contact information</pdfaType:description>
      <pdfaType:field>
        <rdf:Seq>
          <rdf:li rdf:parseType="Resource">
            <pdfaField:name>CiAdrExtadr</pdfaField:name>
            <pdfaField:valueType>Text</pdfaField:valueType>
            <pdfaField:description>contact address</pdfaField:description>
          </rdf:li>
          <rdf:li rdf:parseType="Resource">
            <pdfaField:name>CiAdrCity</pdfaField:name>
            <pdfaField:valueType>Text</pdfaField:valueType>
            <pdfaField:description>contact city</pdfaField:description>
          </rdf:li>
          <rdf:li rdf:parseType="Resource">
            <pdfaField:name>CiAdrRegion</pdfaField:name>
            <pdfaField:valueType>Text</pdfaField:valueType>
            <pdfaField:description>contact region</pdfaField:description>
          </rdf:li>
          <rdf:li rdf:parseType="Resource">
            <pdfaField:name>CiAdrPcode</pdfaField:name>
            <pdfaField:valueType>Text</pdfaField:valueType>
            <pdfaField:description>contact postal code</pdfaField:description>
          </rdf:li>
          <rdf:li rdf:parseType="Resource">
            <pdfaField:name>CiAdrCtry</pdfaField:name>
            <pdfaField:valueType>Text</pdfaField:valueType>
            <pdfaField:description>contact country</pdfaField:description>
          </rdf:li>
        </rdf:Seq>
      </pdfaType:field>
    </rdf:li>
  </rdf:Seq>
</pdfaSchema:valueType>

```

```

        </rdf:li>
        <rdf:li rdf:parseType="Resource">
          <pdfaField:name>CiTelWork</pdfaField:name>
          <pdfaField:valueType>Text</pdfaField:valueType>
          <pdfaField:description>contact telephone number</pdfaField:de
        </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>2019-02-25T21:36:24-07:00</xmp:CreateDate>
  <xmp:ModifyDate>2019-02-25T21:36:24-07:00</xmp:ModifyDate>
  <xmp:MetadataDate>2019-02-25T21:36:24-07:00</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:6d1ac9ec-4ff2-515a-954b-648eeb4853b0
    </xmpMM:DocumentID>
    <xmpMM:InstanceID>
      uuid:059b675f-2bb6-4262-80a3-9b6c9b6759b6
    </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 schema. This change addresses a bug reported by Martin Münch 29
General: Initial version 1		
v1.1			
\hyxmp@construct@packet:			
Explicitly set the category codes of characters $\langle EF \rangle$, $\langle BB \rangle$, and $\langle BF \rangle$ to “letter”. Thanks to Daniel Schömer for the bug report 51			General: Introduced the <code>pdfmetalang</code> package option, which enables an author to specify the language in which he wrote the document’s metadata 20
v1.2			
General: Added support for the X _Y L _A T _E X backend (<code>xdvipdfmx</code>) . . 1		v1.4	
Added support for the Photoshop schema 1		\hyxmp@mm@schema: Renamed the <code>xapMM</code> namespace prefix to <code>xmpMM</code> 45	
Made the package compatible with <code>ngerman</code> . Thanks to Tobias Mueller for the bug report. 13		\hyxmp@rdf@dc: Included metadata in the <code>x-default</code> language regardless of the specified metadata language 41	
v1.3			
\hyxmp@reencode: Introduced this macro to re-encode Unicode		\hyxmp@xmpRights@schema: Renamed the <code>xapRights</code> namespace prefix to <code>xmpRights</code> 44	

v1.5	General: Made the XMP inclusion more robust. Thanks to Heiko Oberdiek for the bug report and suggested modifications. .	13	from line-wrapping the XMP packet	31
v2.0	\ProcessKeyvalOptions: Added this macro	19	\hyxmp@toxml@unicodetex: Added by Heiko Oberdiek	31
	\XMPTruncateList: Added this macro	23	\hyxmp@xetex@crap: Added by Heiko Oberdiek	32
	\hyxmp@ProcessKeyvalOptions: Added this macro	19	\hyxmp@xmlify: Completely rewritten by Heiko Oberdiek to better support Unicode-enabled T _E X programs	29
	\hyxmp@SpaceOther: Added by Heiko Oberdiek	33	\hyxmp@xmp@basic@schema: Added this macro	45
	\hyxmp@add@to@xml: Updated also to replace commas	39	\hyxmp@xmpRights@schema: Modified to include xmpRights:Marked only when pdfcopyright is specified and xmpRights:WebStatement only when pdflicenseurl is specified .	44
	\hyxmp@bom: Added by Heiko Oberdiek	51	\hyxmp@zero: Added by Heiko Oberdiek	34
	\hyxmp@comma: Added this macro	22	\ifhyxmp@unicodetex: Added by Heiko Oberdiek	29
	\hyxmp@construct@packet: Modified by Heiko Oberdiek to use an appropriate BOM representation via \hyxmp@bom	51	\xmpcomma: Added this macro	22
	\hyxmp@crap@convert: Added by Heiko Oberdiek	33	\xmpquote: Added this macro	23
	\hyxmp@crap@test: Added by Heiko Oberdiek	33	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	43	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	30	New \AtBeginDocument code from Heiko Oberdiek to properly encode \@pdfmetalang	20
	\hyxmp@list@to@xml: Modified by Heiko Oberdiek to use the new Unicode-processing macros	42	v2.1	
	\hyxmp@photoshop@schema: Simplified using \hyxmp@add@simple	46	\hypersetup: Added this macro	19
	\hyxmp@reencode: Replaced with an empty macro by Heiko Oberdiek	29	\hyxmp@hypersetup: Added this macro	19
	\hyxmp@skiptorelax: Added by Heiko Oberdiek	33	\hyxmp@redefine@Hyp: Added this macro	17
	\hyxmp@skipzeros: Added by Heiko Oberdiek	32	General: Enabled hyperxmp and hyperref to be loaded in either order. This addresses a bug report by Yuri Donskoy	17
	\hyxmp@string: Added this macro	40	v2.2	
	\hyxmp@toxml: Added by Heiko Oberdiek	31	\hyxmp@iptc@extensions: Added this macro to support PDF/A	
	Escaped parentheses written with pdfmarks to prevent dvips			

generation	49	\hyxmp@pdf@schema: Made	
\hyxmp@list@to@lines: Added		\hyxmp@pdf@schema <i>always</i>	
this macro	47	include the Adobe PDF schema,	
\hyxmp@photometa@schema: Added		even when empty. Florian	
this macro	48	Breitwieser noted that this is	
\hyxmp@text@resource: Added		necessary for PDF/A-1b	
this macro	50	compliance	40
\xmpcomma: Changed the default		\hyxmp@pdf@to@xmp@date: Added	
from \relax to an ordinary		this macro	24
comma	22	\hyxmp@pdfa@id@schema: Added	
\xmplinesep: Added this macro	47	this macro	51
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	27
the		\hyxmp@today@define: Added this	
lptc4xmpCore:CreatorContactInfo		macro	26
fields a unique pdfaType:prefix		\hyxmp@xmp@to@pdf@date: Added	
to better support conversion of		this macro	25
the document to PDF/A	49	\xmptilde: Added this macro	23
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	20	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	23	text	39
v2.4		\hyxmp@textunderscore: Added	
\hyxmp@add@simple@var: Added		this macro	14
this macro	41	\hyxmp@uscore: Added this macro	22
\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	37	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	43	General: Automatically use \title	
\hyxmp@parse@time: Added this		and \author if pdftitle and	
macro	24	pdfauthor are left unspecified.	
\hyxmp@parse@tz: Added this		Thanks to Maciej Radziejewski	
macro	24	for the suggestion	21
\hyxmp@parse@tz@char: Added		v2.8	
this macro	24	\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)	39	packet—uncompressed in both pdf _T E _X and pre-0.85 Lua _T E _X	53
v2.9		v3.2	
\hyxmp@pdfa@id@schema: Let the author specify the PDF/A part and conformance IDs, as requested by Leonid Sinev	51	\hyxmp@as@pdf@date: Added this macro	25
\hyxmp@photometa@schema: Use lptc4xmpCore instead of lptc4ContInfo as the contact-information metadata prefix. Leonid Sinev reports that Acrobat's PDF/A validator seems to prefer lptc4xmpCore	48	\hyxmp@as@xmp@date: Added this macro	23
General: Force inclusion of dc:creator, dc:title, and dc:description—even if empty—when hyperref is loaded with the pdfa option (suggested by Leonid Sinev)	1	\hyxmp@today@define: Modified to include hours and minutes	26
Introduced the pdf _T ype package option, which enables an author to specify the type of document being produced	1	\hyxmp@xmp@basic@schema: Honor hyperref's pdfcreationdate and pdfmoddate options plus a new pdfmetadate option. Leonid Sinev requested this additional control and helped test the resulting hyperxmp code	45
v3.0		v3.3	
\hyxmp@embed@packet@luatex: Added this macro	53	\@pdfsource: Added this macro and the corresponding pdfsource option, at Niklas Beisert's request	16
\hyxmp@today@define: Modified to accept the name of a macro to define	26	\XMPLangAlt: Added this macro based on a request—and some code—by Niklas Beisert to support metadata expressed in multiple languages	35
\hyxmp@xmp@basic@schema: Made the XMP CreateDate, ModifyDate, and MetadataDate match the PDF CreationDate	45	\hyxmp@rdf@dc: Bug fix: Output the metadata language as correct XML even when hyperref is loaded with the unicode option	41
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 hyperxmp code	1	General: Don't overwrite an existing pdfmetalang with pdflang or x-default. This addresses a bug report by Niklas Beisert	20
v3.1		v3.4	
\hyxmp@embed@packet@luatex: Updated to use \pdfextension obj uncompressed as suggested by Hans Hagen	53	\hyxmp@seed@string: Correctly handle an author field of all spaces. Bug reported by Gaëtan Leurent	38
\hyxmp@embed@packet@pdf _T ex: Leave the XMP packet—and only the XMP		General: Use ifmtarg to test for empty arguments, including non-empty but all spaces	1
		v3.5	
		\hyxmp@DocumentID: Added the pdfdocumentid option, at Michael Osipov's request	16
		\hyxmp@InstanceID: Added the pdfinstanceid option, at Michael	

Osipov's request	16	pdfdocumentid and pdfinstanceid options	45
\hyxmp@mm@schema: Generate \hyxmp@DocumentID and \hyxmp@InstanceID only if the document does not already define these using the		\hyxmp@seed@string: Seed with the T _E X timestamp in addition to the document-specified timestamp	38

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in *roman* refer to the code lines where the entry is used.

Symbols			
\#	733, 979	\@pdfcontactcity	76, 149, 1006, 1024
\&	463, 495, 978	\@pdfcontactcountry	82, 150, 1009, 1027
\,	726	\@pdfcontactemail	86, 151, 1011, 1031
\@author	200, 201	\@pdfcontactphone	84, 152, 1010, 1030
\@baseurl	144, 954	\@pdfcontactpostcode	80, 153, 1008, 1026
\@elt	247, 833, 990, 995	\@pdfcontactregion	78, 154, 1007, 1025
\@elt@first	988	\@pdfcontacturl	88, 155, 1012, 1032
\@elt@rest	990, 992	\@pdfcopyright	46, 156, 856, 876, 886
\@firstofone	542	\@pdfcreationdate	157, 931, 935
\@firstoftwo	454	\@pdfcreator	953
\@gobble	252, 539	\@pdfdatetime	24, 158, 189, 191
\@gobbletwo 326, 341, 343		\@pdfkeywords	120, 159
\@hyxmp@gobbletwo	326, 339	\@pdflang	160, 178, 179, 182, 185, 860
\@ifmtarg	16	\@pdflicenseurl	50, 161, 872, 892
\@ifmtargexp	16, 193, 199, 788, 823, 899, 900	\@pdfmetadatetitle	35, 162, 948, 951
\@ifnotmtarg	17	\@pdfmetalang	56, 181, 183, 185, 188, 799, 801
\@ifnotmtargexp	16, 194, 200, 983	\@pdfmoddate 163, 941, 945	
\@pdfaformance	60, 1112	\@pdfsource 62, 862, 864	
\@pdfapart	58, 1111	\@pdfsubject	164, 855
\@pdfauthor	99, 145, 199, 698, 706		
\@pdfauthortitle	52, 146, 960, 968		
\@pdfcaptionwriter	54, 147, 960, 969		
\@pdfcontactaddress	68, 148, 1005, 1023		
		\@pdftitle	165, 193, 698, 706, 854
		\@pdftype	48, 166, 861
		\@secondoftwo	456
		\@tempswafalse 788, 823	
		\@tempswattrue	788, 790, 823, 825
		\@title	194, 195
		\^	235, 239, 399, 726, 727, 1125–1127
		_	725, 727
		\~	244, 533, 534
		_	499, 534, 725
		A	
		\and	99
		ASCII	14, 30
		\AtBeginDocument	176
		\AtEndDocument	5
		\AtEndDvi	8
		atenddvi	13
		Author	8, 23
		B	
		Bag	64
		baseurl (option)	3, 11, 13, 45
		BOM	51, 63
		C	
		CiAdrCity	2, 48
		CiAdrCtry	2, 48
		CiAdrExtadr	2, 48

CiAdrPcode	2, 48	hyperref	1, 3–6, 9, 11, 13, 14, 17, 19– 21, 40, 52–54, 63–65	\hyxmp@big@prime 620 , 623, 633, 643
CiAdrRegion	2, 48	\hypersetup 137 , 195, 201		\hyxmp@big@prime@ii 620 , 642
CiEmailWork	2, 48	hyperxmp	1–6, 9–17, 20–23, 29, 36, 40, 53, 56, 63, 65	\hyxmp@bom . . . 1118 , 1133
CiTelWork	2, 48	\hyxmp@@is@unicode . 443		\hyxmp@comma 70, 100, 121, 234
CiUrlWork	2, 48	\hyxmp@add@simple 759 , 762, 768 , 864, 892, 905, 906, 929, 932, 934, 939, 942, 944, 949, 951, 953, 954, 968, 969, 1024– 1027, 1111, 1112		\hyxmp@commas@to@list . 218 , 249, 834, 997
CreateDate	45, 65	\hyxmp@add@simple@var 754 , 755, 778		\hyxmp@commas@to@list@i 220 , 222
CreationDate	65	\hyxmp@add@to@xml 712 , 738, 750, 764, 773, 782, 795, 802, 806, 811, 816, 828, 836, 842, 849, 866, 882, 888, 893, 901, 907, 923, 955, 963, 972, 985, 989, 993, 999, 1017, 1036, 1043, 1084, 1097, 1107, 1113, 1133, 1147		\hyxmp@concat@metadata 142
D		\hyxmp@alt@description 597 , 607		\hyxmp@construct@packet 1131 , 1155
Date	27	\hyxmp@alt@rights 597 , 608		\hyxmp@count@non@spaces 1255 , 1266
\day	359, 360, 362	\hyxmp@alt@title 597 , 606		\hyxmp@count@spaces 1254 , 1257
dc:creator	2, 9, 43, 65	\hyxmp@and 99		\hyxmp@crap@convert 525 , 559
dc:date	2, 43	\hyxmp@append@hex . . . 646 , 665–667, 671		\hyxmp@crap@result 515 , 551
dc:description	3, 34, 43, 65	\hyxmp@append@hex@iii . 664 , 670, 680, 691		\hyxmp@crap@test 522 , 547
dc:format	2	\hyxmp@append@hex@iv 669 , 675, 676, 678, 693–695		\hyxmp@create@uuid 673 , 701, 710
dc:language	2, 43, 63, 64	\hyxmp@as@pdf@date . 286		\hyxmp@createdate 911 , 929, 932, 939, 942, 949
dc:rights	2, 13, 34, 43	\hyxmp@as@xmp@date . 30, 41, 258 , 935, 945		\hyxmp@cur@lang 603 , 611
dc:source	2, 43, 63	\hyxmp@at@end . . . 3 , 205		\hyxmp@dc@schema 848 , 1141
dc:subject	2, 43			\hyxmp@def@DocumentID 697 , 899
dc:title	3, 34, 43, 65			\hyxmp@def@InstanceID 703 , 900
dc:type	2			\hyxmp@define@createdate 911 , 927
\define@key				\hyxmp@DocumentID . . . 64 , 697 , 899, 905
.	25, 36, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 77, 79, 81, 83, 85, 87, 89, 99, 120, 601			\hyxmp@dq@code . 1 , 1287
\do	603, 610, 809			\hyxmp@driver . . . 3 , 1154
dvipdf (option)	54			\hyxmp@embed@packet 207 , 1154
dvipdfm	54			\hyxmp@embed@packet@dvipdfm 1166 , 1236
dvips (option)	54			\hyxmp@embed@packet@luatex 1162 , 1199
dvips	7, 31, 63			\hyxmp@embed@packet@pdfmark 1178 , 1206
dvipsone (option)	54			
dviwindo (option)	54			
E				
\EdefEscapeHex	422, 435			
\EdefUnescapeHex	439			
\EdefUnescapeString	409			
ETX	22, 29			
G				
Ghostscript	7, 8			
H				
\Hy@driver				
.	4, 1157, 1161, 1165, 1169, 1174			
\Hy@unicodefalse	27, 38			

\hyxmp@embed@packet@pdftex	\hyxmp@pdf@to@xmp@date	\hyxmp@sublist
..... 1158, <u>1186</u> 260, <u>265</u> ,	. 223, 224, 227, 228	
\hyxmp@embed@packet@xetex	383, 386, 916, 919	\hyxmp@temp@list	... <u>247</u>
..... 1170, <u>1274</u>	\hyxmp@pdfa@id@schema	\hyxmp@temp@str	... <u>247</u>
\hyxmp@find@metadata <u>1105</u> , 1145	\hyxmp@text
..... <u>142</u> , 206	\hyxmp@pdfauthor	. 407, <u>485</u> , <u>515</u> , <u>559</u>	
\hyxmp@first@char 90, 99, 857	\hyxmp@text@resource	.. 1076–1083, <u>1096</u>
\hyxmp@first@char@i	\hyxmp@pdfkeywords	\hyxmp@textunderscore <u>18</u>
.... 256, 259, 287 <u>90</u> , <u>120</u> , 858	\hyxmp@today
\hyxmp@gobbletwo	\hyxmp@pdfstringdef	. 191, <u>379</u> , 706, 859	
\hyxmp@hash <u>18</u> , 29, 40,	\hyxmp@today@define	. 352, 381, 704, 914
1046–1049, 1137	47, 49, 51, 53, 55,	\hyxmp@toxml	.. 437, <u>460</u>
\hyxmp@Hyp@pdfauthor	57, 59, 61, 63, 65,	\hyxmp@toxml@unicodetex 425, <u>485</u>
\hyxmp@Hyp@pdfkeywords	67, 72, 77, 79, 81,	\hyxmp@trimb	.. 392, <u>395</u>
..... <u>114</u>	83, 85, 87, 89, 602	\hyxmp@trimc	.. 395, <u>396</u>
\hyxmp@hypersetup	\hyxmp@photometa@data	\hyxmp@trimspaces	..
\hyxmp@InstanceID <u>1003</u> 227, 388	
. 66, <u>703</u> , 900, 906	\hyxmp@photometa@schema	\hyxmp@try <u>515</u>
\hyxmp@iptc@extensions <u>1003</u> , 1143	\hyxmp@unicodetexfalse <u>398</u>
..... 1016, <u>1042</u>	\hyxmp@photoshop@data	\hyxmp@unicodetextrue <u>398</u>
\hyxmp@is@unicode <u>959</u>	\hyxmp@uscore	.. 20, <u>238</u>
.... 411, 428, <u>443</u>	\hyxmp@photoshop@schema	\hyxmp@value	.. <u>602</u> , <u>794</u>
\hyxmp@LA@accept <u>959</u> , 1142	\hyxmp@x@default 183, <u>748</u> , 799, 807
.... 600, 606–608	\hyxmp@ProcessKeyvalOptions	\hyxmp@xetex@crap	..
\hyxmp@legal <u>132</u> 416, <u>515</u>	
\hyxmp@list	\hyxmp@rand@num	\hyxmp+xml
. 834, 840, 997, 998	639, 648, 683, 700, 709	729, <u>736</u> , <u>1131</u> ,	
\hyxmp@list@to@lines	\hyxmp@rdf@dc	1195, 1203, 1226,	
..... <u>982</u> , 787, 854–856	1237, 1244, 1275	
1023, 1030–1032	\hyxmp@redefine@Hyp	\hyxmp@xmlified	<u>407</u> ,
\hyxmp@list@to@xml <u>92</u> , 134, 139	774, 783, 794,	
.... <u>822</u> , 857–861	\hyxmp@reencode	803, 812, 834, 997	
\hyxmp@mm@schema	... <u>404</u>	\hyxmp@xmlify
..... <u>898</u> , 1146	\hyxmp@rights 188, <u>407</u> ,	
\hyxmp@modulo@a	. 871, 874, 878, 880	772, 781, 793,	
614, 633, 643, 649, 684	\hyxmp@seed@rng	801, 810, 833, 996	
\hyxmp@new@xml 622, 699, 708	\hyxmp@xmp@basic@schema <u>922</u> , 1144
728, 729	\hyxmp@seed@rng@i	\hyxmp@xmp@to@pdf@date 290, <u>293</u>
\hyxmp@num 624, <u>626</u>	\hyxmp@xmp@to@pdf@date@i 294, <u>296</u>
\hyxmp@one@token	\hyxmp@seed@string		
622, <u>626</u> , 1258, 697, <u>703</u>		
1259, 1267, 1268	\hyxmp@set@rand@num		
\hyxmp@padding 639, 647, 682		
\hyxmp@parse@time	\hyxmp@skiptorelax		
..... 267, <u>269</u> 552, <u>558</u>		
\hyxmp@parse@tz	\hyxmp@skipzeros		
.... 276, 279, <u>283</u>	... <u>510</u>		
\hyxmp@parse@tz@char	\hyxmp@SpaceOther		
..... 271, <u>273</u> 519, <u>532</u>		
\hyxmp@pdf@schema	\hyxmp@string		
..... <u>749</u> , 1139 <u>768</u>		
	\hyxmp@string@len		
 1237, <u>1252</u>		

pdfauthor (option) 3, 10, 11, 13, 17, 18, 21, 43, 64	pdflicenseurl (option) 4, 5, 11, 44, 63	\special 1238, 1246, 1275, 1281
pdfauthortitle (option) 4, 5, 11	pdfmark (option) 54	stringenc 14
pdfcaptionwriter (op- tion) 4, 5	\pdfmark 1207, 1210, 1214, 1224, 1228, 1232	\StringEncodingConvert 412, 418, 429, 432, 527
\pdfcatalog 1196	pdfmetadate (option) 4, 5, 15, 65	Subject 8
\pdfcompresslevel . 1190	pdfmetalang (option) 4, 6, 11, 12, 34, 62, 65	T
pdfcontactaddress (op- tion) 4, 5, 10	\pdfminorversion . . . 759	TeX 12, 13, 26, 27, 29, 31, 32, 35, 36, 38, 45, 55, 56
pdfcontactcity (option) 4, 5	pdfmoddate (option) 4, 5, 65	texdate 12
pdfcontactcountry (op- tion) 4, 5	\pdfobj 1192	Text 50
pdfcontactemail (op- tion) 4, 5	pdfproducer (option) 4, 13	\textunderscore 19, 20, 22
pdfcontactphone (op- tion) 4, 5	pdfsource (option) 4, 6, 65	textures (option) 54
pdfcontactpostcode (op- tion) 4, 5	\pdfstringdef 21	\time 364, 372
pdfcontactregion (op- tion) 4, 5	pdfsubject (option) 4, 13, 43	Title 8
pdfcontacturl (option) 4, 5, 11	pdfTeX 13, 31, 53, 56, 65	U
pdfcopyright (option) 4, 5, 43, 44, 63	pdftitle (option) . . 4, 11, 13, 21, 43, 64	Unicode . . 11, 14, 29– 33, 43, 48, 51, 56, 63
pdfcreationdate (option) 3, 5, 65	pdftype (option) . . 4, 6, 65	unicode (option) . . 11, 65
\pdfcreationdate 383, 916	\pdfvariable 762	URL 2, 5, 11, 15, 17, 44–46, 49
pdfdate (option) 4, 5, 11, 14, 38, 64	photoshop:AuthorsPosition 3, 46	UTF-16BE 30
PDFDocEncoding 17, 29, 30	photoshop:CaptionWriter 3, 46	UTF-32BE 29, 30
pdfdocumentid (option) 4, 6, 65, 66	PI 38	UTF-8 30
pdfescape 14	\ProcessKeyvalOptions 132	UUID 5, 6, 16, 35, 37, 38, 64
\pdfextension 1200, 1204	Producer 40	V
\pdffeedback 386, 919, 1204	ps2pdf (option) 54	\vfuzz 396
pdfinstanceid (option) 4, 6, 65, 66	Q	vtexpdfmark (option) . 54
pdfkeywords (option) 3, 10, 13, 17, 43	\Q 388, 397	X
pdflang (option) 3, 6, 11–13, 20, 34, 43, 65	R	\x 515
\pdflastobj 1196	RDF 42	xdvipdfmx 10, 56
pdfL ^A T _E X 3, 7, 10, 27, 40, 45	rdf:li 2	X _q L ^A T _E X 7, 10, 27, 40, 45, 65
	rdf:Seq 2	X _q L _E X 14, 29, 32, 56, 62, 63
	\renewcommand 133	XML 1, 2, 10, 21, 29–32, 38–43, 47, 49, 51, 65
	\RequirePackage 7, 10–15, 1185	XMP . . 1–3, 5–15, 20– 25, 27, 28, 31, 34, 35, 38–42, 44, 45, 49, 53, 54, 56, 62–65
	S	xmp:BaseURL 2
	\SE->pdfdoc@03 405	xmp:CreateDate 2
	\SE->pdfdoc@15 406	xmp:CreatorTool 3
	\setkeys 612	

xmp:MetadataDate	2	xmpMM:DocumentID	2, 35, 45	xmpRights:WebStatement	2, 44, 63
xmp:ModifyDate	2	xmpMM:InstanceID	2, 35, 45	\xmptilde	<u>243</u>
\xmpcomma	70, 73, <u>99</u> , <u>120</u> , <u>233</u>	\xmpquote		\XMPTruncateList	<u>247</u>
xmpincl	3	\XMPLangAlt	71, 74, <u>99</u> , <u>120</u> , <u>242</u>		Y
\xmplinesep <u>977</u> , 993, <u>1028</u>		xmpRights:Marked 2, 44, 63		\year	353