

The hyperxmp package^{*}

Scott Pakin
scott+hyxmp@pakin.org

February 23, 2017

Abstract

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

1 Introduction

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

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

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

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

^{*}This document corresponds to `hyperxmp` v3.2, dated 2017/02/23.

```

    </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`:

- `pdfaconformance`
- `pdfapart`
- `pdfauthortitle`
- `pdfcaptionwriter`
- `pdfcontactaddress`
- `pdfcontactcity`
- `pdfcontactcountry`
- `pdfcontactemail`
- `pdfcontactphone`
- `pdfcontactpostcode`
- `pdfcontactregion`
- `pdfcontacturl`
- `pdfcopyright`
- `pdfdate`
- `pdflicenseurl`
- `pdfmetadate`
- `pdfmetalang`
- `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 L^AT_EX `\date` command, and, like `\date`, defaults to the date the document was built. It must be specified in either XMP format [4] or PDF format [3]. XMP dates are written in the form `YYYY-MM-DDThh:mm:ss+TT:tt`.¹ A W3C recommendation [12] discusses this format in more detail, but as an example, 14 hours, 15 minutes, 9 seconds past midnight U.S. Mountain Daylight Time (UTC-6) on the 23rd day of September in the year 2014 should be written as `2014-09-23T14:15:09-06:00`. This can be truncated (with loss of information) to `2014-09-23T14:15:09` or `2014-09-23T14:15` or `2014-09-23` or `2014-09` or `2014` but no other subsets. PDF dates are written in the form `D:YYYYMMDDhhmmss+TT’tt’`. The same date in the preceding example would be written as `D:20140923141509-06’00’` in PDF format.

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

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

`pdfmetalang` indicates the natural language in which the metadata is written, typically as 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 is always included in addition to the specified metadata language, as the user reading the document may not have specified a language preference.

`pdftype` describes the type of document being produced. This refers

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

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{hyperref}
\title{%
  On a heuristic viewpoint concerning the production and
  transformation of light}
\author{Albert Einstein}
\date{March 17, 1905}
\hypersetup{%
  pdftitle={%
    On a heuristic viewpoint concerning the production and
    transformation of light},
  pdfauthor={Albert Einstein},
  pdfauthortitle={Technical Assistant, Level III},
  pdfdate={1905-03-17},
  pdfcopyright={Copyright (C) 1905, Albert Einstein},
  pdfsubject={photoelectric effect},
  pdfkeywords={energy quanta, Hertz effect, quantum physics},
  pdflicenseurl={http://creativecommons.org/licenses/by-nc-nd/3.0/},
  pdfcaptionwriter={Scott Pakin},
  pdfcontactaddress={Kramgasse 49},
  pdfcontactcity={Bern},
  pdfcontactpostcode={3011},
  pdfcontactcountry={Switzerland},
  pdfcontactphone={031 312 00 91},
  pdfcontactemail={aeinstein@ipi.ch},
  pdfcontacturl={%
    http://einstein.biz/,
    https://www.facebook.com/AlbertEinstein
  },
  pdflang={en},
  baseurl={http://mirror.ctan.org/macros/latex/contrib/hyperxmp/}
}
```

```

\begin{document}
\maketitle
A profound formal difference exists between the theoretical
concepts that physicists have formed about gases and other
ponderable bodies, and Maxwell's theory of electromagnetic
processes in so-called empty space\dots
\end{document}

```

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

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

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

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

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

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

```
\XMPTruncateList{pdfauthor}
```

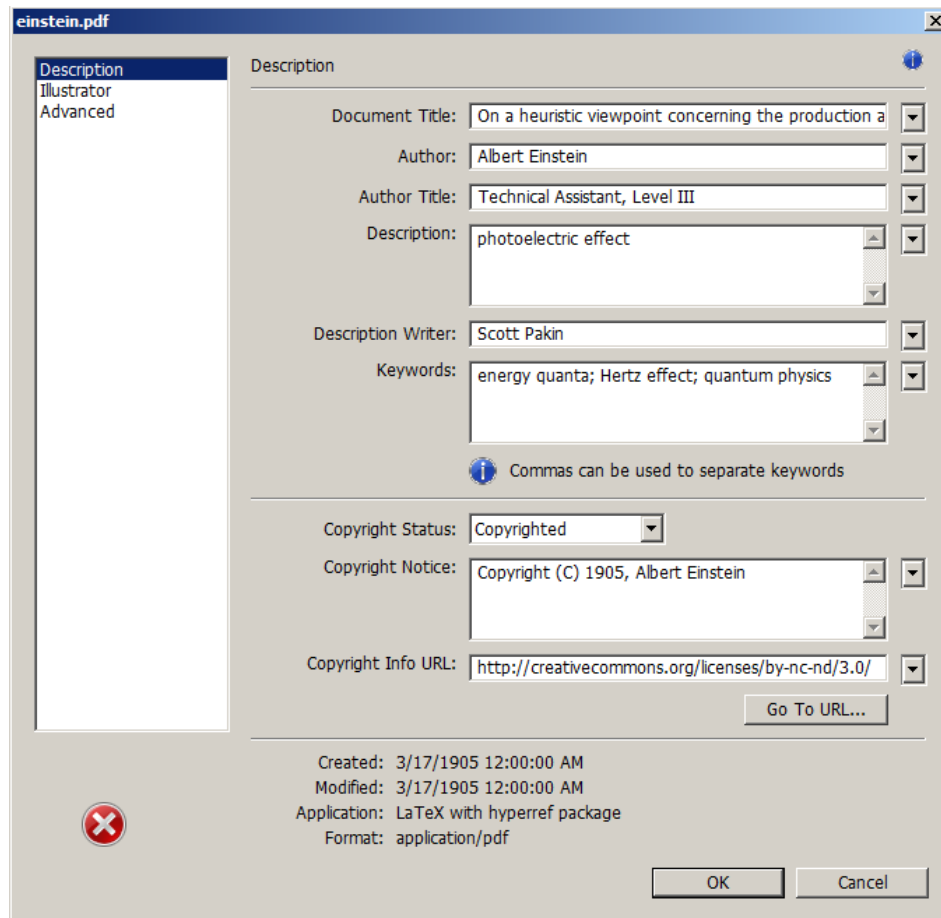


Figure 1: XMP metadata as it appears in Adobe Acrobat

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

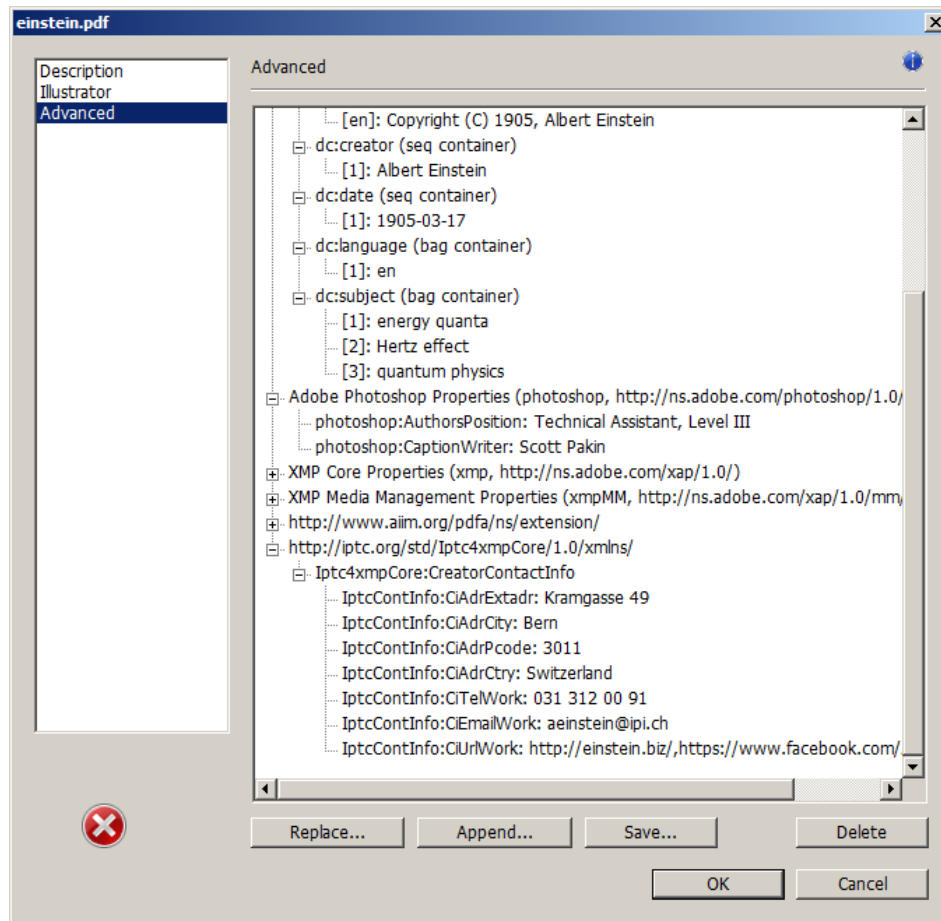


Figure 2: Additional XMP metadata as it appears in Adobe Acrobat

commas appearing in `pdfcontactaddress`'s argument with semicolons:

```
\renewcommand*{\xmllinesep}{;}}
```

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 `Lua \LaTeX` earlier than 0.85, `hyperxmp` leaves all PDF objects uncompressed. This is due to `Lua \LaTeX` treating object compression as a global parameter, unlike `pdf \LaTeX` , which treats it as a local parameter. Hence, when `hyperxmp` requests that the XMP packet be left uncompressed,

Lua \LaTeX 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 Lua \LaTeX v0.85 onwards.

2. X \LaTeX (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., Lua \LaTeX), (2) pass the `--output-driver="xdvipdfmx -z0"` option to X \LaTeX 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.

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 `pdfTeX`, the standard `\AtEndDocument` works well enough. For all the other backends we use `\AtEndDvi` from the `atenddvi` package, which is more robust but requires an addition \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 page 4. For consistency with `hyperref`'s document-metadata naming conventions (which are in turn based on L^AT_EX's document-metadata naming conventions), we do not prefix metadata-related macro names with our package-specific `\hyxmp@` prefix. That is, we use names like `\@pdfcopyright` instead of `\hyxmp@pdfcopyright`.

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

```

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

```

```

\hyxmp@pdfstringdef
\hyxmp@textunderscore

```

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

```

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

```

```

\@pdfdatetime

```

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

```

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

```

```

\next

```

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

```

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

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

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

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

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

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

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

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

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

```

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

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

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

```
57 \def\@pdfaconformance{B}
58 \define@key{Hyp}{pdfaconformance}{\hymp@pdfstringdef\@pdfaconformance{#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.

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

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

```
67 \def\@pdfcontactcity{}
68 \define@key{Hyp}{pdfcontactcity}{\hymp@pdfstringdef\@pdfcontactcity{#1}}
```

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

```
69 \def\@pdfcontactregion{}
70 \define@key{Hyp}{pdfcontactregion}{\hymp@pdfstringdef\@pdfcontactregion{#1}}
```

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

```
71 \def\@pdfcontactpostcode{}
72 \define@key{Hyp}{pdfcontactpostcode}{\hymp@pdfstringdef\@pdfcontactpostcode{#1}}
```

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

```
73 \def\@pdfcontactcountry{}
74 \define@key{Hyp}{pdfcontactcountry}{\hymp@pdfstringdef\@pdfcontactcountry{#1}}
```

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

```
75 \def\@pdfcontactphone{}
76 \define@key{Hyp}{pdfcontactphone}{\hyxmp@pdfstringdef\@pdfcontactphone{#1}}
```

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

```
77 \def\@pdfcontactemail{}
78 \define@key{Hyp}{pdfcontactemail}{\hyxmp@pdfstringdef\@pdfcontactemail{#1}}
```

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

```
79 \def\@pdfcontacturl{}
80 \define@key{Hyp}{pdfcontacturl}{\hyxmp@pdfstringdef\@pdfcontacturl{#1}}
```

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

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

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

```
\hyxmp@pdfkeywords
81 \def\hyxmp@pdfauthor{}
82 \def\hyxmp@pdfkeywords{}
```

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

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

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

```
84 \ifundefined{KV@Hyp@pdfauthor}{\}%
85 \ifundefined{hyxmp@Hyp@pdfauthor}{%
86 \expandafter\let\expandafter\hyxmp@Hyp@pdfauthor
87 \csname KV@Hyp@pdfauthor\endcsname
88 }{}%
89 }%
```

`\KV@Hyp@pdfauthor` Redefine `\KV@Hyp@pdfauthor` to process its argument twice. The first time, `\xmpcomma` is defined as a placeholder character (`\hyxmp@comma`) and `\xmpquote`

```
\xmpquote
\hyxmp@and
\and
\hyxmp@pdfauthor
\@pdfauthor
```

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, `\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.

```

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

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

```

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

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

```

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



```

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

```

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

```

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

```

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

```

128 \let\hyxmp@hypersetup=\hypersetup
129 \def\hypersetup{%
130   \hyxmp@redefine@Hyp
131   \hyxmp@hypersetup
132 }

```

`\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.

```

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

```

```

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

```

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.

```

167 \AtBeginDocument{%
168 \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`.

```

169 \ifx\pdflang\relax
170 \let\pdflang=\@empty
171 \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.

```

172 \ifx\pdflang\@empty
173 \let\pdfmetalang=\hyxmp@xdefault
174 \else
175 \edef\pdfmetalang{\pdflang}%
176 \fi
177 \hyxmp@xmlify\pdfmetalang

```

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

```

178 \ifx\pdfdatetime\@empty
179 \else
180 \edef\hyxmp@today{\pdfdatetime}%
181 \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`.

```

182 \ifx\@pdftitle\@empty
183 \ifx\@title\@empty
184 \else
185 \hypersetup{pdftitle={\@title}}}%
186 \fi
187 \fi
188 \ifx\@pdfauthor\@empty
189 \ifx\@author\@empty
190 \else
191 \hypersetup{pdfauthor={\@author}}}%
192 \fi
193 \fi

```

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

```

194 \hyxmp@at@end{%
195 \hyxmp@find@metadata
196 \hyxmp@embed@packet
197 }%
198 }{%
199 \PackageWarningNoLine{hyperxmp}{%
200 \jobname.tex failed to include a\MessageBreak
201 \string\usepackage\string{hyperref\string}
202 in the preamble.\MessageBreak
203 Consequently, all hyperxmp functionality will be\MessageBreak
204 disabled}%
205 }%
206 }

```

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.)
207 \newcommand*{\hyxmp@commas@to@list}[2]{%

```

```

208 \gdef#1{%
209 \expandafter\hyxmp@commas@to@list@i\expandafter#1#2,,%
210 }

```

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

```

\next 211 \def\hyxmp@commas@to@list@i#1#2,{%
212 \gdef\hyxmp@sublist{#2}%
213 \ifx\hyxmp@sublist\@empty
214 \let\next=\relax
215 \else
216 \hyxmp@trimspaces\hyxmp@sublist
217 \@cons{#1}{\hyxmp@sublist}%
218 \def\next{\hyxmp@commas@to@list@i{#1}}%
219 \fi
220 \next
221 }

```

`\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.

```
222 \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.

```

223 \bgroup
224 \catcode'\^^C=11
225 \gdef\hyxmp@comma{^^C}
226 \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.

```

227 \bgroup
228 \catcode'\^^U=11
229 \gdef\hyxmp@uscore{^^U}
230 \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.

```
231 \let\xmpquote=\relax

```

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

```

232 \bgroup
233 \catcode'\~=12%
234 \gdef\xmptilde{~}%
235 \egroup

```

`\XMPTruncateList` As a workaround for Adobe Acrobat’s inability to display author lists correctly (cf. “Acrobat Author bug” on page 7) we introduce a hack that replaces a list with its first element. One can then write “`\XMPTruncateList{pdfauthor}`” and have Adobe Acrobat display the author list correctly. It’s sad that this is necessary, though.

```

236 \newcommand{\XMPTruncateList}[1]{%
237 \edef\hyxmp@temp@str{\csname hyxmp@#1\endcsname}%
238 \hyxmp@commas@to@list{\hyxmp@temp@list}{\hyxmp@temp@str}%
239 \def\@elt##1{%
240 \expandafter\gdef\csname @#1\endcsname{##1}%
241 \let\@elt=\@gobble
242 }
243 \hyxmp@temp@list
244 }}

```

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:20170223004110-07’00’) [3], while XMP timestamps are of the form “YYYY-MM-DDThh:mm:ss+TT:tt” (e.g., 2017-02-23T00:41:10-07:00) [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 245 \def\hyxmp@first@char#1{\hyxmp@first@char@i#1\relax}
246 \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.

```

247 \def\hyxmp@as@xmp@date#1{%
248 \expandafter\ifx\hyxmp@first@char@i#1\relax D%
249 \hyxmp@pdf@to@xmp@date{#1}%
250 \else
251 #1%
252 \fi
253 }

```

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

```

254 \def\hyxmp@pdf@to@xmp@date#1:#2#3#4#5#6#7#8#9{%
255   #2#3#4#5-#6#7-#8#9%
256   \hyxmp@parse@time
257 }

```

`\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`.

```

258 \def\hyxmp@parse@time#1#2#3#4#5#6{%
259   T#1#2:#3#4:#5#6%
260   \hyxmp@parse@tz@char
261 }

```

`\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.

```

262 \def\hyxmp@parse@tz@char#1{%
263   #1%
264   \ifx#1-%
265     \expandafter\hyxmp@parse@tz
266   \else
267     \ifx#1+%
268       \expandafter\hyxmp@parse@tz
269     \fi
270   \fi
271 }

```

`\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.

```

272 \def\hyxmp@parse@tz#1'#2' {%
273   #1:#2%
274 }

```

`\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.

```

275 \def\hyxmp@as@pdf@date#1{%
276   \expandafter\ifx\hyxmp@first@char@i#1\relax D%
277   #1%
278   \else

```

```

279     \hyxmp@xmp@to@pdf@date{#1}%
280   \fi
281 }

```

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

```

282 \def\hyxmp@xmp@to@pdf@date#1{%
283   D:\hyxmp@xmp@to@pdf@date@i#1\relax\relax
284 }

```

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

```

285 \def\hyxmp@xmp@to@pdf@date@i#1#2#3#4#5#6{%
286   #1#2#3#4%
287   \ifx#5-%
288     \expandafter\hyxmp@xmp@to@pdf@date@ii\expandafter#6%
289   \fi
290 }

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

`\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.

```

315 \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).

```

316 \def\hyxmp@xmp@to@pdf@date@vi#1#2#3#4{%
317   #1#2%
318   \ifx#3+%
319     +\expandafter\hyxmp@xmp@to@pdf@date@vii
320   \fi
321   \ifx#3-%
322     -\expandafter\hyxmp@xmp@to@pdf@date@vii
323   \fi
324   \ifx#3Z%
325     Z%
326   \fi
327   \ifx#3\relax
328     \expandafter\@hyxmp@gobbletwo
329   \fi
330   \@gobbletwo #4%
331 }

```

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

```

332 \def\hyxmp@xmp@to@pdf@date@vii#1\@gobbletwo#2#3#4#5{%
333   #2#3%
334   \ifx#4:%
335     \expandafter\hyxmp@xmp@to@pdf@date@viii\expandafter#5%
336   \fi
337 }

```

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

```

338 \def\hyxmp@xmp@to@pdf@date@viii#1#2#3#4{%
339   '#1#2'%
340 }

```

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

```

341 \def\hyxmp@today@define#1{%

```

The date is a straightforward representation of T_EX's `\year`, `\month`, and `\day` primitives, with the latter two zero-padded to two digits apiece.

```

342   \xdef#1{\the\year}%
343   \ifnum\month<10
344     \xdef#1{#1-0\the\month}%
345   \else

```



```

346     \xdef#1{#1-\the\month}%
347 \fi
348 \ifnum\day<10
349     \xdef#1{#1-0\the\day}%
350 \else
351     \xdef#1{#1-\the\day}%
352 \fi

```

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

```

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

```

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

```

368 \@ifundefined{pdffeedback}{%
369     \@ifundefined{pdfcreationdate}{%

```

Case 1: Neither `\pdffeedback` nor `\pdfcreationdate` is defined (Xe_{La}TeX and regular L^AT_EX).

```

370     \hyxmp@today@define\hyxmp@today
371 }{%

```

Case 2: `\pdfcreationdate` is defined (pdf_{La}TeX and pre-0.85 Lua_{La}TeX).

```

372     \edef\hyxmp@today{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
373 }%
374 }{%

```

Case 3: `\pdffeedback` is defined (Lua_{La}TeX 0.85+).

```

375     \edef\hyxmp@today{\expandafter\hyxmp@pdf@to@xmp@date\pdffeedback creationdate}%
376 }

```

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.

```
377 \catcode'\Q=3
\hyxmp@trimspaces\x redefines \x to have the same replacement text sans leading
and trailing space tokens.
378 \newcommand{\hyxmp@trimspaces}[1]{%
  Use grouping to emulate a multi-token afterassignment queue.
379 \begingroup
  Put “\toks 0 {” into the afterassignment queue.
380 \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.
381 \expandafter\hyxmp@trimb\expandafter\noexpand#1Q Q}%
  Transfer the trimmed text back into #1.
382 \edef#1{\the\toks0}%
383 }
```

`\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.

```
384 \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.

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

```

387 \newif\ifhyxmp@unicodetex
388 \ifnum64='^0040\relax
389   \hyxmp@unicodetextrue
390 \else
391   \hyxmp@unicodetexfalse
392 \fi

\hyxmp@reencode This is now a placeholder macro needed only for \@pdfmetalang in the
                 \begin{document}.
393 \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.
394 \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.
395 \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;.
396 \newcommand*{\hyxmp@xmlify}[1]{%
397   \gdef\hyxmp@xmlified{%
                Escaped PDF string → PDFDocEncoding/Unicode
398   \EdefUnescapeString\hyxmp@text{#1}%
399   \ifhyxmp@unicodetex
                PDFDocEncoding/Unicode → UTF-32BE
400     \hyxmp@is@unicode\hyxmp@text{%
401       \StringEncodingConvert
402       \hyxmp@text\hyxmp@text{utf16be}{utf32be}%
403     }{%
404       \ifxetex
405         \hyxmp@xetex@crap
406       \else
407         \StringEncodingConvert
408         \hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
409       \fi
410     }%
                UTF-32BE → UTF-32BE as hex string
411   \EdefEscapeHex\hyxmp@text{\hyxmp@text}%

```

UTF-32BE → XML in ASCII

```

412 \edef\hyxmp@text{%
413 \expandafter
414 }\expandafter\hyxmp@toxml@unicodetex\hyxmp@text
415 \relax\relax\relax\relax\relax\relax\relax
416 \else

```

PDFDocEncoding/Unicode → UTF-8

```

417 \hyxmp@is@unicode\hyxmp@text{%
418 \StringEncodingConvert
419 \hyxmp@text\hyxmp@text{utf16be}{utf8}%
420 }{%
421 \StringEncodingConvert
422 \hyxmp@text\hyxmp@text{pdfdoc}{utf8}%
423 }%

```

UTF-8 → UTF-8 as hex string

```

424 \EdefEscapeHex\hyxmp@text{\hyxmp@text}%

```

UTF-8 as hex string → XML in UTF-8 as hex string

```

425 \edef\hyxmp@text{%
426 \expandafter\hyxmp@toxml\hyxmp@text\@empty\@empty
427 }%

```

XML in UTF-8 as hex string → XML in UTF-8

```

428 \EdefUnescapeHex\hyxmp@text{\hyxmp@text}%
429 \fi
430 \global\let\hyxmp@xmlified\hyxmp@text
431 }

```

\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.

```

432 \begingroup
433 \lccode'\<=254 %
434 \lccode'\>=255 %
435 \catcode254=12 %
436 \catcode255=12 %
437 \lowercase{\endgroup
438 \def\hyxmp@is@unicode#1{%
439 \expandafter\hyxmp@@is@unicode#1<>\@nil
440 }%
441 \def\hyxmp@@is@unicode#1<>#2\@nil{%
442 \ifx\#1\%
443 \expandafter\@firstoftwo
444 \else
445 \expandafter\@secondoftwo
446 \fi
447 }%
448 }

```

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

```

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

```

459   \@ifundefined{pdfmark}{%
460     #1#2%
461   }{%
462     \ifnum"#1#2='\( %
463       5C28% \(
464     \else\ifnum"#1#2='\) %
465       5C29% \)
466     \else
467       #1#2%
468     \fi\fi
469   }%
470   \fi\fi\fi
471   \expandafter\hyxmp@toxml
472 \fi
473 }

```

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

`\hyxmp@text`

```

474 \def\hyxmp@toxml@unicodetex#1#2#3#4#5#6#7#8{%
475   \ifx#1\relax
476   \else
477     \ifnum"#1#2#3#4#5#6#7#8>127 %
478       \uccode'\*="#1#2#3#4#5#6#7#8\relax
479       \uppercase{%
480         \edef\hyxmp@text{\hyxmp@text *}%

```

```

481     }%
482     \else\ifnum"#7#8='< %
483         \edef\hyxmp@text{\hyxmp@text &lt;}%
484     \else\ifnum"#7#8='& %
485         \edef\hyxmp@text{\hyxmp@text &amp;}%
486     \else\ifnum"#7#8='> %
487         \edef\hyxmp@text{\hyxmp@text &gt;}%
488     \else\ifnum"#7#8='\ %
489         \edef\hyxmp@text{\hyxmp@text\space}%
490     \else
491         \uccode'\*="#7#8\relax
492         \uppercase{%
493             \edef\hyxmp@text{\hyxmp@text *}%
494         }%
495     \fi\fi\fi\fi\fi
496     \expandafter\hyxmp@toxml@unicodetex
497 \fi
498 }

```

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

```

499 \def\hyxmp@skipzeros#1{%
500     \ifx#10%
501         \expandafter\hyxmp@skipzeros
502     \fi
503 }

```

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

```

\hyxmp@xetex@crap \hyxmp@try
\hyxmp@crap@result \def\x#1{\endgroup
\hyxmp@text \def\hyxmp@xetex@crap{%
507     \edef\hyxmp@try{%
508         \expandafter\hyxmp@SpaceOther\hyxmp@text#1\@nil
509     }%
510     \let\hyxmp@crap@result=N%
511     \expandafter\hyxmp@crap@test\hyxmp@try\relax
512     \ifx\hyxmp@crap@result Y%
513         \let\hyxmp@text\@empty
514         \expandafter\hyxmp@crap@convert\hyxmp@try\relax
515     \else
516         \StringEncodingConvert\hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
517     \fi
518 }%
519 }
520 \x{ }

```

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

```

521 \begingroup
522 \catcode'\~ =12 %

```

```

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

```

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

```

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

```

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

```

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

```

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

```

\hyxmp@num 548 \def\hyxmp@crap@convert#1{%
\hyxmp@text 549   \ifx#1\relax
550   \else
551     \edef\hyxmp@num{\number'#1}%
552     \ifnum\hyxmp@num>"FFFFFF %
553       \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"1000000}\relax
554       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
555       \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"1000000}}%
556     \else
557       \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
558     \fi
559     \ifnum\hyxmp@num>"FFFF %
560       \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"10000}\relax
561       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
562       \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"10000}}%
563     \else
564       \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
565     \fi

```

```

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

```

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

```

582 \begingroup
583 \catcode0=12 %
584 \gdef\hyxmp@zero{^^00}%
585 \endgroup

```

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

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

```

586 \def\hyxmp@modulo@a#1{%
587 \@tempcntb=\@tempcnta
588 \divide\@tempcntb by #1
589 \multiply\@tempcntb by #1
590 \advance\@tempcnta by -\@tempcntb
591 }

```

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

```

\hyxmp@big@prime@ii 592 \def\hyxmp@big@prime{536870923}
593 \def\hyxmp@big@prime@ii{536870027}

```

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

```

\hyxmp@one@token 594 \def\hyxmp@seed@rng#1{%
595 \@tempcnta=\hyxmp@big@prime
596 \futurelet\hyxmp@one@token\hyxmp@seed@rng@i#1\@empty
597 }

```


`\hyxmp@seed@rng@i` Do all of the work for `\hyxmp@seed@rng`. For each character code c of the input

`\hyxmp@one@token` text, assign $\backslash@tempcnta \leftarrow 3 \cdot \backslash@tempcnta + c \pmod{\backslashhyxmp@big@prime}$.

```

\next 598 \def\hyxmp@seed@rng@i{%
599   \ifx\hyxmp@one@token\empty
600     \let\next=\relax
601   \else
602     \def\next##1{%
603       \multiply\@tempcnta by 3
604       \advance\@tempcnta by '##1
605       \hyxmp@modulo@a{\hyxmp@big@prime}%
606       \futurelet\hyxmp@one@token\hyxmp@seed@rng@i
607     }%
608   \fi
609 \next
610 }

```

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

```

611 \def\hyxmp@set@rand@num{%
612   \@tempcnta=\hyxmp@rand@num
613   \multiply\@tempcnta by 3
614   \advance\@tempcnta by \hyxmp@big@prime@ii
615   \hyxmp@modulo@a{\hyxmp@big@prime}%
616   \xdef\hyxmp@rand@num{\the\@tempcnta}%
617 }

```

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

```

618 \def\hyxmp@append@hex#1{%
619   \hyxmp@set@rand@num
620   \@tempcnta=\hyxmp@rand@num
621   \hyxmp@modulo@a{16}%
622   \ifnum\@tempcnta<10
623     \xdef#1{#1\the\@tempcnta}%
624   \else

```

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

```

625     \advance\@tempcnta by -10
626     \ifcase\@tempcnta
627       \xdef#1{#1a}%
628       \or\xdef#1{#1b}%
629       \or\xdef#1{#1c}%
630       \or\xdef#1{#1d}%
631       \or\xdef#1{#1e}%
632       \or\xdef#1{#1f}%
633     \fi
634   \fi
635 }

```

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

```
636 \def\hyxmp@append@hex@iii#1{%
637   \hyxmp@append@hex#1%
638   \hyxmp@append@hex#1%
639   \hyxmp@append@hex#1%
640 }
```

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

```
641 \def\hyxmp@append@hex@iv#1{%
642   \hyxmp@append@hex@iii#1%
643   \hyxmp@append@hex#1%
644 }
```

`\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`.

```
645 \def\hyxmp@create@uuid#1{%
646   \def#1{uuid:}%
647   \hyxmp@append@hex@iv#1%
648   \hyxmp@append@hex@iv#1%
649   \g@addto@macro#1{-}%
650   \hyxmp@append@hex@iv#1%
651   \g@addto@macro#1{-4}%
652   \hyxmp@append@hex@iii#1%
653   \g@addto@macro#1{-}%

```

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

```
654   \hyxmp@set@rand@num
655   \@tempcnta=\hyxmp@rand@num
656   \hyxmp@modulo@a{4}%
657   \ifcase\@tempcnta
658     \g@addto@macro#1{8}%
659     \or\g@addto@macro#1{9}%
660     \or\g@addto@macro#1{a}%
661     \or\g@addto@macro#1{b}%
662   \fi
663   \hyxmp@append@hex@iii#1%
664   \g@addto@macro#1{-}%
665   \hyxmp@append@hex@iv#1%
666   \hyxmp@append@hex@iv#1%
667   \hyxmp@append@hex@iv#1%
668 }
```

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

```
669 \newcommand*{\hyxmp@def@DocumentID}{%
```

```

670 \edef\hyxmp@seed@string{\jobname:\@pdftitle:\@pdfauthor}%
671 \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
672 \edef\hyxmp@rand@num{\the\@tempcnta}%
673 \hyxmp@create@uuid\hyxmp@DocumentID
674 }

```

`\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@create@uuid` to define `\hyxmp@InstanceID` as a random UUID.

```

675 \newcommand*{\hyxmp@def@InstanceID}{%
676 \edef\hyxmp@seed@string{\jobname:\@pdftitle:\@pdfauthor:\hyxmp@today}%
677 \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
678 \edef\hyxmp@rand@num{\the\@tempcnta}%
679 \hyxmp@create@uuid\hyxmp@InstanceID
680 }

```

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.

```

681 \newcommand*{\hyxmp@add@to@xml}[1]{%
682 \bgroup
683 \@tempcnta=0
684 \ifhyxmp@unicodetex
685 \@tempcntb=65536%
686 \else
687 \@tempcntb=256%
688 \fi
689 \loop
690 \lccode\@tempcnta=\@tempcnta
691 \advance\@tempcnta by 1
692 \ifnum\@tempcnta<\@tempcntb
693 \repeat

```

```

694 \lccode'\_='\ \relax
695 \lccode'\^^C='\ \relax
696 \lccode'\^^U='\ \relax
697 \lowercase{\xdef\hyxmp@new@xml{#1}}%
698 \xdef\hyxmp@xml{\hyxmp@xml\hyxmp@new@xml}%
699 \egroup
700 }

```

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

```

701 \bgroup
702 \catcode'\#=11
703 \gdef\hyxmp@hash{#}
704 \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@padding` is defined to contain 32 lines of 63 spaces and a newline apiece for a total of 2048 characters of whitespace.

```

705 \bgroup
706 \xdef\hyxmp@xml{}%
707 \hyxmp@add@to@xml{%
708 -----^^J%
709 }
710 \xdef\hyxmp@padding{\hyxmp@xml}%
711 \egroup
712 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
713 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
714 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
715 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
716 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}

```

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

```

717 \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.

```

718 \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.

```

719 \hyxmp@add@to@xml{%
720 -----<rdf:Description rdf:about=""^^J%
721 -----xmlns:pdf="http://ns.adobe.com/pdf/1.3/">^^J%
722 }%
723 \hyxmp@add@simple@var{pdf:Keywords}{@pdfkeywords}%
724 \hyxmp@add@simple@var{pdf:Producer}{@pdfproducer}%
725 \@ifundefined{pdfvariable}{%
726   \@ifundefined{pdfminorversion}{%

```

Case 1: Neither `\pdfvariable` nor `\pdfminorversion` is defined (X_YLaTeX and regular LaTeX).

```

727 }%

```

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

```

728   \hyxmp@add@simple{pdf:PDFVersion}{1.\the\pdfminorversion}%
729 }%
730 }%

```

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

```

731   \hyxmp@add@simple{pdf:PDFVersion}{1.\the\pdfvariable minorversion}%
732 }%
733 \hyxmp@add@to@xml{%
734 -----</rdf:Description>^^J%
735 }%
736 }

```

`\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`

```

737 \newcommand*{\hyxmp@add@simple}[2]{%
738   \edef\hyxmp@string{#2}%
739   \ifx\hyxmp@string\@empty
740   \else
741     \hyxmp@xmllify{\hyxmp@string}%
742     \hyxmp@add@to@xml{%
743 -----<#1>\hyxmp@xmllified</#1>^^J%
744   }%
745   \fi
746 }

```

`\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.

```

747 \newcommand*{\hyxmp@add@simple@var}[2]{%
748   \expandafter\ifx\csname#2\endcsname\relax
749   \else
750     \hyxmp@xmllify{\csname#2\endcsname}%

```

```

751 \hyxmp@add@to@xml{%
752 -----<#1>\hyxmp@xmlified</#1>^^J%
753 }%
754 \fi
755 }

```

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.

```

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

```

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

```

757 \@tempwafalse
758 \ifx#3\@empty
759 \else
760 \@tempwattrue
761 \fi
762 #1
763 \@tempwattrue
764 \fi

```

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

```

765 \if@tempswa
766 \hyxmp@xmlify{#3}%
767 \hyxmp@add@to@xml{%
768 -----<dc:#2>^^J%
769 -----<rdf:Alt>^^J%
770 }%
771 \ifx\@pdfmetalang\hyxmp@x@default
772 \else
773 \hyxmp@add@to@xml{%
774 -----<rdf:li xml:lang="\@pdfmetalang">\hyxmp@xmlified</rdf:li>^^J%
775 }%
776 \fi
777 \hyxmp@add@to@xml{%
778 -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@xmlified</rdf:li>^^J%
779 -----</rdf:Alt>^^J%
780 -----</dc:#2>^^J%
781 }%
782 \fi
783 }%

```

`\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.

```

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

```

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

```

785  \@tempswafalse
786  \ifx#4\@empty
787  \else
788    \@tempswatrue
789  \fi
790  #1
791    \@tempswatrue
792  \fi

```

Append the corresponding XML only if \@tempswatrue.

```

793  \if@tempswa
794    \hyxmp@add@to@xml{%
795  -----<dc:#2>^^J%
796  -----<rdf:#3>^^J%
797    }%
798  \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.

```

799    \hyxmp@xmlify{#4}%
800    \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
801    \def\@elt##1{%
802      \hyxmp@add@to@xml{%
803  -----<rdf:li>##1</rdf:li>^^J%
804      }%
805    }%
806    \hyxmp@list
807  \egroup
808  \hyxmp@add@to@xml{%
809  -----</rdf:#3>^^J%
810  -----</dc:#2>^^J%
811    }%
812  \fi
813 }

```

\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.

```

814 \newcommand*{\hyxmp@dc@schema}{%
815   \hyxmp@add@to@xml{%
816     -----<rdf:Description rdf:about=""^^J%
817     -----_xmlns:dc="http://purl.org/dc/elements/1.1/">^^J%

```

```

818 -----<dc:format>application/pdf</dc:format>^^J%
819  }%
820   \hyxmp@rdf@dc[\ifHy@pdfa]{title}{\@pdftitle}%
821   \hyxmp@rdf@dc[\ifHy@pdfa]{description}{\@pdfsubject}%
822   \hyxmp@rdf@dc{rights}{\@pdfcopyright}%
823   \hyxmp@list@to@xml[\ifHy@pdfa]{creator}{Seq}{\hyxmp@pdfauthor}%
824   \hyxmp@list@to@xml{subject}{Bag}{\hyxmp@pdfkeywords}%
825   \hyxmp@list@to@xml{date}{Seq}{\hyxmp@today}%
826   \hyxmp@list@to@xml{language}{Bag}{\@pdflang}%
827   \hyxmp@list@to@xml{type}{Bag}{\@pdftype}%
828   \hyxmp@add@simple{dc:source}{\jobname.tex}%
829   \hyxmp@add@to@xml{%
830 -----</rdf:Description>^^J%
831  }%
832 }

```

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.

```

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

```

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

```

834 \let\hyxmp@rights=\@empty
835 \ifx\@pdflicenseurl\@empty
836 \else
837   \def\hyxmp@rights{YES}%
838 \fi
839 \ifx\@pdfcopyright\@empty
840 \else
841   \def\hyxmp@rights{YES}%
842 \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.

```

843 \ifx\hyxmp@rights\@empty
844 \else

```

Header

```

845   \hyxmp@add@to@xml{%
846 -----<rdf:Description rdf:about=""^^J%
847 -----xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/">^^J%
848   }%

```

Copyright indication

```

849   \ifx\@pdfcopyright\@empty
850   \else

```



```

851      \hyxmp@add@to@xml{%
852  -----<xmpRights:Marked>True</xmpRights:Marked>^^J%
853      }%
854  \fi

License URL

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

Trailer

856      \hyxmp@add@to@xml{%
857  -----</rdf:Description>^^J%
858      }%
859  \fi
860 }

```

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.

```

861 \gdef\hyxmp@mm@schema{%
862   \hyxmp@def@DocumentID
863   \hyxmp@def@InstanceID
864   \hyxmp@add@to@xml{%
865     -----<rdf:Description rdf:about=""^^J%
866     -----_xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"^^J%
867     -----<xmpMM:DocumentID>\hyxmp@DocumentID</xmpMM:DocumentID>^^J%
868     -----<xmpMM:InstanceID>\hyxmp@InstanceID</xmpMM:InstanceID>^^J%
869     -----</rdf:Description>^^J%
870   }%
871 }

```

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.

```

872 \newcommand*{\hyxmp@define@createdate}{%
873   \@ifundefined{pdffeedback}{%
874     \@ifundefined{pdfcreationdate}{%

```

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

```

875     \hyxmp@today@define\hyxmp@createdate
876   }{%

```

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

```

877     \edef\hyxmp@createdate{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%

```

```

878 }%
879 }{%
    Case 3: \pdffeedback is defined (LuaLATEX 0.85+).
880 \edef\hyxmp@createdate{\expandafter\hyxmp@pdf@to@xmp@date\pdffeedback creationdate}%
881 }%
882 }

```

`\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`.

```

883 \newcommand*{\hyxmp@xmp@basic@schema}{%
884   \hyxmp@add@to@xml{%
885     <rdf:Description rdf:about=""^^J%
886     <-----xmlns:xmp="http://ns.adobe.com/xap/1.0/">^^J%
887   }%
888   \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`.

```

889 \ifundefined{@pdfcreationdate}{%
890   \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@createdate}%
891 }{%
892   \ifx\@pdfcreationdate\@empty
893     \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@createdate}%
894   \else
895     \hyxmp@add@simple{xmp:CreateDate}{%
896       \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfcreationdate}}%
897   \fi
898 }%

```

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

```

899 \ifundefined{@pdfmoddate}{%
900   \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@createdate}%
901 }{%
902   \ifx\@pdfmoddate\@empty
903     \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@createdate}%
904   \else
905     \hyxmp@add@simple{xmp:ModifyDate}{%
906       \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfmoddate}}%
907   \fi
908 }%

```

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

```

909 \ifx\@pdfmetadatetime\@empty
910   \hyxmp@add@simple{xmp:MetadataDate}{\hyxmp@createdate}%
911 \else
912   \hyxmp@add@simple{xmp:MetadataDate}{\@pdfmetadatetime}%
913 \fi

```

Define the creation tool and the base URL.

```

914 \hyxmp@add@simple{xmp:CreatorTool}{\@pdfcreator}%
915 \hyxmp@add@simple{xmp:BaseURL}{\@baseurl}%
916 \hyxmp@add@to@xml{%
917 -----</rdf:Description>^^J%
918 }%
919 }

```

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.

```

920 \gdef\hyxmp@photoshop@schema{%
921 \edef\hyxmp@photoshop@data{\@pdfauthortitle\@pdfcaptionwriter}%
922 \ifx\hyxmp@photoshop@data\@empty
923 \else
924 \hyxmp@add@to@xml{%
925 -----<rdf:Description rdf:about=""^^J%
926 -----_xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/">^^J%
927 }%
928 \fi
929 \hyxmp@add@simple{photoshop:AuthorsPosition}{\@pdfauthortitle}%
930 \hyxmp@add@simple{photoshop:CaptionWriter}{\@pdfcaptionwriter}%
931 \ifx\hyxmp@photoshop@data\@empty
932 \else
933 \hyxmp@add@to@xml{%
934 -----</rdf:Description>^^J%
935 }%
936 \fi
937 }

```

3.5.8 The IPTC Photo Metadata schema

`\xmplinesep` Lines in multiline fields are separated by `\xmplinesep` in the generated XML. This defaults to an LF (`^^J`) character but written as an XML character entity for consistency across operating systems.

```

938 \begingroup
939 \catcode'\&=12
940 \catcode'\#=12
941 \gdef\xmplinesep{&#xA;}
942 \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.

```

943 \newcommand*{\hyxmp@list@to@lines}[2]{%
944 \ifx#2\@empty
945 \else

```

```

946     \bgroup
947     \hyxmp@add@to@xml{%
948     -----<#1>%
949     }%

\@elt@first The first element of the list is output as is.
950     \def\@elt@first##1{%
951     \hyxmp@add@to@xml{##1}%
952     \let\@elt=\@elt@rest
953     }%

\@elt@rest The remaining elements of the list are output with a preceding line separator
(\xmplinesep).
954     \def\@elt@rest##1{%
955     \hyxmp@add@to@xml{\xmplinesep##1}%
956     }%

\@elt Re-encode the text from Unicode if necessary. Then redefine \@elt to insert a line
separator between terms.
957     \let\@elt=\@elt@first
958     \hyxmp@xmlify{#2}%
959     \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
960     \hyxmp@list
961     \hyxmp@add@to@xml{</#1>^^J}%
962     \egroup
963     \fi
964 }

\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 lptc4xmpCore:CreatorContactInfo/CiAdrExtadr,
lptc4xmpCore:CreatorContactInfo/CiAdrCity, lptc4xmpCore:CreatorContactInfo/
CiAdrRegion, lptc4xmpCore:CreatorContactInfo/CiAdrPcode,
lptc4xmpCore:CreatorContactInfo/CiAdrCtry, lptc4xmpCore:CreatorContactInfo/
CiTelWork, lptc4xmpCore:CreatorContactInfo/CiEmailWork, and
lptc4xmpCore:CreatorContactInfo/CiUrlWork properties.
965 \gdef\hyxmp@photometa@schema{%
966 \edef\hyxmp@photometa@data{%
967 \@pdfcontactaddress
968 \@pdfcontactcity
969 \@pdfcontactregion
970 \@pdfcontactpostcode
971 \@pdfcontactcountry
972 \@pdfcontactphone
973 \@pdfcontactemail
974 \@pdfcontacturl
975 }%
976 \ifx\hyxmp@photometa@data\@empty

```

```

977 \else
978   \hyxmp@iptc@extensions
979   \hyxmp@add@to@xml{%
980 -----<rdf:Description rdf:about=""^^J%
981 -----xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"^^J%
982 -----<Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">^^J%
983   }%
984 \fi
985 \hyxmp@list@to@lines{Iptc4xmpCore:CiAdrExtadr}{\@pdfcontactaddress}%
986 \hyxmp@add@simple{Iptc4xmpCore:CiAdrCity}{\@pdfcontactcity}%
987 \hyxmp@add@simple{Iptc4xmpCore:CiAdrRegion}{\@pdfcontactregion}%
988 \hyxmp@add@simple{Iptc4xmpCore:CiAdrPcode}{\@pdfcontactpostcode}%
989 \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.

```

990 \bgroup
991   \def\xmplinesep{,}%
992   \hyxmp@list@to@lines{Iptc4xmpCore:CiTelWork}{\@pdfcontactphone}%
993   \hyxmp@list@to@lines{Iptc4xmpCore:CiEmailWork}{\@pdfcontactemail}%
994   \hyxmp@list@to@lines{Iptc4xmpCore:CiUrlWork}{\@pdfcontacturl}%
995 \egroup
996 \ifx\hyxmp@photometa@data\@empty
997 \else
998   \hyxmp@add@to@xml{%
999 -----</Iptc4xmpCore:CreatorContactInfo>^^J%
1000 -----</rdf:Description>^^J%
1001   }%
1002 \fi
1003 }

```

\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.

```

1004 \newcommand*{\hyxmp@iptc@extensions}{%
1005   \hyxmp@add@to@xml{%
1006 -----<rdf:Description rdf:about=""^^J%
1007 -----xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"^^J%
1008 -----xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema\hyxmp@hash"^^J%
1009 -----xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property\hyxmp@hash"^^J%

```

```

1010 _____xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type\hyxmp@hash"^^J%
1011 _____xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field\hyxmp@hash"^^J%
1012 _____<pdfaExtension:schemas>^^J%
1013 _____<rdf:Bag>^^J%
1014 _____<rdf:li rdf:parseType="Resource">^^J%
1015 _____<pdfaSchema:schema>IPTC Core Schema</pdfaSchema:schema>^^J%
1016 _____<pdfaSchema:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaSchema:
1017 _____<pdfaSchema:prefix>Iptc4xmpCore</pdfaSchema:prefix>^^J%
1018 _____<pdfaSchema:property>^^J%
1019 _____<rdf:Seq>^^J%
1020 _____<rdf:li rdf:parseType="Resource">^^J%
1021 _____<pdfaProperty:name>CreatorContactInfo</pdfaProperty:name>^^J%
1022 _____<pdfaProperty:valueType>contactinfo</pdfaProperty:valueType>^^J%
1023 _____<pdfaProperty:category>external</pdfaProperty:category>^^J%
1024 _____<pdfaProperty:description>contact information for the document's creator</p
1025 _____</rdf:li>^^J%
1026 _____</rdf:Seq>^^J%
1027 _____</pdfaSchema:property>^^J%
1028 _____<pdfaSchema:valueType>^^J%
1029 _____<rdf:Seq>^^J%
1030 _____<rdf:li rdf:parseType="Resource">^^J%
1031 _____<pdfaType:type>contactinfo</pdfaType:type>^^J%
1032 _____<pdfaType:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaTyp
1033 _____<pdfaType:prefix>Iptc4xmpCore</pdfaType:prefix>^^J%
1034 _____<pdfaType:description>contact information</pdfaType:description>^^J%
1035 _____<pdfaType:field>^^J%
1036 _____<rdf:Seq>^^J%
1037 }%

1038 \hyxmp@text@resource{CiAdrExtadr}{contact address}%
1039 \hyxmp@text@resource{CiAdrCity}{contact city}%
1040 \hyxmp@text@resource{CiAdrRegion}{contact region}%
1041 \hyxmp@text@resource{CiAdrPcode}{contact postal code}%
1042 \hyxmp@text@resource{CiAdrCtry}{contact country}%
1043 \hyxmp@text@resource{CiTelWork}{contact telephone number}%
1044 \hyxmp@text@resource{CiEmailWork}{contact email address}%
1045 \hyxmp@text@resource{CiUrlWork}{contact url}%

1046 \hyxmp@add@to+xml{%
1047 _____</rdf:Seq>^^J%
1048 _____</pdfaType:field>^^J%
1049 _____</rdf:li>^^J%
1050 _____</rdf:Seq>^^J%
1051 _____</pdfaSchema:valueType>^^J%
1052 _____</rdf:li>^^J%
1053 _____</rdf:Bag>^^J%
1054 _____</pdfaExtension:schemas>^^J%
1055 _____</rdf:Description>^^J%
1056 }%
1057 }

```

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

```

1058 \newcommand*{\hyxmp@text@resource}[2]{%
1059   \hyxmp@add@to@xml{%
1060     -----<rdf:li rdf:parseType="Resource">^^J%
1061     -----<pdfaField:name>#1</pdfaField:name>^^J%
1062     -----<pdfaField:valueType>Text</pdfaField:valueType>^^J%
1063     -----<pdfaField:description>#2</pdfaField:description>^^J%
1064     -----</rdf:li>^^J%
1065   }
1066 }
```

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`.

```

1067 \newcommand*{\hyxmp@pdfa@id@schema}{%
1068   \ifHy@pdfa
1069     \hyxmp@add@to@xml{%
1070       -----<rdf:Description rdf:about=""^^J%
1071       -----xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/">^^J%
1072     }%
1073     \hyxmp@add@simple{pdfaid:part}{\@pdfapart}%
1074     \hyxmp@add@simple{pdfaid:conformance}{\@pdfaconformance}%
1075     \hyxmp@add@to@xml{%
1076       -----</rdf:Description>^^J%
1077     }%
1078   \fi
1079 }
```

3.5.10 Combining schemata into an XMP packet

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

```

1080 \begingroup
1081   \ifhyxmp@unicodetex
1082     \lccode'\!="FEFF %
1083     \lowercase{%
1084       \gdef\hyxmp@bom{!}
1085     }%
1086   \else
1087     \catcode'\^^ef=12
1088     \catcode'\^^bb=12
1089     \catcode'\^^bf=12
1090     \gdef\hyxmp@bom{^^ef^^bb^^bf}%
1091   \fi
1092 \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.

```

1093 \def\hyxmp@construct@packet{%
1094   \gdef\hyxmp@xml{%
1095     \hyxmp@add@to@xml{<?xpacket begin="\hyxmp@bom" %
1096 id="W5M0MpCehiHzreSzNTczkc9d"?>^^J%
1097 <x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="3.1-702">^^J%
1098 ___<rdf:RDF
1099 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\hyxmp@hash">^^J%
1100   }%
1101   \hyxmp@pdf@schema
1102   \hyxmp@xmpRights@schema
1103   \hyxmp@dc@schema
1104   \hyxmp@photoshop@schema
1105   \hyxmp@photometa@schema
1106   \hyxmp@xmp@basic@schema
1107   \hyxmp@pdfa@id@schema
1108   \hyxmp@mm@schema
1109   \hyxmp@add@to@xml{%
1110 ___</rdf:RDF>^^J%
1111 </x:xmpmeta>^^J%
1112 \hyxmp@padding
1113 <?xpacket end="w"?>^^J%
1114   }%
1115 }
```

3.6 Embedding the XMP packet

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

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

```

1116 \newcommand*{\hyxmp@embed@packet}{%
1117   \hyxmp@construct@packet
1118   \def\hyxmp@driver{hpdftex}%
1119   \ifx\hyxmp@driver\Hy@driver
1120     \hyxmp@embed@packet@pdftex
1121   \else
1122     \def\hyxmp@driver{hluatex}%
1123     \ifx\hyxmp@driver\Hy@driver
1124       \hyxmp@embed@packet@luatex
1125     \else
1126       \def\hyxmp@driver{hdvipdfm}%
1127       \ifx\hyxmp@driver\Hy@driver
1128         \hyxmp@embed@packet@dvipdfm
1129       \else
1130         \def\hyxmp@driver{hxdetex}%

```



```

1131     \ifx\hyxmp@driver\Hy@driver
1132     \hyxmp@embed@packet@xetex
1133   \else
1134     \@ifundefined{pdfmark}{%
1135       \PackageWarningNoLine{hyperxmp}{%
1136         Unrecognized hyperref driver ‘\Hy@driver’.\MessageBreak
1137         \jobname.tex’s XMP metadata will *not* be\MessageBreak
1138         embedded in the resulting file}%
1139     }{%
1140       \hyxmp@embed@packet@pdfmark
1141     }%
1142   \fi
1143 \fi
1144 \fi
1145 \fi
1146 }

```

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.

```

1147 \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.

```

1148 \newcommand*{\hyxmp@embed@packet@pdftex}{%
1149   \bgroup
1150   \ifluatex
1151   \else
1152     \pdfcompresslevel=0
1153   \fi
1154   \immediate\pdfobj \ifluatex uncompressed\fi stream attr {%
1155     /Type /Metadata
1156     /Subtype /XML
1157   }{\hyxmp@xml}%
1158   \pdfcatalog {/Metadata \the\pdflastobj\space 0 R}%

```

```

1159 \egroup
1160 }

```

3.6.2 Embedding using LuaTeX 0.85+

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

```

1161 \newcommand*{\hyxmp@embed@packet@luatex}{%
1162 \immediate\pdfextension obj uncompressed stream attr {%
1163   /Type /Metadata
1164   /Subtype /XML
1165 }{\hyxmp@xml}%
1166 \pdfextension catalog {/Metadata \the\numexpr\pdffeedback lastobj\relax\space 0 R}%
1167 }

```

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.

```

1168 \newcommand*{\hyxmp@embed@packet@pdfmark}{%
1169 \pdfmark{%
1170   pdfmark=/NamespacePush
1171 }%
1172 \pdfmark{%
1173   pdfmark=/OBJ,
1174   Raw={/_objdef \string\hyxmp@Metadata\string} /type /stream}%
1175 }%
1176 \pdfmark{%
1177   pdfmark=/PUT,
1178   Raw={\string\hyxmp@Metadata\string}
1179   2 dict begin
1180     /Type /Metadata def
1181     /Subtype /XML def
1182     currentdict
1183   end
1184 }%
1185 }%
1186 \pdfmark{%
1187   pdfmark=/PUT,
1188   Raw={\string\hyxmp@Metadata\string} (\hyxmp@xml)}%
1189 }%
1190 \pdfmark{%
1191   pdfmark=/Metadata,
1192   Raw={\string\Catalog\string} \string\hyxmp@Metadata\string}%
1193 }%
1194 \pdfmark{%
1195   pdfmark=/NamespacePop
1196 }%
1197 }

```

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.

```

1198 \newcommand*{\hyxmp@embed@packet@dvipdfm}{%
1199   \hyxmp@string@len{\hyxmp@xml}%
1200   \special{pdf: object @hyxmp@Metadata
1201     <<
1202       /Type /Metadata
1203       /Subtype /XML
1204       /Length \the\@tempcnta
1205     >>
1206     stream^^J\hyxmp@xml endstream%
1207   }%
1208   \special{pdf: docview
1209     <<
1210       /Metadata @hyxmp@Metadata
1211     >>
1212   }%
1213 }
```

`\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.

```

1214 \newcommand*{\hyxmp@string@len}[1]{%
1215   \@tempcnta=0
1216   \expandafter\hyxmp@count@spaces#1 {} %
1217   \expandafter\hyxmp@count@non@spaces#1{}%
1218 }
```

`\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.

```

1219 \def\hyxmp@count@spaces#1 {%
1220   \def\hyxmp@one@token{#1}%
1221   \ifx\hyxmp@one@token\@empty
1222     \advance\@tempcnta by -1
1223   \else
1224     \advance\@tempcnta by 1
1225     \expandafter\hyxmp@count@spaces
1226   \fi
1227 }
```

`\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.

```

1228 \newcommand*{\hyxmp@count@non@spaces}[1]{%
1229   \def\hyxmp@one@token{#1}%
1230   \ifx\hyxmp@one@token\@empty
1231     \else
1232       \advance\@tempcnta by 1
1233       \expandafter\hyxmp@count@non@spaces
1234     \fi
1235 }

```

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.

```

1236 \newcommand*{\hyxmp@embed@packet@xetex}{%
1237   \special{pdf:stream @hyxmp@Metadata (\hyxmp+xml)
1238     <<
1239       /Type /Metadata
1240       /Subtype /XML
1241     >>
1242   }%
1243   \special{pdf:put @catalog
1244     <<
1245       /Metadata @hyxmp@Metadata
1246     >>
1247   }%
1248 }

```

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.

```

1249 \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^ATeX

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:Alt>
      </dc:title>
      <dc:description>
        <rdf:Alt>
          <rdf:li xml:lang="en">photoelectric effect</rdf:li>
          <rdf:li xml:lang="x-default">photoelectric effect</rdf:li>
        </rdf:Alt>
      </dc:description>
      <dc:rights>
        <rdf:Alt>
          <rdf:li xml:lang="en">
            Copyright (C) 1905, Albert Einstein
```

```

        </rdf:li>
        <rdf:li xml:lang="x-default">
            Copyright (C) 1905, Albert Einstein
        </rdf:li>
    </rdf:Alt>
</dc:rights>
<dc:creator>
    <rdf:Seq>
        <rdf:li>Albert Einstein</rdf:li>
    </rdf:Seq>
</dc:creator>
<dc:subject>
    <rdf:Bag>
        <rdf:li>energy quanta</rdf:li>
        <rdf:li>Hertz effect</rdf:li>
        <rdf:li>quantum physics</rdf:li>
    </rdf:Bag>
</dc:subject>
<dc:date>
    <rdf:Seq>
        <rdf:li>1905-03-17</rdf:li>
    </rdf:Seq>
</dc:date>
<dc:language>
    <rdf:Bag>
        <rdf:li>en</rdf:li>
    </rdf:Bag>
</dc:language>
<dc:type>
    <rdf:Bag>
        <rdf:li>Text</rdf:li>
    </rdf:Bag>
</dc:type>
<dc:source>einstein.tex</dc:source>
</rdf:Description>
<rdf:Description rdf:about=""
    xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/">
    <photoshop:AuthorsPosition>
        Technical Assistant, Level III
    </photoshop:AuthorsPosition>
    <photoshop:CaptionWriter>Scott Pakin</photoshop:CaptionWriter>
</rdf:Description>
<rdf:Description rdf:about=""
    xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"
    xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema#"
    xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property#"

```

```

xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type#"
xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field#">
<pdfaExtension:schemas>
  <rdf:Bag>
    <rdf:li rdf:parseType="Resource">
      <pdfaSchema:schema>IPTC Core Schema</pdfaSchema:schema>
      <pdfaSchema:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaSchema: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:Seq>
            </pdfaType:field>
          </rdf:li>
        </rdf:Seq>
      </pdfaSchema:valueType>
    </rdf:li>
  </rdf:Bag>
</pdfaExtension:schemas>

```

```

        </rdf:li>
        <rdf:li rdf:parseType="Resource">
            <pdfaField:name>CiAdrCtry</pdfaField:name>
            <pdfaField:valueType>Text</pdfaField:valueType>
            <pdfaField:description>contact country</pdfaField:description>
        </rdf:li>
        <rdf:li rdf:parseType="Resource">
            <pdfaField:name>CiTelWork</pdfaField:name>
            <pdfaField:valueType>Text</pdfaField:valueType>
            <pdfaField:description>contact telephone number</pdfaField:description>
        </rdf:li>
        <rdf:li rdf:parseType="Resource">
            <pdfaField:name>CiEmailWork</pdfaField:name>
            <pdfaField:valueType>Text</pdfaField:valueType>
            <pdfaField:description>contact email address</pdfaField:description>
        </rdf:li>
        <rdf:li rdf:parseType="Resource">
            <pdfaField:name>CiUrlWork</pdfaField:name>
            <pdfaField:valueType>Text</pdfaField:valueType>
            <pdfaField:description>contact url</pdfaField:description>
        </rdf:li>
    </rdf:Seq>
</pdfaType:field>
</rdf:li>
</rdf:Seq>
</pdfaSchema:valueType>
</rdf:li>
</rdf:Bag>
</pdfaExtension:schemas>
</rdf:Description>
<rdf:Description rdf:about=""
    xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/">
    <Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">
        <Iptc4xmpCore:CiAdrExtadr>Kramgasse 49</Iptc4xmpCore:CiAdrExtadr>
        <Iptc4xmpCore:CiAdrCity>Bern</Iptc4xmpCore:CiAdrCity>
        <Iptc4xmpCore:CiAdrPcode>3011</Iptc4xmpCore:CiAdrPcode>
        <Iptc4xmpCore:CiAdrCtry>Switzerland</Iptc4xmpCore:CiAdrCtry>
        <Iptc4xmpCore:CiTelWork>031 312 00 91</Iptc4xmpCore:CiTelWork>
        <Iptc4xmpCore:CiEmailWork>aeinstein@ipi.ch</Iptc4xmpCore:CiEmailWork>
        <Iptc4xmpCore:CiUrlWork>
            http://einstein.biz/,
            https://www.facebook.com/AlbertEinstein
        </Iptc4xmpCore:CiUrlWork>
    </Iptc4xmpCore:CreatorContactInfo>
</rdf:Description>
<rdf:Description rdf:about=""

```



```

xmlns:xmp="http://ns.adobe.com/xap/1.0/">
<xmp:CreateDate>2017-02-23T00:41:10-07:00</xmp:CreateDate>
<xmp:ModifyDate>2017-02-23T00:41:10-07:00</xmp:ModifyDate>
<xmp:MetadataDate>2017-02-23T00:41:10-07:00</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:0595fdce-41dc-e4c4-6c418dc4ce46
</xmpMM:DocumentID>
<xmpMM:InstanceID>
uuid:efd754c4-1d7f-200a-ef754ce413ea
</xmpMM:InstanceID>
</rdf:Description>
</rdf:RDF>
</x:xmpmeta>
<?xpacket end="w"?>

```

References

- [1] Adobe Systems, Inc., San Jose, California. *Adobe Acrobat X SDK Help, pdfmark Reference*. Available from <http://www.adobe.com/devnet/acrobat/documentation.html>.
- [2] Adobe Systems, Inc. *PostScript Language Reference Manual*. Addison-Wesley, 2nd edition, January 1996, ISBN: 0-201-18127-4.
- [3] Adobe Systems, Inc., San Jose, California. *Document Management—Portable Document Format—Part 1: PDF 1.7*, July 2008. ISO 32000-1 standard document. Available from http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000_2008.pdf.
- [4] Adobe Systems, Inc., San Jose, California. *XMP Specification Part 1: Data model, Serialization, and Core Properties*, April 2012. Available from <http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/xmp/pdfs/cc-201306/XMPSpecificationPart1.pdf>.
- [5] DCMI Usage Board *DCMI Metadata Terms*, June 14, 2012. Available from <http://dublincore.org/documents/dcmi-terms/>.
- [6] Michael Downes. Around the bend #15, answers, 4th (last) installment. `comp.text.tex` newsgroup posting, January 3, 1994.

Archived by Google at <http://groups.google.com/group/comp.text.tex/msg/7da7643b9e8f3b48>.

- [7] International Press Telecommunications Council. *IPTC Photo Metadata: Core 1.1/Extension 1.1*, July 2010. Revision 1. Available from http://www.iptc.org/std/photometadata/specification/IPTC-PhotoMetadata-201007_1.pdf.
- [8] Internet Assigned Numbers Authority. Language subtag registry, January 11, 2011. Available from <http://www.iana.org/assignments/language-subtag-registry>.
- [9] Paul J. Leach, Michael Mealling, and Rich Salz. A Universally Unique Identifier (UUID) URN namespace. Request for Comments 4122, Internet Engineering Task Force, Network Working Group, July 2005. Category: Standards Track. Available from <http://www.ietf.org/rfc/rfc4122.txt>.
- [10] PDF/A Competence Center, Berlin, Germany. *TechNote 0008: Predefined 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		Photoshop schema	1
General: Initial version	1	Made the package compatible with <code>ngerman</code> . Thanks to Tobias Mueller for the bug report.	11
v1.1			
<code>\hyxmp@construct@packet:</code>			
Explicitly set the category codes of characters <code><EF></code> , <code><BB></code> , and <code><BF></code> to “letter”. Thanks to Daniel Schömer for the bug report	47		
		v1.3	
		<code>\hyxmp@reencode:</code> Introduced this macro to re-encode Unicode strings as 8-bit strings before manipulating them into XMP schema. This change addresses a bug reported by Martin Münch	27
v1.2			
General: Added support for the <code>X₁TeX</code> backend (<code>xdvipdfmx</code>)	1		
Added support for the			

General: Introduced the <code>pdfmetalang</code> package option, which enables an author to specify the language in which he wrote the document's metadata	18	<code>\hyxmp@photoshop@schema:</code> Simplified using <code>\hyxmp@add@simple</code>	43
v1.4		<code>\hyxmp@reencode:</code> Replaced with an empty macro by Heiko Oberdiek	27
<code>\hyxmp@mm@schema:</code> Renamed the <code>xapMM</code> namespace prefix to <code>xmpMM</code>	41	<code>\hyxmp@skiptorelax:</code> Added by Heiko Oberdiek	31
<code>\hyxmp@rdf@dc:</code> Included metadata in the <code>x-default</code> language regardless of the specified metadata language	38	<code>\hyxmp@skipzeros:</code> Added by Heiko Oberdiek	30
<code>\hyxmp@xmpRights@schema:</code> Renamed the <code>xapRights</code> namespace prefix to <code>xmpRights</code>	40	<code>\hyxmp@string:</code> Added this macro	37
v1.5		<code>\hyxmp@toxml:</code> Added by Heiko Oberdiek	28
General: Made the XMP inclusion more robust. Thanks to Heiko Oberdiek for the bug report and suggested modifications.	11	Escaped parentheses written with <code>pdfmarks</code> to prevent dvips from line-wrapping the XMP packet	29
v2.0		<code>\hyxmp@toxml@unicodetex:</code> Added by Heiko Oberdiek	29
<code>\ProcessKeyvalOptions:</code> Added this macro	17	<code>\hyxmp@xetex@crap:</code> Added by Heiko Oberdiek	30
<code>\XMPTruncateList:</code> Added this macro	21	<code>\hyxmp@xmllify:</code> Completely rewritten by Heiko Oberdiek to better support Unicode-enabled TeX programs	27
<code>\hyxmp@ProcessKeyvalOptions:</code> Added this macro	17	<code>\hyxmp@xmp@basic@schema:</code> Added this macro	42
<code>\hyxmp@SpaceOther:</code> Added by Heiko Oberdiek	30	<code>\hyxmp@xmpRights@schema:</code> Modified to include <code>xmpRights:Marked</code> only when <code>pdfcopyright</code> is specified and <code>xmpRights:WebStatement</code> only when <code>pdflicenseurl</code> is specified	40
<code>\hyxmp@add@to+xml:</code> Updated also to replace commas	35	<code>\hyxmp@zero:</code> Added by Heiko Oberdiek	32
<code>\hyxmp@bom:</code> Added by Heiko Oberdiek	47	<code>\ifhyxmp@unicodetex:</code> Added by Heiko Oberdiek	26
<code>\hyxmp@comma:</code> Added this macro	20	<code>\xmpcomma:</code> Added this macro	20
<code>\hyxmp@construct@packet:</code> Modified by Heiko Oberdiek to use an appropriate BOM representation via <code>\hyxmp@bom</code>	47	<code>\xmpquote:</code> Added this macro	20
<code>\hyxmp@crap@convert:</code> Added by Heiko Oberdiek	31	General: Added support for the XMP Basic schema and miscellaneous other bits of metadata	1
<code>\hyxmp@crap@test:</code> Added by Heiko Oberdiek	31	Heiko Oberdiek's major rewrite of the code to better support native-Unicode TeX implementations (XeTeX and LuaTeX)	1
<code>\hyxmp@dc@schema:</code> Added support for <code>dc:language</code> and <code>dc:source</code>	39	New <code>\AtBeginDocument</code> code from Heiko Oberdiek to	
<code>\hyxmp@is@unicode:</code> Added by Heiko Oberdiek	28		
<code>\hyxmp@list@to+xml:</code> Modified by Heiko Oberdiek to use the new Unicode-processing macros	38		

properly encode		version 4 (random or pseudorandom) UUID	34
\@pdfmetalang	18	\hyxmp@dc@schema: Made dc:language a Bag instead of an individual item so as to conform to the latest XMP specifications, a detail identified by Florian Breitwieser	39
v2.1		\hyxmp@parse@time: Added this macro	22
\hypersetup: Added this macro	17	\hyxmp@parse@tz: Added this macro	22
\hyxmp@hypersetup: Added this macro	17	\hyxmp@parse@tz@char: Added this macro	22
\hyxmp@redefine@Hyp: Added this macro	15	\hyxmp@pdf@schema: Made \hyxmp@pdf@schema <i>always</i> include the Adobe PDF schema, even when empty. Florian Breitwieser noted that this is necessary for PDF/A-1b compliance	36
General: Enabled hyperxmp and hyperref to be loaded in either order. This addresses a bug report by Yury Donskoy	15	\hyxmp@pdf@to@xmp@date: Added this macro	22
v2.2		\hyxmp@pdfa@id@schema: Added this macro	47
\hyxmp@iptc@extensions: Added this macro to support PDF/A generation	45	\hyxmp@today: Modified the code to parse the time and timezone from \pdfcreationdate, as proposed by Florian Breitwieser	25
\hyxmp@list@to@lines: Added this macro	43	\hyxmp@today@define: Added this macro	24
\hyxmp@photometa@schema: Added this macro	44	\hyxmp@xmp@to@pdf@date: Added this macro	23
\hyxmp@text@resource: Added this macro	46	\xmptilde: Added this macro	21
\xmpcomma: Changed the default from \relax to an ordinary comma	20	General: Added support for the PDF/A Identification schema, as requested by Florian Breitwieser	1
\xmplinesep: Added this macro	43	v2.5	
General: Added support for the IPTC Photo Metadata schema	1	\hyxmp@add@to+xml: Updated also to replace underscores and to modify only the text being added, not the already-modified text	35
v2.3		\hyxmp@textunderscore: Added this macro	12
\hyxmp@iptc@extensions: Gave the lptc4xmpCore:CreatorContactInfo fields a unique pdfaType:prefix to better support conversion of the document to PDF/A	45	\hyxmp@uscore: Added this macro	20
v2.3a		General: Enabled “_” to work within email addresses, as requested by Leonid Sinev	1
General: Bug fix: Redefine \@pdflang as \@empty when hyperref has set it to \relax	18		
v2.3b			
\XMPTruncateList: Made all definitions local to avoid spurious Too many unprocessed floats errors when running with memoir	21		
v2.4			
\hyxmp@add@simple@var: Added this macro	37		
\hyxmp@create@uuid: Modified this macro to produce a proper			

v2.6	General: Added support for a new <code>pdfdate</code> key to explicitly specify the document date (and optionally time)	1	Added this macro	50
v2.7	General: Automatically use <code>\title</code> and <code>\author</code> if <code>pdftitle</code> and <code>pdfauthor</code> are left unspecified. Thanks to Maciej Radziejewski for the suggestion	18	<code>\hyxmp@today@define</code> : Modified to accept the name of a macro to define	24
v2.8	<code>\hyxmp@add@to+xml</code> : Corrected inadvertent lowercasing of non-Latin characters when run under \TeX or \LaTeX (bug reported by Leonid Sinev)	35	<code>\hyxmp@xmp@basic@schema</code> : Made the XMP <code>CreateDate</code> , <code>ModifyDate</code> , and <code>MetadataDate</code> match the PDF <code>CreationDate</code>	42
v2.9	<code>\hyxmp@pdfa@id@schema</code> : Let the author specify the PDF/A part and conformance IDs, as requested by Leonid Sinev	47	General: Made the code compatible with \LaTeX 0.85. Thanks to Robert Schlicht, Leonid Sinev, and David Carlisle for bug reports and to Leonid Sinev for helping test the new <code>hyperxmp</code> code	1
	<code>\hyxmp@photometa@schema</code> : Use <code>lptc4xmpCore</code> instead of <code>lptc4ContInfo</code> as the contact-information metadata prefix. Leonid Sinev reports that Acrobat’s PDF/A validator seems to prefer <code>lptc4xmpCore</code>	44	<code>\hyxmp@embed@packet@luatex</code> : Updated to use <code>\pdfextension</code> <code>obj uncompressed</code> as suggested by Hans Hagen	50
	General: Force inclusion of <code>dc:creator</code> , <code>dc:title</code> , and <code>dc:description</code> —even if empty—when <code>hyperref</code> is loaded with the <code>pdfa</code> option (suggested by Leonid Sinev)	1	<code>\hyxmp@embed@packet@pdftex</code> : Leave the XMP packet—and only the XMP packet—uncompressed in both \pdfTeX and pre-0.85 \LaTeX	49
	Introduced the <code>pdftype</code> package option, which enables an author to specify the type of document being produced	1	v3.2	
v3.0	<code>\hyxmp@embed@packet@luatex</code> :		<code>\hyxmp@as@pdf@date</code> : Added this macro	22
			<code>\hyxmp@as@xmp@date</code> : Added this macro	21
			<code>\hyxmp@today@define</code> : Modified to include hours and minutes	24
			<code>\hyxmp@xmp@basic@schema</code> : Honor <code>hyperref</code> ’s <code>pdfcreationdate</code> and <code>pdfmoddate</code> options plus a new <code>pdfmetadate</code> option. Leonid Sinev requested this additional control and helped test the resulting <code>hyperxmp</code> code	42

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		
\#	702, 940	\@pdfcreationdate 148, 892, 896
\&	452, 484, 939	\@pdfcreator 914
\,	695	\@pdfdatetime
\@author	189, 191	. 21, 149, 178, 180
\@baseurl	135, 915	\@pdfkeywords . 111, 150
\@elt . 236, 799, 952, 957		\@pdflang . . 151, 169,
\@elt@first	950	170, 172, 175, 826
\@elt@rest	952, 954	\@pdflicenseurl
\@firstofone	531	. 47, 152, 835, 855
\@firstoftwo	443	\@pdfmetadatetitle
\@gobble	241, 528	. 32, 153, 909, 912
\@gobbletwo 315, 330, 332		\@pdfmetalang 53, 173,
\@hyxmp@gobbletwo	315, 328	175, 177, 771, 774
\@pdfaformance	57, 1074	\@pdfmoddate 154, 902, 906
\@pdfapart	55, 1073	\@pdfsubject . . 155, 821
\@pdfauthor	90, 136, 188, 670, 676	\@pdftitle 156,
\@pdfauthortitle	49, 137, 921, 929	182, 670, 676, 820
\@pdfcaptionwriter	51, 138, 921, 930	\@pdftype . . 45, 157, 827
\@pdfcontactaddress	59, 139, 967, 985	\@secondoftwo 445
\@pdfcontactcity	67, 140, 968, 986	\@tempswafalse 757, 785
\@pdfcontactcountry	73, 141, 971, 989	\@tempswatrue
\@pdfcontactemail	77, 142, 973, 993	. 760, 763, 788, 791
\@pdfcontactphone	75, 143, 972, 992	\@title 183, 185
\@pdfcontactpostcode	71, 144, 970, 988	\^ 224, 228, 388,
\@pdfcontactregion	69, 145, 969, 987	695, 696, 1087–1089
\@pdfcontacturl	79, 146, 974, 994	_ 694, 696
\@pdfcopyright . 43, 147, 822, 839, 849		\~ 233, 522, 523
		A
		\and 90
		ASCII 12, 28
		\AtBeginDocument . . . 167
		\AtEndDocument 5
		\AtEndDvi 8
		atenddvi 11
		Author 7, 21
		B
		Bag 60
		C
		baseurl (option)
	 3, 11, 12, 42
		BOM 47, 59
		D
		Date 25
		\day 348, 349, 351
		dc:creator . . . 2, 7, 39, 61
		dc:date 2, 39
		dc:description . . 3, 39, 61
		dc:format 2
		dc:language . . 2, 39, 59, 60
		dc:rights 2, 39
		dc:source 2, 39, 59
		dc:subject 2, 39
		dc:title 3, 39, 61
		dc:type 2
		\define@key 22,
		33, 44, 46, 48, 50,
		52, 54, 56, 58, 60,
		68, 70, 72, 74,
		76, 78, 80, 90, 111
		dvipdf (option) 50
		dvipdfm 51
		dvips (option) 50
		dvips 7, 29, 59
		dvipsone (option) 50
		dviwindo (option) 50
		E
		\EdefEscapeHex 411, 424
		\EdefUnescapeHex . . . 428

\EdefUnescapeString 398	\hyxmp@as@pdf@date . 275	\hyxmp@embed@packet@luatex
ETX 20, 27	\hyxmp@as@xmp@date 1124, 1161
	27, 38, 247, 896, 906	\hyxmp@embed@packet@pdfmark
G	\hyxmp@at@end ... 3, 194 1140, 1168
Ghostscript 7	\hyxmp@big@prime ...	\hyxmp@embed@packet@pdftex
	. 592, 595, 605, 615 1120, 1148
H	\hyxmp@big@prime@ii	\hyxmp@embed@packet@xetex
\Hy@driver 592, 614 1132, 1236
. 4, 1119, 1123,	\hyxmp@bom .. 1080, 1095	\hyxmp@find@metadata
1127, 1131, 1136	\hyxmp@comma 133, 195
\Hy@unicodedefalse . 24, 35	. 61, 91, 112, 223	\hyxmp@first@char .. 245
hyperref 1, 3–	\hyxmp@commas@to@list	\hyxmp@first@char@i
7, 11, 12, 15, 17–	. 207, 238, 800, 959 245, 248, 276
19, 36, 48–50, 60, 61	\hyxmp@commas@to@list@i	\hyxmp@gobbletwo ... 315
\hypersetup 128, 185, 191 209, 211	\hyxmp@hash 701,
hyperxmp 1–	\hyxmp@concat@metadata	1008–1011, 1099
15, 18–21, 27, 32, 133	\hyxmp@Hyp@pdfauthor 84
36, 49, 52, 53, 60, 61	\hyxmp@construct@packet	\hyxmp@Hyp@pdfkeywords
\hyxmp@is@unicode . 432 1093, 1117 105
\hyxmp@add@simple ..	\hyxmp@count@non@spaces	\hyxmp@hypersetup .. 128
..... 728, 731, 1217, 1228	\hyxmp@InstanceID ..
737, 828, 855,	\hyxmp@count@spaces 675, 868
890, 893, 895, 1216, 1219	\hyxmp@iptc@extensions
900, 903, 905,	\hyxmp@crap@convert 978, 1004
910, 912, 914, 514, 548	\hyxmp@is@unicode ..
915, 929, 930,	\hyxmp@crap@result 400, 417, 432
986–989, 1073, 1074 504, 540	\hyxmp@legal 834
\hyxmp@add@simple@var	\hyxmp@crap@test 511, 536	\hyxmp@list
.... 723, 724, 747	\hyxmp@create@uuid .	. 800, 806, 959, 960
\hyxmp@add@to@xml 645, 673, 679	\hyxmp@list@to@lines
..... 681, 707,	\hyxmp@createdate ..	. 943, 985, 992–994
719, 733, 742, 872, 890,	\hyxmp@list@to@xml .
751, 767, 773,	893, 900, 903, 910 784, 823–827
777, 794, 802,	\hyxmp@dc@schema ...	\hyxmp@mm@schema ...
808, 815, 829, 814, 1103 861, 1108
845, 851, 856,	\hyxmp@def@DocumentID	\hyxmp@modulo@a 586,
864, 884, 916, 669, 862	605, 615, 621, 656
924, 933, 947,	\hyxmp@def@InstanceID	\hyxmp@new@xml 697, 698
951, 955, 961, 675, 863	\hyxmp@num 548
979, 998, 1005,	\hyxmp@define@createdate	\hyxmp@one@token ...
1046, 1059, 1069, 872, 888	594, 598, 1220,
1075, 1095, 1109	\hyxmp@DocumentID ..	1221, 1229, 1230
\hyxmp@and 90 669, 867	\hyxmp@padding 705, 1112
\hyxmp@append@hex ..	\hyxmp@dq@code . 1, 1249	\hyxmp@parse@time ..
. 618, 637–639, 643	\hyxmp@driver .. 3, 1116 256, 258
\hyxmp@append@hex@iii	\hyxmp@embed@packet	\hyxmp@parse@tz ...
. 636, 642, 652, 663 196, 1116 265, 268, 272
\hyxmp@append@hex@iv	\hyxmp@embed@packet@dvipdfm	\hyxmp@parse@tz@char
..... 641, 647, 1128, 1198 260, 262
648, 650, 665–667		

\hyxmp@pdf@schema	718, 1101	\hyxmp@string@len	1199, 1214	\hyxmp@xmp@to@pdf@date@ii	288, 291
\hyxmp@pdf@to@xmp@date	249, 254, 372, 375, 877, 880	\hyxmp@sublist	212, 213, 216, 217	\hyxmp@xmp@to@pdf@date@iii	294, 297
\hyxmp@pdfa@id@schema	1067, 1107	\hyxmp@temp@list	236	\hyxmp@xmp@to@pdf@date@iv	300, 303
\hyxmp@pdfauthor	81, 90, 823	\hyxmp@temp@str	236	\hyxmp@xmp@to@pdf@date@v	306, 309
\hyxmp@pdfkeywords	81, 111, 824	\hyxmp@text	396, 474, 504, 548	\hyxmp@xmp@to@pdf@date@vi	312, 316
\hyxmp@pdfstringdef	15, 26, 37, 44, 46, 48, 50, 52, 54, 56, 58, 63, 68, 70, 72, 74, 76, 78, 80	\hyxmp@text@resource	1038–1045, 1058	\hyxmp@xmp@to@pdf@date@vii	319, 322, 332
\hyxmp@photometa@data	965	\hyxmp@text@underscore	15	\hyxmp@xmp@to@pdf@date@viii	335, 338
\hyxmp@photometa@schema	965, 1105	\hyxmp@today	180, 368, 676, 825	\hyxmp@xmpRights@schema	833, 1102
\hyxmp@photoshop@data	920	\hyxmp@today@define	341, 370, 875	\hyxmp@zero	557, 564, 571, 577, 582
\hyxmp@photoshop@schema	920, 1104	\hyxmp@toxml	426, 449		
\hyxmp@ProcessKeyvalOptions	123	\hyxmp@toxml@unicodetex	414, 474		
\hyxmp@rand@num	611, 620, 655, 672, 678	\hyxmp@trimb	381, 384		
\hyxmp@rdfsdc	756, 820–822	\hyxmp@trimc	384, 385		
\hyxmp@redefine@Hyp	83, 125, 130	\hyxmp@trimspaces	216, 377		
\hyxmp@reencode	393	\hyxmp@try	504		
\hyxmp@rights	834, 837, 841, 843	\hyxmp@unicodetextfalse	387		
\hyxmp@seed@rng	594, 671, 677	\hyxmp@unicodetexttrue	387		
\hyxmp@seed@rng@i	596, 598	\hyxmp@uscore	17, 227		
\hyxmp@seed@string	670, 671, 676, 677	\hyxmp@x@default	173, 717, 771, 778		
\hyxmp@set@rand@num	611, 619, 654	\hyxmp@xetex@crap	405, 504		
\hyxmp@skiptorelax	541, 547	\hyxmp+xml	698, 705, 1093, 1157, 1165, 1188, 1199, 1206, 1237		
\hyxmp@skipzeros	499	\hyxmp@xmlified	396, 743, 752, 774, 778, 800, 959		
\hyxmp@SpaceOther	508, 521	\hyxmp@xmlify	177, 396, 741, 750, 766, 799, 958		
\hyxmp@string	737	\hyxmp@xmp@basic@schema	883, 1106		
		\hyxmp@xmp@to@pdf@date	279, 282		
		\hyxmp@xmp@to@pdf@date@i	283, 285		
				I	
				IETF	5
				\if@tempwa	765, 793
				\iffalse	756, 784
				\ifHy@pdfa	820, 821, 823, 1068
				\ifhyxmp@unicodetex	387, 399, 684, 1081
				ifluatex	49
				\ifluatex	1150, 1154
				ifxetex	12
				\ifxetex	404
				Info	7
				intcalc	12
				\intcalcDiv	553, 560, 567
				\intcalcMod	555, 562, 569
				IPTC	8, 14, 35, 44, 45, 60
				lptc4xmpCore:CreatorContactInfo	2, 44, 60
				ISO	13
				J	
				\jobname	161, 200, 670, 676, 828, 1137
				K	
				Keywords	7, 36
				\KV@Hyp@pdfauthor	90
				\KV@Hyp@pdfkeywords	111
				kvoptions	12, 17

L		
LF	43	
Lua \LaTeX	7, 9,	
	10, 25, 37, 41, 42, 61	
Lua \TeX	26,	
	29, 49, 50, 52, 59, 61	
M		
memoir	60	
Metadata	7, 48, 52	
MetadataDate	41, 61	
ModifyDate	41, 61	
\month	343, 344, 346	
N		
NAK	12, 20, 27	
nativepdf (option)	50	
\newif	387	
\next	25, 36, 211, 598	
ngerman	11, 58	
\number	551, 553, 555,	
	560, 562, 567, 569	
\numexpr	1166	
O		
options		
baseurl	3, 11, 12, 42	
dvipdf	50	
dvips	50	
dvipsone	50	
dviwindo	50	
nativepdf	50	
pdfa	4, 61	
pdfaconformance	4, 47	
pdfapart	4, 47	
pdfauthor	3, 10–	
	12, 15, 16, 18, 39, 61	
pdfauthortitle	4, 5, 11	
pdfcaptionwriter	4, 5	
pdfcontactaddress		
	4, 5, 8, 9	
pdfcontactcity	4, 5	
pdfcontactcountry	4, 5	
pdfcontactemail	4, 5	
pdfcontactphone	4, 5	
pdfcontactpostcode		
	4, 5	
pdfcontactregion	4, 5	
pdfcontacturl	4, 5, 11	
pdfcopyright		
	4, 5, 39, 40, 59	
pdfcreationdate	3, 5, 61	
pdfdate	4, 5, 11, 12, 61	
pdfkeywords		
	3, 10, 12, 15, 39	
pdflang		
	3, 5, 11, 12, 18, 39	
pdflicenseurl		
	4, 5, 11, 40, 59	
pdfmark	50	
pdfmetadate	4, 5, 13, 61	
pdfmetalang	4, 5, 11, 59	
pdfmoddate	4, 5, 61	
pdfproducer	4, 12	
pdfsubject	4, 12, 39	
pdftitle	4,	
	11, 12, 18, 39, 61	
pdftype	4–6, 61	
ps2pdf	50	
textures	50	
unicode	11	
vtexpdfmark	50	
\pdfcompresslevel	1152	
pdfcontactaddress (op-		
tion)	4, 5, 8, 9	
pdfcontactcity (option)		
	4, 5	
pdfcontactcountry (op-		
tion)	4, 5	
pdfcontactemail (op-		
tion)	4, 5	
pdfcontactphone (op-		
tion)	4, 5	
pdfcontactpostcode (op-		
tion)	4, 5	
pdfcontactregion (op-		
tion)	4, 5	
pdfcontacturl (option)		
	4, 5, 11	
pdfcopyright (option)		
	4, 5, 39, 40, 59	
pdfcreationdate (option)		
	3, 5, 61	
\pdfcreationdate	372, 877	
pdfdate (option)		
	4, 5, 11, 12, 61	
PDFDocEncoding		
	15, 27, 28	
pdfescape	12	
\pdfextension	1162, 1166	
\pdffeedback		
	375, 880, 1166	
pdfkeywords (option)		
	3, 10, 12, 15, 39	
pdflang (option)		
	3, 5, 11, 12, 18, 39	
\pdflastobj	1158	
pdf \LaTeX	3, 7, 9, 25, 37, 41	
pdflicenseurl (option)		
	4, 5, 11, 40, 59	
pdfmark (option)	50	
\pdfmark	1169,	
	1172, 1176,	
	1186, 1190, 1194	
pdfmetadate (option)		
	4, 5, 13, 61	
pdfmetalang (option)		
	4, 5, 11, 59	
\pdfminorversion	728	
pdfmoddate (option)		
	4, 5, 61	
\pdfobj	1154	
P		
\PackageWarningNoLine		
	160, 199, 1135	
PDF	1–3, 5,	
	7–10, 12, 13, 19,	
	21–23, 29, 34–36,	
	41, 48, 49, 52, 60, 61	
PDF/A	3, 4, 14,	
	35, 36, 45, 47, 60, 61	
pdf:Keywords	2, 36	
pdf:PDFVersion	3, 36	
pdf:Producer	3, 36	
pdfa (option)	4, 61	
pdfaconformance (op-		
tion)	4, 47	
pdfaid:conformance	3	
pdfaid:part	3	
pdfapart (option)	4, 47	
pdfaType:prefix	60	
pdfauthor (option)	3, 10–	
	12, 15, 16, 18, 39, 61	
pdfauthortitle (option)		
	4, 5, 11	
pdfcaptionwriter (op-		
tion)	4, 5	
\pdfcatalog	1158	

pdfproducer (option)	4, 12	stringenc	12	X _q LaTeX	7,
\pdfstringdef	18	\StringEncodingConvert			10, 25, 37, 41, 61
pdfsubject (option)	4, 12, 39		401, 407, 418, 421, 516	X _q TeX	12,
pdfTeX	11, 29, 49, 52, 61	Subject	7		26, 29, 30, 52, 58, 59
pdftitle (option)	4, 11, 12, 18, 39, 61			XML	1, 2, 8,
pdftype (option)	4–6, 61	T			19, 26, 28, 29, 35,
\pdfvariable	731	TeX	26, 29, 32, 41, 51, 52, 59		36, 38, 39, 43, 45, 48
photoshop:AuthorsPosition	3, 43	Text	47	XMP	1–3,
photoshop:CaptionWriter	3, 43	\textunderscore			5–13, 18–23, 25,
PI	35		16, 17, 19		29, 32, 35–37, 40–
\ProcessKeyvalOptions	123	textures (option)	50	xmp:BaseURL	2
Producer	36	\time	353, 361	xmp:CreateDate	2
ps2pdf (option)	50	Title	7	xmp:CreatorTool	3
				xmp:MetadataDate	2
Q		U		xmp:ModifyDate	2
\Q	377, 386	Unicode	11, 12, 26–31, 39, 44, 47, 52, 59	\xmpcomma	
R		unicode (option)	11		61, 64, <u>90</u> , <u>111</u> , <u>222</u>
RDF	38	URL	2, 5, 11, 13, 15, 40–43, 45	xmpincl	3
rdf:li	2	UTF-16BE	28	\xmplinesep	<u>938</u> , <u>955</u> , <u>990</u>
rdf:Seq	2	UTF-32BE	27, 28	xmpMM:DocumentID	2, 32, 41
\renewcommand	124	UTF-8	28	xmpMM:InstanceID	2, 32, 41
\RequirePackage	7, 10–14, 1147	UUID	32, 34, 35, 60	\xmpquote	62, 65, <u>90</u> , <u>111</u> , <u>231</u>
S		V		xmpRights:Marked	2, 40, 59
\SE->pdfdoc@03	<u>394</u>	\vfuzz	385	xmpRights:WebStatement	2, 40, 59
\SE->pdfdoc@15	<u>395</u>	vtexpdfmark (option)	50	\xmptilde	<u>232</u>
\special	1200, 1208, 1237, 1243	X		\XMPTruncateList	<u>236</u>
		\x	<u>504</u>	Y	
		xdvipdfmx	10, 52	\year	342