

The hyperxmp package^{*}

Scott Pakin
scott+hyxmp@pakin.org

March 31, 2019

Abstract

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

1 Introduction

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

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

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

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

^{*}This document corresponds to `hyperxmp` v4.0, dated 2019/03/31.

```

    </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:

- address of primary author (`lptc4xmpCore:CreatorContactInfo.CiAdrExtadr`, `lptc4xmpCore:CreatorContactInfo.CiAdrCity`, `lptc4xmpCore:CreatorContactInfo.CiAdrRegion`, `lptc4xmpCore:CreatorContactInfo.CiAdrPcode`, and `lptc4xmpCore:CreatorContactInfo.CiAdrCtry`)
- author(s) (`dc:creator`)
- base URL for relative references (`xmp:BaseURL`)
- book edition (`prism:bookEdition`)
- copyright (`dc:rights` and `xmpRights:Marked`)
- date (`dc:date`, `xmp:CreateDate`, `xmp:ModifyDate`, and `xmp:MetadataDate`)
- DOI (`prism:doi`)
- email address(es) of primary author (`lptc4xmpCore:CreatorContactInfo.CiEmailWork`)
- file format (`dc:format`)
- file name of main L^AT_EX source file (`dc:source`)
- file size in bytes (`prism:byteCount`)
- ISBN (`prism:isbn`)
- ISSN—both print (`prism:issn`) and electronic (`prism:elssn`)
- issue number of parent publication (`prism:number`)
- keywords (`pdf:Keywords` and `dc:subject`)

- language used (dc:language)
- license URL (xmpRights:WebStatement)
- metadata writer (photoshop:CaptionWriter)
- page count (prism:pageCount)
- page range(s) (prism:pageRange)
- PDF version (pdf:PDFVersion)
- PDF-generating tool (pdf:Producer and xmp:CreatorTool)
- PDF/A compliance level and version (pdfaid:part and pdfaid:conformance)
- position/title of primary author (photoshop:AuthorsPosition)
- publication name of parent publication (prism:publicationName)
- summary (dc:description)
- subtitle (prism:subtitle)
- telephone number(s) of primary author
(Iptc4xmpCore:CreatorContactInfo.CiTelWork)
- title (dc:title)
- type of document (dc:type)
- type of parent publication (prism:aggregationType)
- URL of the document (prism:url)
- URL(s) of the primary author (Iptc4xmpCore:CreatorContactInfo.CiUrlWork)
- UUID for the document (xmpMM:DocumentID)
- UUID for the document instance (xmpMM:InstanceID)
- version identifier for the document (xmpMM:VersionID)
- volume number of parent publication (prism:volume)

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 pdfL^AT_EX and only when in PDF-generating mode. `hyperxmp` additionally works with a few other PDF-producing L^AT_EX 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`
- `pdfkeywords`
- `pdfproducer`
- `pdfauthor`
- `pdflang`
- `pdfsubject`
- `pdfcreationdate`
- `pdfmoddate`
- `pdftitle`

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

- `pdfaformance`
- `pdfcontactemail`
- `pdfeissn`
- `pdfapart`
- `pdfcontactphone`
- `pdfinstanceid`
- `pdfauthortitle`
- `pdfcontactpostcode`
- `pdfisbn`
- `pdfbookedition`
- `pdfcontactregion`
- `pdfissn`
- `pdfbytes`
- `pdfcontacturl`
- `pdfissuenum`
- `pdfcaptionwriter`
- `pdfcopyright`
- `pdflicenseurl`
- `pdfcontactaddress`
- `pdfdate`
- `pdfmetadate`
- `pdfcontactcity`
- `pdfdocumentid`
- `pdfmetalang`
- `pdfcontactcountry`
- `pdfdoi`
- `pdfpagecount`

- `pdfpagerange`
- `pdfsource`
- `pdfurl`
- `pdfpublication`
- `pdfsubtitle`
- `pdfversionid`
- `pdfpubtype`
- `pdftype`
- `pdfvolumenum`

2.1 Option descriptions

`pdftitle` The document title is specified as normal for `hyperref` with `pdftitle`, but see Note 7 on page 14 for instructions on how to specify a title in multiple languages. `hyperxmp` introduces a complementary `pdfsubtitle` option:

```
pdftitle={Frankenstein},
pdfsubtitle={The Modern Prometheus},
```

Unfortunately, the subtitle can appear in only one language. It assumed to be the same language as the document language (`pdflang`) but can be overridden by preceding the text with a bracketed ISO 639-1 two-letter language code and an optional ISO 3166-1 two-letter region code. See the example below for `pdfpublication`.

`pdfauthortitle` `pdfauthortitle` indicates the primary author’s position or title. `pdfcaptionwriter` specifies the name of the person who added the metadata to the document. The next eight items describe how to contact the person or institution responsible for the document (the “contact”). `pdfcontactaddress` is the contact’s street address and can include the institution name if the contact is an institution; `pdfcontactcity` is the contact’s city; `pdfcontactcountry` is the contact’s country; `pdfcontactemail` is the contact’s email address (or multiple, comma-separated email addresses); `pdfcontactphone` is the contact’s telephone number (or multiple, comma-separated telephone numbers); `pdfcontactpostcode` is the contact’s postal code; `pdfcontactregion` is the contact’s state or province; and `pdfcontacturl` is the contact’s URL (or multiple, comma-separated URLs).

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

`pdfmetalang` `pdfmetalang` indicates the natural language in which certain metadata—specifically, the document’s title, subject, and copyright statement—are written. The language should be specified using an IETF language tag [10], for example, “en” for English, “en-US” for specifically United States English, “de” for German, and so forth. If `pdfmetalang` is not specified, `hyperxmp` assumes the metadata language is the same as the document language (`hyperref`’s `pdflang` option). If neither `pdfmetalang` nor `pdflang` is specified, `hyperxmp` uses only “x-default” as the metadata language. Note that “x-default” metadata are always included in addition to the specified metadata language, as the user reading the document may not have specified a language preference.

XMP can include a universally unique identifier (UUID) for each document and for each instance of a given document. By default, `hyperxmp` assigns a version 4 (i.e., pseudorandom) UUID [11] for each of these. How-

pdfdocumentid	ever, a document can alternatively specify a particular document identifier using <code>pdfdocumentid</code> and (not normally recommended) a particular instance identifier using <code>pdfinstanceid</code> . These should be of the form <code>uuid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx</code> , where “x” is a lowercase hexadecimal number. For example, <code>uuid:53ab7f19-a48c-5177-8bb2-403ad907f632</code> is a valid argument to <code>pdfdocumentid</code> (or <code>pdfinstanceid</code>). See Leach, Mealling, and Salz’s UUID specification document for details on how to produce the various forms of UUIDs [11]. A more freeform mechanism than <code>pdfinstanceid</code> for versioning documents is available via <code>pdfversionid</code> . The version specified by <code>pdfversionid</code> can be incremented as 1, 2, 3, ...; identified with a hierarchical numbering scheme (e.g., this document is versioned 4.0 to match the package version); or labeled using any other approach. One possibility is to use a revision number or commit hash from the version-control software maintaining the document. For example, the <code>\gitVer</code> macro from the <code>gitver</code> package is an expandable (see Note 8 on page 14) version of the current Git hash that can suitably be passed to <code>pdfversionid</code> .
pdfinstanceid	
pdfversionid	
pdfisbn	Already-published documents can be identified in a number of ways. <code>pdfisbn</code> specifies the ISBN. <code>pdfissn</code> refers to the ISSN of the <i>print</i> version of the document while <code>pdfeissn</code> refers to the ISSN of the <i>electronic</i> version of the document. <code>pdfdoi</code> specifies the DOI and should include only the DOI name without any URL prefix. For example, specify <code>pdfdoi={10.1145/3149526.3149532}</code> , <i>not</i> <code>pdfdoi={https://doi.org/10.1145/3149526.3149532}</code> . <code>pdfurl</code> points to the complete URL for the document. In contrast, <code>baseurl</code> points one level up and is used to resolve relative URLs.
pdfissn	
pdfeissn	
pdfdoi	
pdfurl	
baseurl	
pdfpublication	Already-published documents can further be identified by the publication in which they appear. <code>pdfpublication</code> specifies the title of the journal, magazine, or other parent document. The title language is assumed to be the same as the document language (<code>pdflang</code>) but can be overridden by preceding the text with a bracketed ISO 639-1 two-letter language code and an optional ISO 3166-1 two-letter region code. For example, <code>pdfpublication={[fr]Charlie Hedbo}</code> indicates a French-language title. Were the language or pronunciation differences significant, <code>fr-FR</code> would indicate specifically the French spoken in France, as opposed to that spoken in, say, Canada (<code>fr-CA</code>) or Belgium (<code>fr-BE</code>).
pdfpubtype	<code>pdfpubtype</code> indicates the type of publication in which the document was published. This should be one of the PRISM aggregation types [8] such as <code>book</code> , <code>journal</code> , <code>magazine</code> , <code>manual</code> , <code>report</code> , or <code>whitepaper</code> . For publications in journals, magazines, and similar periodicals, a document can specify the volume number with <code>pdfvolumenum</code> and the issue number within the volume with <code>pdfissuenum</code> . <code>pdfpagerange</code> indicates the page numbers at which the document appears within the publication. The intention is that this be a comma-separated list of dash-separated ranges, as in <code>pdfpagerange={1,4-5}</code> . See Note 9 on page 15 for advice on how to assign <code>pdfpagerange</code> semi-automatically. For books, <code>pdfbookedition</code> names the edition of the book. This is specified as text, not a number. As with <code>pdfpublication</code> (above), <code>pdfbookedition</code> accepts a bracketed language code, as in <code>pdfbookedition={[en]Second edition}</code> .
pdfvolumenum	
pdfissuenum	
pdfpagerange	
pdfbookedition	
pdfnumpages	The number of pages in the published, print version of the document can be expressed with <code>pdfnumpages</code> . Note 9 on page 15 explains how to automatically

assign a value to `pdfnumpages`.

<code>pdfdate</code>	<p>XMP metadata can include a number of dates (in fact, timestamps, as they include both date and time components). <code>pdfdate</code> specifies the document date. It is analogous to the \LaTeX <code>\date</code> command, and, like <code>\date</code>, 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 <code>YYYY-MM-DDThh:mm:ss+TT:tt</code>.¹ A W3C recommendation [14] 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 <code>2014-09-23T14:15:09-06:00</code>. This can be truncated (with loss of information) to <code>2014-09-23T14:15:09</code>, <code>2014-09-23T14:15</code>, <code>2014-09-23</code>, <code>2014-09</code>, or <code>2014</code> but no other subsets. PDF dates are written in the form <code>D:YYYYMMDDhhmmss+TT'tt'</code>. The same date in the preceding example would be written as <code>D:20140923141509-06'00'</code> in PDF format.</p>
<code>pdfcreationdate</code>	<p>The document's creation date, modification date, and metadata date are normally set automatically, but <code>pdfcreationdate</code>, <code>pdfmoddate</code>, and <code>pdfmetadate</code> can be used to override the defaults. Like <code>pdfdate</code>, <code>pdfmetadate</code> can be specified in either XMP or PDF format. However, because <code>hyperref</code> defines <code>pdfcreationdate</code> and <code>pdfmoddate</code> and expects these to be written as PDF dates, <code>hyperxmp</code> 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 <code>pdfcreationdate</code>, <code>pdfmoddate</code>, or <code>pdfmetadate</code>.</p>
<code>pdfmoddate</code>	
<code>pdfmetadate</code>	
<code>pdftype</code>	<p><code>pdftype</code> describes the type of document being produced. This refers to “the nature or genre of the resource” [4] such as <code>poem</code>, <code>novel</code> or <code>working paper</code>, as opposed to the file format (always <code>application/pdf</code> when generated by <code>hyperxmp</code>). Although <code>pdftype</code> 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 <code>Collection</code>, <code>Dataset</code>, <code>Event</code>, <code>Image</code>, <code>InteractiveResource</code>, <code>MovingImage</code>, <code>PhysicalObject</code>, <code>Service</code>, <code>Software</code>, <code>Sound</code>, <code>StillImage</code>, and <code>Text</code>. <code>pdftype</code> defaults to <code>Text</code>, which refers to “books, letters, dissertations, poems, newspapers, articles, archives of mailing lists,” [5] and other forms of text—all things \LaTeX is commonly used to typeset.</p>
<code>pdfbytes</code>	<p>The <code>pdfbytes</code> option expresses the document's file size in bytes. The intention is for this to be used to display an estimate of download time to a user or to serve as a quick check on whether a file was transmitted correctly between systems. This feature is easiest to use in conjunction with $\text{pdf}\text{\LaTeX}$'s <code>\pdffilesize</code> primitive: <code>“pdfbytes={\pdffilesize{\jobname.pdf}}”</code>. Note that this requires a second run of <code>pdftex</code> because it queries the size of the PDF file from the <i>previous</i> run.</p>
<code>pdfsource</code>	<p>A rarely needed option, <code>pdfsource</code>, overrides the name of the \LaTeX source file. It defaults to <code>\jobname.tex</code> but can be replaced by any other string. If <code>pdfsource</code> is given an empty argument, no document source will be specified at all.</p>
<code>pdfaconformance</code>	<p>Two other rarely needed options, <code>pdfaconformance</code> and <code>pdfapart</code>, are used in conjunction with <code>hyperref</code>'s <code>pdfa</code> option to claim a particular PDF/A standard by</p>
<code>pdfapart</code>	

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

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).

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.

2.2 A complete example

The following is a sample \LaTeX document that provides values for most of the metadata options that `hyperxmp` recognizes:

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage{hyperxmp}
\usepackage[unicode]{hyperref}

\title{%
  On a heuristic viewpoint concerning the production and
  transformation of light}
\author{Albert Einstein}
\date{March 17, 1905}

\hypersetup{%
  pdftitle={%
    On a heuristic viewpoint concerning the production and
    transformation of light},
  pdfsubtitle={[en-US]Putting that bum Maxwell in his place},
  pdfauthor={Albert Einstein},
  pdfauthortitle={\xmpquote{Technical Assistant\xmpcomma\ Level III}},
  pdfdate={1905-03-17},
  pdfcopyright={Copyright (C) 1905, Albert Einstein},
  pdfsubject={photoelectric effect},
  pdfkeywords={energy quanta, Hertz effect, quantum physics},
  pdflicenseurl={http://creativecommons.org/licenses/by-nc-nd/3.0/},
  pdfcaptionwriter={Scott Pakin},
  pdfcontactaddress={Kramgasse 49},
  pdfcontactcity={Bern},
  pdfcontactpostcode={3011},
  pdfcontactcountry={Switzerland},
  pdfcontactphone={031 312 00 91},
  pdfcontactemail={aeinstein@ipi.ch},
  pdfcontacturl={%
    http://einstein.biz/,
    https://www.facebook.com/AlbertEinstein
  },
  pdfdocumentid={uuid:6d1ac9ec-4ff2-515a-954b-648eeb4853b0},
  pdfversionid={2.998e8},
```



```

pdfpublication={[de]Annalen der Physik},
pdfpubtype={journal},
pdfvolumenum={322},
pdfissuenum={6},
pdfpagerange={132-148},
pdfnumpages={17},
pdfissn={0003-3804},
pdfeissn={1521-3889},
pdflang={en},
pdfmetalang={en},
pdfurl={http://www.physik.uni-augsburg.de/annalen/history/einstein-papers/1905_17_132-148.p},
pdfdoi={10.1002/andp.19053220607},
pdfbytes={\pdffilesize{\jobname.pdf}} % Requires pdflatex
}
\XMPLangAlt{de}{pdftitle={Über einen die Erzeugung und Verwandlung des
  Lichtes betreffenden heuristischen Gesichtspunkt}}

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

```

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

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

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

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

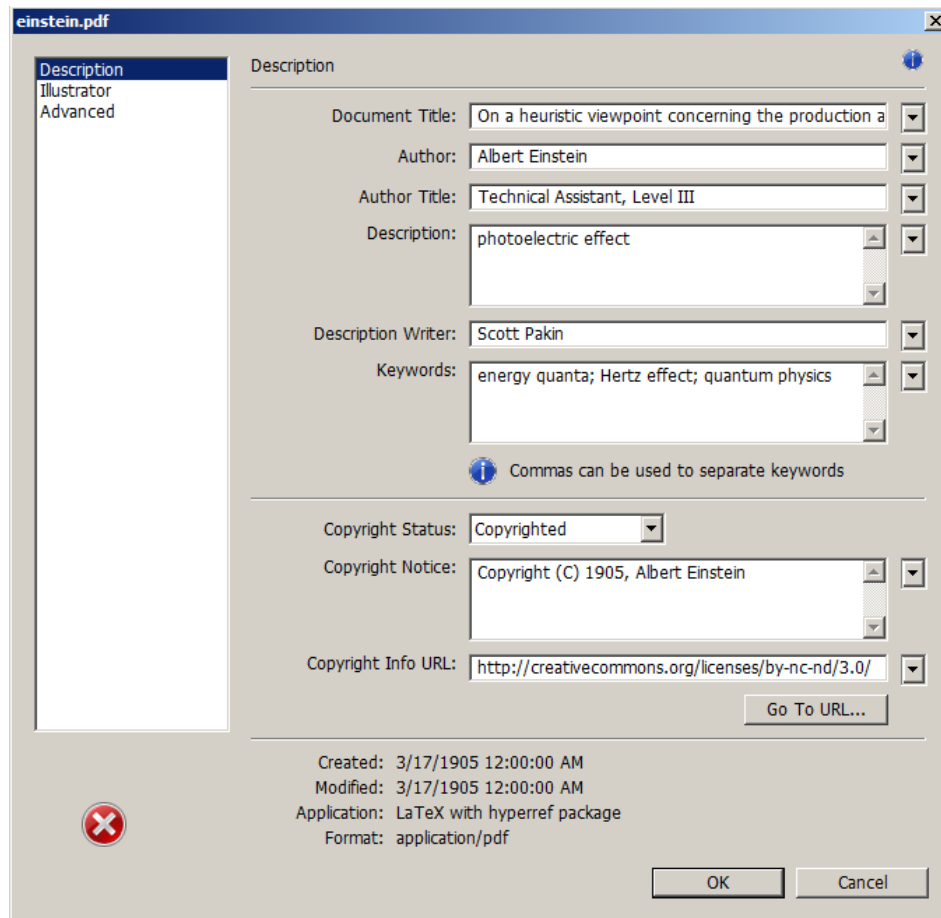


Figure 1: XMP metadata as it appears in Adobe Acrobat

2.3 Usage notes

Note 1: Conflicting metadata in PDF/A documents A PDF file includes an Info dictionary containing Author, Title, Subject, and Keywords keys. The hyperref package's pdfauthor, pdftitle, pdfsubject, and pdfkeywords options assign values to those keys. The hyperxmp package additionally uses those options to assign values to various XMP metadata: dc:creator, dc:title, dc:description, and pdf:Keywords. The PDF/A specification indicates that values that appear in both the PDF Info dictionary and XMP packet must match. The problem is that in XMP, the author and keywords can be proper lists, as in

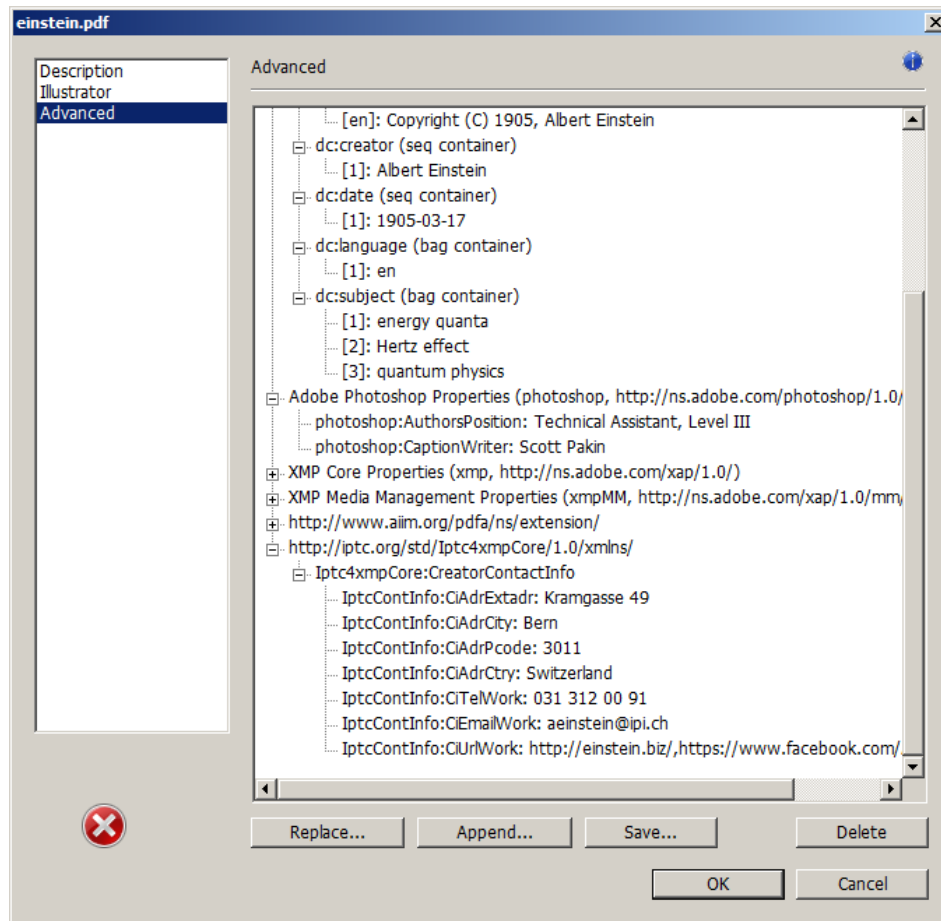


Figure 2: Additional XMP metadata as it appears in Adobe Acrobat

```
<dc:creator>
  <rdf:Seq>
    <rdf:li>Curly Howard</rdf:li>
    <rdf:li>Larry Fine</rdf:li>
    <rdf:li>Moe Howard</rdf:li>
  </rdf:Seq>
</dc:creator>
```

while in PDF, the author and keywords are specified as flat strings. Alas, there is no definition of how a list should be collapsed to a flat string: “Curly Howard, Larry Fine, Moe Howard” or “Curly Howard; Larry Fine; Moe Howard” or something else. I have not yet found a form of flat string that passes all PDF/A validators. Furthermore, when Adobe Acrobat—at least Adobe Acrobat DC (2019)

and earlier versions—converts a PDF file to PDF/A format, it does so by discarding all but the first author, which is an unsatisfying solution.

Starting with version 4.0, `hyperxmp`’s solution is to suppress writing metadata to the PDF `Info` dictionary and write it only to the XMP packet. This appears to pacify PDF/A validators yet retains the author and keyword lists in their non-truncated form. If desired, the `Info` dictionary can be retained by passing the `keeppdfinfo` option to `\hypersetup`.

Note 2: Acrobat multiline-field bug The IPTC Photo Metadata schema states that “the [contact] address is a multiline field” [9]. `hyperxmp` converts commas in `pdfcontactaddress`’s argument to line breaks in the generated XML. Unfortunately, a bug in Adobe Acrobat—at least in Adobe Acrobat DC (2019) and earlier versions—causes that PDF reader to discard line breaks in the contact address. Interestingly, Adobe Illustrator CS5 correctly displays the contact address. If you find Adobe Acrobat’s behavior bothersome, you can redefine the `\xmplinesep` macro as a string to use as an address-line separator. For example, the following replaces all commas appearing in `pdfcontactaddress`’s argument with semicolons:

`\xmplinesep`

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

Note 3: Object compression One intention of XMP is that metadata embedded in a file be readable even without knowledge of the file’s format. That is, the metadata are expected to appear as plain text. Although `hyperxmp` does its best to honor that intention, it faces a few challenges:

1. When run with versions of `LuaATEX` earlier than 0.85, `hyperxmp` leaves all PDF objects uncompressed. This is due to `LuaATEX` treating object compression as a global parameter, unlike `pdfATEX`, which treats it as a local parameter. Hence, when `hyperxmp` requests that the XMP packet be left uncompressed, `LuaATEX` 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 `LuaATEX` v0.85 onwards.
2. `XYATEX` (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., `LuaATEX`), (2) pass the `--output-driver="xdvipdfmx -z0"` option to `XYATEX` to instruct `xdvipdfmx` to turn off all compression (which will of course make the PDF file substantially larger), or (3) postprocess the generated PDF file by loading it into the commercial version of Adobe Acrobat and re-saving it with the Save As... menu option.

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

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

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

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

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

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

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

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

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

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

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

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

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

Note 7: Multilingual metadata The `pdfmetalang` option specifies the language in which the document’s metadata is written. It defaults to the value of `pdflang`, which specifies the document language. As of version 3.3 of `hyperxmp`, it is possible to include certain metadata—specifically, the document’s title, subject, and copyright statement—in more than one language. The `\XMPLangAlt` macro provides this functionality. Usage is as follows:

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

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

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

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

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

```
\usepackage{texdate}
\initcurrdate
```

```

\hypersetup{%
  pdfcopyright={Copyright \textcopyright\ \printdate{Y}, Scott Pakin}
}

```

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

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

Note 9: Automatic page counting Although `pdfnumpages` and `pdfpagerange` are intended to refer to pages in the final, published version of a document, it would be convenient for them to be generated automatically when producing a standalone PDF file that is not intended to be incorporated into a book, journal, or other publication (or if it is known that the pages will not be renumbered for publication). One approach is to use the `totpages` package to keep track of the number of pages.

```

\hypersetup{%
  pdfnumpages={\ref*{TotPages}}
}

```

`totpages` can likewise help generate `pdfpagerange`. For documents numbered from 1 to n , a simple

```

\hypersetup{%
  pdfpagerange={1-\ref*{TotPages}}
}

```

should suffice. A bit more effort is needed for documents that change numbering schemes, such as using lowercase Roman numerals for the front matter and Arabic numerals for the main matter and back matter. One approach is to use `\label` to mark the first and last page of each numbering scheme and specify `pdfpagerange` as in the following:

```

\hypersetup{%
  pdfpagerange={%
    \pageref*{page:begin-front}-\pageref*{page:end-front},%
    1-\pageref*{TotPages}}%
}

```

I don't know how unnumbered pages (e.g., blank pages and the title page) are supposed to be handled. I suppose blank pages can be omitted from `pdfpagerange`, and title page can be either omitted or listed as `title`, for example.

It appears that at least with version 2.00 of `totpages`, the `TotPages` label is not defined until after the `\begin{document}`. Consequently, using `TotPages` within a `\hypersetup` invocation in the document's preamble will produce “??” as the page count in the XMP packet. The solution is either to assign `pdfnumpages` and `pdfpagerange` after the `\begin{document}` or to ask \LaTeX to do that on your behalf:

```
\AtBeginDocument{%
  \hypersetup{%
    pdfnumpages={\ref*{TotPages}},
    pdfpagerange={1-\ref*{TotPages}}
  }%
}
```

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` `\hyxmp@driver` 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 additional \LaTeX run.

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


3.2 Integration with hyperref

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

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

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

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

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

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

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

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

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

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

```

28 \edef\next{%
29 \noexpand\hyxmp@pdfstringdef\noexpand\@pdfdatettime{%
30 \noexpand\hyxmp@as@xmp@date{#1}}%
31 }%
32 \next
33 \endgroup
34 }

```

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

```

35 \def\@pdfmetadatettime{}
36 \define@key{Hyp}{pdfmetadate}{%
37 \beginngroup
38 \Hy@unicodetfalse

```

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

```

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

```

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

```

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

```

`\@pdftype` Prepare to store the document's logical type, which defaults to "Text".

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

<code>\@pdfapart</code>	Prepare to store the PDF/A part ID, which defaults to “1”. 58 <code>\def\@pdfapart{1}</code> 59 <code>\define@key{Hyp}{pdfapart}{\hymp@pdfstringdef\@pdfapart{#1}}</code>
<code>\@pdfaconformance</code>	Prepare to store the PDF/A conformance ID, which defaults to “B”. 60 <code>\def\@pdfaconformance{B}</code> 61 <code>\define@key{Hyp}{pdfaconformance}{\hymp@pdfstringdef\@pdfaconformance{#1}}</code>
<code>\@pdfsource</code>	Prepare to store the document’s source, which defaults to the value of <code>\jobname</code> . 62 <code>\edef\@pdfsource{\jobname.tex}</code> 63 <code>\define@key{Hyp}{pdfsource}{\hymp@pdfstringdef\@pdfsource{#1}}</code>
<code>\hymp@DocumentID</code>	Prepare to store a UUID that represents the document. 64 <code>\def\hymp@DocumentID{}</code> 65 <code>\define@key{Hyp}{pdfdocumentid}{\hymp@pdfstringdef\hymp@DocumentID{#1}}</code>
<code>\hymp@InstanceID</code>	Prepare to store a UUID that represents the current instance of the document. 66 <code>\def\hymp@InstanceID{}</code> 67 <code>\define@key{Hyp}{pdfinstanceid}{\hymp@pdfstringdef\hymp@InstanceID{#1}}</code>
<code>\@pdfversionid</code>	Prepare to store a UUID that represents the current version of the document. 68 <code>\def\@pdfversionid{}</code> 69 <code>\define@key{Hyp}{pdfversionid}{\hymp@pdfstringdef\@pdfversionid{#1}}</code>
<code>\@pdfpublication</code>	Prepare to store the name of the publication in which the document was published. 70 <code>\def\@pdfpublication{}</code> 71 <code>\define@key{Hyp}{pdfpublication}{\hymp@pdfstringdef\@pdfpublication{#1}}</code>
<code>\@pdfpubtype</code>	Prepare to store the type of the publication in which the document was published. 72 <code>\def\@pdfpubtype{}</code> 73 <code>\define@key{Hyp}{pdfpubtype}{\hymp@pdfstringdef\@pdfpubtype{#1}}</code>
<code>\@pdfbytes</code>	Prepare to store the size of the file in bytes. 74 <code>\def\@pdfbytes{}</code> 75 <code>\define@key{Hyp}{pdfbytes}{\hymp@pdfstringdef\@pdfbytes{#1}}</code>
<code>\@pdfnumpages</code>	Prepare to store the number of pages in the file. 76 <code>\def\@pdfnumpages{}</code> 77 <code>\define@key{Hyp}{pdfnumpages}{\hymp@pdfstringdef\@pdfnumpages{#1}}</code>
<code>\@pdfissn</code>	Prepare to store the ISSN of the publication in which the document was published. 78 <code>\def\@pdfissn{}</code> 79 <code>\define@key{Hyp}{pdfissn}{\hymp@pdfstringdef\@pdfissn{#1}}</code>
<code>\@pdfeissn</code>	Prepare to store the ISSN of the electronic version of the publication in which the document was published. 80 <code>\def\@pdfeissn{}</code> 81 <code>\define@key{Hyp}{pdfeissn}{\hymp@pdfstringdef\@pdfeissn{#1}}</code>

<code>\@pdfisbn</code>	Prepare to store the ISBN of the publication in which the document was published. 82 <code>\def\@pdfisbn{}</code> 83 <code>\define@key{Hyp}{pdfisbn}{\hyxmp@pdfstringdef\@pdfisbn{#1}}</code>
<code>\@pdfbookedition</code>	Prepare to store the edition of the book in which the document was published. 84 <code>\def\@pdfbookedition{}</code> 85 <code>\define@key{Hyp}{pdfbookedition}{\hyxmp@pdfstringdef\@pdfbookedition{#1}}</code>
<code>\@pdfvolumenum</code>	Prepare to store the volume identifier of the publication in which the document was published. 86 <code>\def\@pdfvolumenum{}</code> 87 <code>\define@key{Hyp}{pdfvolumenum}{\hyxmp@pdfstringdef\@pdfvolumenum{#1}}</code>
<code>\@pdfissuenum</code>	Prepare to store the identifier of the issue within a volume of the publication in which the document was published. 88 <code>\def\@pdfissuenum{}</code> 89 <code>\define@key{Hyp}{pdfissuenum}{\hyxmp@pdfstringdef\@pdfissuenum{#1}}</code>
<code>\@pdfpagerange</code>	Prepare to store the document's range of pages within the publication in which the document was published. 90 <code>\def\@pdfpagerange{}</code> 91 <code>\define@key{Hyp}{pdfpagerange}{\hyxmp@pdfstringdef\@pdfpagerange{#1}}</code>
<code>\@pdfdoi</code>	Prepare to store a DOI that represents the current instance of the document. 92 <code>\def\@pdfdoi{}</code> 93 <code>\define@key{Hyp}{pdfdoi}{\hyxmp@pdfstringdef\@pdfdoi{#1}}</code>
<code>\@pdfurl</code>	Prepare to store a URL that represents where the document can be found. Note that we do not prepend <code>baseurl</code> to the value provided. 94 <code>\def\@pdfurl{}</code> 95 <code>\define@key{Hyp}{pdfurl}{\hyxmp@pdfstringdef\@pdfurl{#1}}</code>
<code>\@pdfsubtitle</code>	Prepare to store the document's subtitle. 96 <code>\def\@pdfsubtitle{}</code> 97 <code>\define@key{Hyp}{pdfsubtitle}{\hyxmp@pdfstringdef\@pdfsubtitle{#1}}</code>

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.

<code>\@pdfcontactaddress</code>	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.

```

98 \def\@pdfcontactaddress{}
99 \define@key{Hyp}{pdfcontactaddress}{%
100   \let\xmpcomma=\hyxmp@comma
101   \def\xmpquote##1{##1}%
102   \hyxmp@pdfstringdef\@pdfcontactaddress{#1}%
103   \def\xmpcomma{,}%
104   \let\xmpquote=\relax
105 }

```

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

```

106 \def\@pdfcontactcity{}
107 \define@key{Hyp}{pdfcontactcity}{\hyxmp@pdfstringdef\@pdfcontactcity{#1}}

```

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

```

108 \def\@pdfcontactregion{}
109 \define@key{Hyp}{pdfcontactregion}{\hyxmp@pdfstringdef\@pdfcontactregion{#1}}

```

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

```

110 \def\@pdfcontactpostcode{}
111 \define@key{Hyp}{pdfcontactpostcode}{\hyxmp@pdfstringdef\@pdfcontactpostcode{#1}}

```

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

```

112 \def\@pdfcontactcountry{}
113 \define@key{Hyp}{pdfcontactcountry}{\hyxmp@pdfstringdef\@pdfcontactcountry{#1}}

```

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

```

114 \def\@pdfcontactphone{}
115 \define@key{Hyp}{pdfcontactphone}{\hyxmp@pdfstringdef\@pdfcontactphone{#1}}

```

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

```

116 \def\@pdfcontactemail{}
117 \define@key{Hyp}{pdfcontactemail}{\hyxmp@pdfstringdef\@pdfcontactemail{#1}}

```

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

```

118 \def\@pdfcontacturl{}
119 \define@key{Hyp}{pdfcontacturl}{\hyxmp@pdfstringdef\@pdfcontacturl{#1}}

```

`\hyxmp@suppress@pdf@metadata` Suppress hyperref from writing Author, Title, Subject, and other PDF metadata into the Info dictionary. This prevents conflicts between the PDF metadata and the XMP metadata, which cause PDF/A validation to fail. The PDF metadata can be restored by passing the `keeppdfinfo` option to `\hypersetup`.

```

120 \def\hyxmp@suppress@pdf@metadata{%
121   \global\let\PDF@FinishDoc=\@empty
122 }
123 \define@key{Hyp}{keeppdfinfo}[true]{%
124   \gdef\hyxmp@suppress@pdf@metadata{}%
125 }

```

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 126 \def\hyxmp@pdfauthor{}
                  127 \def\hyxmp@pdfkeywords{}

\hyxmp@redefine@Hyp If not already redefined, redefine hyperref's pdfauthor and pdfkeywords options to
                    properly handle \xmpcomma and \xmpquote.
                    128 \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.

                    129 \ifundefined{KV@Hyp@pdfauthor}{\{}{%
                    130   \ifundefined{hyxmp@Hyp@pdfauthor}{\{}{%
                    131     \expandafter\let\expandafter\hyxmp@Hyp@pdfauthor
                    132       \csname KV@Hyp@pdfauthor\endcsname
                    133   }\}%
                    134 }\}%

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

135 \define@key{Hyp}{pdfauthor}{%
136   \let\xmpcomma=\hyxmp@comma
137   \def\xmpquote####1{####1}%
138   \let\hyxmp@and=\and

```

```

139 \def\and{,}%
140 \hyxmp@Hyp@pdfauthor{##1}%
141 \global\let\hyxmp@pdfauthor=\@pdfauthor
142 \def\and{and\space}%
143 \def\xmpcomma{,}%
144 \def\xmpquote####1{"####1"%
145 \hyxmp@Hyp@pdfauthor{##1}%
146 \def\xmpcomma{,}%
147 \let\xmpquote=\relax
148 \let\and=\hyxmp@and
149 }%

```

`\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`.

```

150 \ifundefined{KV@Hyp@pdfkeywords}{%
151 \ifundefined{hyxmp@Hyp@pdfkeywords}{%
152 \expandafter\let\expandafter\hyxmp@Hyp@pdfkeywords
153 \csname KV@Hyp@pdfkeywords\endcsname
154 }{}%
155 }%

```

`\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).

```

156 \define@key{Hyp}{pdfkeywords}{%
157 \let\xmpcomma=\hyxmp@comma
158 \def\xmpquote####1{####1}%
159 \hyxmp@Hyp@pdfkeywords{##1}%
160 \global\let\hyxmp@pdfkeywords=\@pdfkeywords
161 \def\xmpcomma{,}%
162 \def\xmpquote####1{"####1"%
163 \hyxmp@Hyp@pdfkeywords{##1}%
164 \def\xmpcomma{,}%
165 \let\xmpquote=\relax
166 }%
167 }

```

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

```

168 \let\hyxmp@ProcessKeyvalOptions=\ProcessKeyvalOptions
169 \renewcommand*{\ProcessKeyvalOptions}{%
170 \hyxmp@redefine@Hyp
171 \hyxmp@ProcessKeyvalOptions
172 }

```

<code>\hyxmp@hypersetup</code> <code>\hypersetup</code>	<p>Redefine hyperref's <code>\hypersetup</code> command to invoke <code>\hyxmp@redefine@Hyp</code> before performing its normal option processing.</p> <pre> 173 \let\hyxmp@hypersetup=\hypersetup 174 \def\hypersetup{% 175 \hyxmp@redefine@Hyp 176 \hyxmp@hypersetup 177 }</pre>
<code>\hyxmp@find@metadata</code> <code>\hyxmp@concat@metadata</code>	<p>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 <code>\@pdfmetalang</code> as metadata as that value is meaningful only when used in conjunction with other information. We also don't examine <code>\@pdfapart</code> or <code>\@pdfaconformance</code> because those have nonempty default values.</p> <pre> 178 \newcommand*{\hyxmp@find@metadata}{% 179 \edef\hyxmp@concat@metadata{% 180 \@baseurl 181 \@pdfauthor 182 \@pdfauthortitle 183 \@pdfbookedition 184 \@pdfbytes 185 \@pdfcaptionwriter 186 \@pdfcontactaddress 187 \@pdfcontactcity 188 \@pdfcontactcountry 189 \@pdfcontactemail 190 \@pdfcontactphone 191 \@pdfcontactpostcode 192 \@pdfcontactregion 193 \@pdfcontacturl 194 \@pdfcopyright 195 \@pdfcreationdate 196 \@pdfdatetime 197 \@pdfdoi 198 \@pdfeissn 199 \@pdfisbn 200 \@pdfissn 201 \@pdfissuenum 202 \@pdfkeywords 203 \@pdflang 204 \@pdflicenseurl 205 \@pdfmetadatetitle 206 \@pdfmoddate 207 \@pdfnumpages 208 \@pdfpagerange 209 \@pdfpublication 210 \@pdfpubtype 211 \@pdfsubject 212 \@pdfsubtitle 213 \@pdftitle</pre>


```

214 \pdftype
215 \pdfurl
216 \pdfversionid
217 \pdfvolumenum
218 }%
219 \ifx\hyxmp@concat@metadata\@empty
220 \PackageWarningNoLine{hyperxmp}{%
221 \jobname.tex did not specify any metadata to\MessageBreak
222 include in the XMP packet.\space\space Please see the\MessageBreak
223 hyperxmp documentation for instructions on how to\MessageBreak
224 provide metadata values to hyperxmp}%
225 \fi
226 }

```

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.

```

227 \AtBeginDocument{%
228 \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`.

```

229 \ifx\pdflang\relax
230 \let\pdflang=\@empty
231 \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.

```

232 \ifx\pdfmetlang\@empty
233 \ifx\pdflang\@empty
234 \let\pdfmetlang=\hyxmp@x@default
235 \else
236 \edef\pdfmetlang{\pdflang}%
237 \fi
238 \fi
239 \hyxmp@xmlify\pdfmetlang

```

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

```

240 \ifx\pdfdatetime\@empty
241 \else
242 \edef\hyxmp@today{\pdfdatetime}%
243 \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`.

```

244 \ifmtargexp{\@pdftitle}{%
245 \ifnotmtargexp{\@title}{%
246 \hypersetup{pdftitle={\@title}}}%
247 }%
248 }%
249 {}%
250 \ifmtargexp{\@pdfauthor}{%
251 \ifnotmtargexp{\@author}{%
252 \hypersetup{pdfauthor={\@author}}}%
253 }%
254 }%
255 {}%
```

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`.

```

256 \hyxmp@at@end{%
257 \hyxmp@suppress@pdf@metadata
258 \hyxmp@find@metadata
259 \hyxmp@embed@packet
260 }%
261 }{%
262 \PackageWarningNoLine{hyperxmp}{%
263 \jobname.tex failed to include a\MessageBreak
264 \string\usepackage\string{hyperref}\string}
265 in the preamble.\MessageBreak
266 Consequently, all hyperxmp functionality will be\MessageBreak
267 disabled}%
268 }%
269 }
```

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.)

```

270 \newcommand*{\hyxmp@commas@to@list}[2]{%
271   \gdef#1{%
272     \expandafter\hyxmp@commas@to@list@i\expandafter#1#2,,%
273   }

```

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

```

\next 274 \def\hyxmp@commas@to@list@i#1#2,{%
275   \gdef\hyxmp@sublist{#2}%
276   \ifx\hyxmp@sublist\@empty
277     \let\next=\relax
278   \else
279     \hyxmp@trimspaces\hyxmp@sublist
280     \@cons{#1}{\hyxmp@sublist}%
281     \def\next{\hyxmp@commas@to@list@i{#1}}%
282   \fi
283   \next
284 }

```

`\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.

```

285 \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.

```

286 \bgroup
287   \catcode'\^^C=11
288   \gdef\hyxmp@comma{^^C}
289 \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.

```

290 \bgroup
291   \catcode'\^^U=11
292   \gdef\hyxmp@uscore{^^U}
293 \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.

```
294 \let\xmpquote=\relax
```

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

```
295 \bgroup
296 \catcode'\~ =12%
297 \gdef\xmptilde{~}%
298 \egroup
```

`\XMPTruncateList` As a workaround for the inability of older Adobe Acrobat versions to display author lists correctly we introduce a hack that replaces a list with its first element.
`\hyxmp@temp@str`
`\hyxmp@temp@list` One can then write “`\XMPTruncateList{pdfauthor}`” and have Adobe Acrobat display the author list correctly.
`\@elt`

```
299 \newcommand{\XMPTruncateList}[1]{%
300   \PackageWarning{hyperxmp}{%
301     \noexpand\XMPTruncateList has been deprecated since\MessageBreak
302     hyperxmp 4.0 and may be removed in future\MessageBreak
303     versions of the package. \noexpand\XMPTruncateList\MessageBreak
304     was found}%
305   \edef\hyxmp@temp@str{\csname hyxmp@#1\endcsname}%
306   \hyxmp@commas@to@list{\hyxmp@temp@list}{\hyxmp@temp@str}%
307   \def\@elt##1{%
308     \expandafter\gdef\csname @#1\endcsname{##1}%
309     \let\@elt=\@gobble
310   }
311   \hyxmp@temp@list
312 }
```

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:20190331223654-06’00’) [3], while XMP timestamps are of the form “YYYY-MM-DDThh:mm:ss+TT:tt” (e.g., 2019-03-31T22:36:54-06:00) [4]. The `\hyxmp@as@pdf@date` and `\hyxmp@as@xmp@date` macros defined in this section facilitate timestamp conversions to PDF and XMP formats, respectively.

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

```
\hyxmp@first@char@i 313 \def\hyxmp@first@char#1{\hyxmp@first@char@i#1\relax}
314 \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.

```

315 \def\hyxmp@as@xmp@date#1{%
316   \expandafter\ifx\hyxmp@first@char@i#1\relax D%
317     \hyxmp@pdf@to@xmp@date{#1}%
318   \else
319     #1%
320   \fi
321 }

```

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

```

322 \def\hyxmp@pdf@to@xmp@date#1:#2#3#4#5#6#7#8#9{%
323   #2#3#4#5-#6#7-#8#9%
324   \hyxmp@parse@time
325 }

```

`\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`.

```

326 \def\hyxmp@parse@time#1#2#3#4#5#6{%
327   T#1#2:#3#4:#5#6%
328   \hyxmp@parse@tz@char
329 }

```

`\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.

```

330 \def\hyxmp@parse@tz@char#1{%
331   #1%
332   \ifx#1-%
333     \expandafter\hyxmp@parse@tz
334   \else
335     \ifx#1+%
336       \expandafter\hyxmp@parse@tz
337     \fi
338   \fi
339 }

```

`\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.

```

340 \def\hyxmp@parse@tz#1'#2'{%
341   #1:#2%
342 }

```

`\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.

```

343 \def\hyxmp@as@pdf@date#1{%
344   \expandafter\ifx\hyxmp@first@char@i#1\relax D%
345     #1%
346   \else
347     \hyxmp@xmp@to@pdf@date{#1}%
348   \fi
349 }

```

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

```

350 \def\hyxmp@xmp@to@pdf@date#1{%
351   D:\hyxmp@xmp@to@pdf@date@i#1\relax\relax
352 }

```

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

```

353 \def\hyxmp@xmp@to@pdf@date@i#1#2#3#4#5#6{%
354   #1#2#3#4%
355   \ifx#5-%
356     \expandafter\hyxmp@xmp@to@pdf@date@ii\expandafter#6%
357   \fi
358 }

```

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

```

359 \def\hyxmp@xmp@to@pdf@date@ii#1#2#3#4{%
360   #1#2%
361   \ifx#3-%
362     \expandafter\hyxmp@xmp@to@pdf@date@iii\expandafter#4%
363   \fi
364 }

```

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

```

365 \def\hyxmp@xmp@to@pdf@date@iii#1#2#3#4{%
366   #1#2%
367   \ifx#3T%
368     \expandafter\hyxmp@xmp@to@pdf@date@iv\expandafter#4%
369   \fi
370 }

```

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

```

371 \def\hyxmp@xmp@to@pdf@date@iv#1#2#3#4{%
372   #1#2%
373   \ifx#3:%
374     \expandafter\hyxmp@xmp@to@pdf@date@v\expandafter#4%
375   \fi
376 }

```

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

```

377 \def\hyxmp@xmp@to@pdf@date@v#1#2#3#4{%
378   #1#2%
379   \ifx#3:%
380     \expandafter\hyxmp@xmp@to@pdf@date@vi\expandafter#4%
381   \fi
382 }

```

`\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.

```

383 \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).

```

384 \def\hyxmp@xmp@to@pdf@date@vi#1#2#3#4{%
385   #1#2%
386   \ifx#3+%
387     +\expandafter\hyxmp@xmp@to@pdf@date@vii
388   \fi
389   \ifx#3-%
390     -\expandafter\hyxmp@xmp@to@pdf@date@vii
391   \fi
392   \ifx#3Z%
393     Z%
394   \fi
395   \ifx#3\relax
396     \expandafter\@hyxmp@gobbletwo
397   \fi
398   \@gobbletwo #4%
399 }

```

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

```

400 \def\hyxmp@xmp@to@pdf@date@vii#1\@gobbletwo#2#3#4#5{%
401   #2#3%
402   \ifx#4:%
403     \expandafter\hyxmp@xmp@to@pdf@date@viii\expandafter#5%
404   \fi
405 }

```

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

```

406 \def\hyxmp@xmp@to@pdf@date@viii#1#2#3#4{%
407   '#1#2'%
408 }

```

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

```

409 \def\hyxmp@today@define#1{%
    The date is a straightforward representation of TEX's \year, \month, and \day
    primitives, with the latter two zero-padded to two digits apiece.
410 \xdef#1{\the\year}%
411 \ifnum\month<10
412     \xdef#1{#1-0\the\month}%
413 \else
414     \xdef#1{#1-\the\month}%
415 \fi
416 \ifnum\day<10
417     \xdef#1{#1-0\the\day}%
418 \else
419     \xdef#1{#1-\the\day}%
420 \fi

```

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

```

421 \@tempcnta=\time
422 \divide\@tempcnta by 60%
423 \ifnum\@tempcnta<10%
424     \xdef#1{#1T0\the\@tempcnta}%
425 \else
426     \xdef#1{#1T\the\@tempcnta}%
427 \fi
428 \multiply\@tempcnta by -60%
429 \advance\@tempcnta by \time
430 \ifnum\@tempcnta<10%
431     \xdef#1{#1:0\the\@tempcnta}%
432 \else
433     \xdef#1{#1:\the\@tempcnta}%
434 \fi
435 }

```

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

```

436 \@ifundefined{pdffeedback}{%
437     \@ifundefined{pdfcreationdate}{%

```

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

```

438     \hyxmp@today@define\hyxmp@today
439 }{%

```

Case 2: `\pdfcreationdate` is defined (pdfL_AT_EX and pre-0.85 LuaL_AT_EX).

```

440     \edef\hyxmp@today{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
441 }%
442 }{%

```


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

```
443 \edef\hyxmp@today{\expandafter\hyxmp@pdf@to@xmp@date\pdffeedback creationdate}%
444 }
```

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.

```
445 \catcode'\Q=3
\hyxmp@trimspaces\x redefines \x to have the same replacement text sans leading
and trailing space tokens.
446 \newcommand{\hyxmp@trimspaces}[1]{%
  Use grouping to emulate a multi-token afterassignment queue.
447 \begingroup
  Put “\toks 0 {” into the afterassignment queue.
448 \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.
449 \expandafter\hyxmp@trimb\expandafter\noexpand#1Q Q}%
  Transfer the trimmed text back into #1.
450 \edef#1{\the\toks0}%
451 }
```

`\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.

```
452 \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.

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

3.3.4 Converting text to XML

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

```

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

\hyxmp@reencode This is now a placeholder macro needed only for \@pdfmetalang in the
\begin{document}.
461 \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.
462 \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.
463 \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;.
464 \newcommand*{\hyxmp@xmlify}[1]{%
465   \gdef\hyxmp@xmlified{%
Escaped PDF string → PDFDocEncoding/Unicode
466   \EdefUnescapeString\hyxmp@text{#1}%
467   \ifhyxmp@unicodetex
PDFDocEncoding/Unicode → UTF-32BE
468     \hyxmp@is@unicode\hyxmp@text{%
469       \StringEncodingConvert
470       \hyxmp@text\hyxmp@text{utf16be}{utf32be}%
471     }{%

```

```

472     \ifxetex
473       \hyxmp@xetex@crap
474     \else
475       \StringEncodingConvert
476       \hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
477     \fi
478   }%

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

  UTF-32BE → XML in ASCII
480   \edef\hyxmp@text{%
481     \expandafter
482   }\expandafter\hyxmp@toxml@unicodetex\hyxmp@text
483   \relax\relax\relax\relax\relax\relax\relax\relax
484   \else

  PDFDocEncoding/Unicode → UTF-8
485   \hyxmp@is@unicode\hyxmp@text{%
486     \StringEncodingConvert
487     \hyxmp@text\hyxmp@text{utf16be}{utf8}%
488   }{%
489     \StringEncodingConvert
490     \hyxmp@text\hyxmp@text{pdfdoc}{utf8}%
491   }%

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

  UTF-8 as hex string → XML in UTF-8 as hex string
493   \edef\hyxmp@text{%
494     \expandafter\hyxmp@toxml\hyxmp@text\@empty\@empty
495   }%

  XML in UTF-8 as hex string → XML in UTF-8
496   \EdefUnescapeHex\hyxmp@text{\hyxmp@text}%
497   \fi
498   \global\let\hyxmp@xmlified\hyxmp@text
499 }

```

`\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.

```

500 \begingroup
501   \lccode'\<=254 %
502   \lccode'\>=255 %
503   \catcode254=12 %
504   \catcode255=12 %
505 \lowercase{\endgroup
506   \def\hyxmp@is@unicode#1{%
507     \expandafter\hyxmp@@is@unicode#1<>\@nil
508   }%

```

```

509 \def\hyxmp@@is@unicode#1<>#2\@nil{%
510   \ifx\\#1\\%
511     \expandafter\@firstoftwo
512   \else
513     \expandafter\@secondoftwo
514   \fi
515 }%
516 }

```

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

```

517 \def\hyxmp@toxml#1#2{%
518   \ifx#1\@empty
519   \else
520     \ifnum"#1#2='\& %
521       26616D703B% &amp;
522     \else\ifnum"#1#2='\< %
523       266C743B% &lt;
524     \else\ifnum"#1#2='\> %
525       2667743B% &gt;
526   \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.

```

527   \@ifundefined{pdfmark}{%
528     #1#2%
529   }{%
530     \ifnum"#1#2='\( %
531       5C28% \(
532     \else\ifnum"#1#2='\) %
533       5C29% \)
534     \else
535       #1#2%
536     \fi\fi
537   }%
538   \fi\fi\fi
539   \expandafter\hyxmp@toxml
540 \fi
541 }

```

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

```

542 \def\hyxmp@toxml@unicodetex#1#2#3#4#5#6#7#8{%
543   \ifx#1\relax
544   \else
545     \ifnum"#1#2#3#4#5#6#7#8>127 %
546       \uccode'\*="#1#2#3#4#5#6#7#8\relax
547       \uppercase{%
548         \edef\hyxmp@text{\hyxmp@text *}%
549       }%
550     \else\ifnum"#7#8='< %
551       \edef\hyxmp@text{\hyxmp@text &lt;};}%
552     \else\ifnum"#7#8='& %
553       \edef\hyxmp@text{\hyxmp@text &amp};}%
554     \else\ifnum"#7#8='> %
555       \edef\hyxmp@text{\hyxmp@text &gt;};}%
556     \else\ifnum"#7#8='\ %
557       \edef\hyxmp@text{\hyxmp@text\space}%
558     \else
559       \uccode'\*="#7#8\relax
560       \uppercase{%
561         \edef\hyxmp@text{\hyxmp@text *}%
562       }%
563     \fi\fi\fi\fi\fi
564     \expandafter\hyxmp@toxml@unicodetex
565   \fi
566 }
```

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

```

567 \def\hyxmp@skipzeros#1{%
568   \ifx#10%
569     \expandafter\hyxmp@skipzeros
570   \fi
571 }
```

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

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

```

\hyxmp@try 572 \begingroup
\hyxmp@crap@result 573 \def\x#1{\endgroup
\hyxmp@text 574   \def\hyxmp@xetex@crap{%
575     \edef\hyxmp@try{%
576       \expandafter\hyxmp@SpaceOther\hyxmp@text#1@nil
577     }%
578     \let\hyxmp@crap@result=N%
579     \expandafter\hyxmp@crap@test\hyxmp@try\relax
580     \ifx\hyxmp@crap@result Y%
581       \let\hyxmp@text\@empty
582       \expandafter\hyxmp@crap@convert\hyxmp@try\relax
583     \else
```

```

584      \StringEncodingConvert\hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
585      \fi
586    }%
587  }
588  \x{ }

```

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

```

589 \begingroup
590   \catcode'\~=12 %
591   \lccode'\~=\' %
592 \lowercase{\endgroup
593   \def\hyxmp@SpaceOther#1 #2\@nil{%
594     #1%
595     \ifx\relax#2\relax
596       \expandafter\@gobble
597     \else
598       ~%
599       \expandafter\@firstofone
600     \fi
601     {\hyxmp@SpaceOther#2\@nil}%
602   }%
603 }

```

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

```

604 \def\hyxmp@crap@test#1{%
605   \ifx#1\relax
606   \else
607     \ifnum'#1>127 %
608       \let\hyxmp@crap@result=Y%
609       \expandafter\expandafter\expandafter\hyxmp@skiptorelax
610     \else
611       \expandafter\expandafter\expandafter\hyxmp@crap@test
612     \fi
613   \fi
614 }

```

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

```

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

```

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

```

\hyxmp@num 616 \def\hyxmp@crap@convert#1{%
\hyxmp@text 617   \ifx#1\relax
618   \else
619     \edef\hyxmp@num{\number'#1}%
620     \ifnum\hyxmp@num>"FFFFFF %
621       \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"1000000}\relax
622       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
623     \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"1000000}}%
624   \else

```

```

625     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
626   \fi
627   \ifnum\hyxmp@num>"FFFF %
628     \lccode'\!=\intcalDiv{\hyxmp@num}{\number"10000}\relax
629     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
630     \edef\hyxmp@num{\intcalMod{\hyxmp@num}{\number"10000}}%
631   \else
632     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
633   \fi
634   \ifnum\hyxmp@num>"FF %
635     \lccode'\!=\intcalDiv{\hyxmp@num}{\number"100}\relax
636     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
637     \edef\hyxmp@num{\intcalMod{\hyxmp@num}{\number"100}}%
638   \else
639     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
640   \fi
641   \ifnum\hyxmp@num>0 %
642     \lccode'\!=\hyxmp@num\relax
643     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
644   \else
645     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
646   \fi
647   \expandafter\hyxmp@crap@convert
648 \fi
649 }

```

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

```

650 \begingroup
651   \catcode0=12 %
652   \gdef\hyxmp@zero{^^00}%
653 \endgroup

```

3.3.5 Providing metadata in multiple languages

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

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

```

654 \def\hyxmp@alt@title{}
655 \def\hyxmp@alt@description{}
656 \def\hyxmp@alt@rights{}

```

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

```

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

```

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

```

659   \hyxmp@pdfstringdef\hyxmp@value{##1}%
660   \xdef#2{#2\noexpand\do {\hyxmp@cur@lang} {\hyxmp@value}}%
661 }
662 }

```

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

```

663 \hyxmp@LA@accept{pdftitle}{\hyxmp@alt@title}
664 \hyxmp@LA@accept{pdfsubject}{\hyxmp@alt@description}
665 \hyxmp@LA@accept{pdfcopyright}{\hyxmp@alt@rights}

```

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

```

666 \newcommand{\XMPLangAlt}[2]{%
667   \let\do=\relax

```

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

```

668   \edef\hyxmp@cur@lang{#1}%
669   \setkeys{hyxmp@LA}{#2}%
670 }

```

3.4 UUID generation

We use a linear congruential generator to produce pseudorandom version 4 UUIDs [11]. True, this method has its flaws but it’s simple to implement in T_EX and is good enough for producing the XMP `xmpMM:DocumentID` and `xmpMM:InstanceID` fields.

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

```

671 \def\hyxmp@modulo@a#1{%
672   \@tempcntb=\@tempcnta
673   \divide\@tempcntb by #1
674   \multiply\@tempcntb by #1
675   \advance\@tempcnta by -\@tempcntb
676 }

```


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

```

\hyxmp@big@prime@ii 677 \def\hyxmp@big@prime{536870923}
678 \def\hyxmp@big@prime@ii{536870027}

```

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

```

\hyxmp@one@token 679 \def\hyxmp@seed@rng#1{%
680   \@tempcnta=\hyxmp@big@prime
681   \futurelet\hyxmp@one@token\hyxmp@seed@rng@i#1\@empty
682 }

```

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

```

\hyxmp@one@token 683 \def\hyxmp@seed@rng@i{%
684   \ifx\hyxmp@one@token\@empty
685     \let\next=\relax
686   \else
687     \def\next##1{%
688       \multiply\@tempcnta by 3
689       \advance\@tempcnta by '##1
690       \hyxmp@modulo@a{\hyxmp@big@prime}%
691       \futurelet\hyxmp@one@token\hyxmp@seed@rng@i
692     }%
693   \fi
694   \next
695 }

```

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

```

\hyxmp@rand@num 696 \def\hyxmp@set@rand@num{%
697   \@tempcnta=\hyxmp@rand@num
698   \multiply\@tempcnta by 3
699   \advance\@tempcnta by \hyxmp@big@prime@ii
700   \hyxmp@modulo@a{\hyxmp@big@prime}%
701   \xdef\hyxmp@rand@num{\the\@tempcnta}%
702 }

```

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

```

703 \def\hyxmp@append@hex#1{%
704   \hyxmp@set@rand@num
705   \@tempcnta=\hyxmp@rand@num
706   \hyxmp@modulo@a{16}%
707   \ifnum\@tempcnta<10
708     \xdef#1{#1\the\@tempcnta}%
709   \else

```

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

```

710     \advance\@tempcnta by -10
711     \ifcase\@tempcnta
712         \xdef#1{#1a}%
713         \or\xdef#1{#1b}%
714         \or\xdef#1{#1c}%
715         \or\xdef#1{#1d}%
716         \or\xdef#1{#1e}%
717         \or\xdef#1{#1f}%
718     \fi
719 \fi
720 }

```

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

```

721 \def\hyxmp@append@hex@iii#1{%
722     \hyxmp@append@hex#1%
723     \hyxmp@append@hex#1%
724     \hyxmp@append@hex#1%
725 }

```

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

```

726 \def\hyxmp@append@hex@iv#1{%
727     \hyxmp@append@hex@iii#1%
728     \hyxmp@append@hex#1%
729 }

```

`\hyxmp@create@uuid` As per the definition of a version 4 UUID [11], 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`.

```

730 \def\hyxmp@create@uuid#1{%
731     \def#1{uuid:}%
732     \hyxmp@append@hex@iv#1%
733     \hyxmp@append@hex@iv#1%
734     \g@addto@macro#1{-}%
735     \hyxmp@append@hex@iv#1%
736     \g@addto@macro#1{-4}%
737     \hyxmp@append@hex@iii#1%
738     \g@addto@macro#1{-}%

```

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

```

739     \hyxmp@set@rand@num
740     \@tempcnta=\hyxmp@rand@num
741     \hyxmp@modulo@a{4}%
742     \ifcase\@tempcnta
743         \g@addto@macro#1{8}%
744         \or\g@addto@macro#1{9}%
745         \or\g@addto@macro#1{a}%

```

```

746     \or\g@addto@macro#1{b}%
747   \fi
748   \hyxmp@append@hex@iii#1%
749   \g@addto@macro#1{-}%
750   \hyxmp@append@hex@iv#1%
751   \hyxmp@append@hex@iv#1%
752   \hyxmp@append@hex@iv#1%
753 }

\hyxmp@def@DocumentID Seed the random-number generator with a function of the current filename, PDF
  \hyxmp@DocumentID document title, and PDF author, then invoke \hyxmp@create@uuid to define
  \hyxmp@seed@string \hyxmp@DocumentID as a random UUID.
754 \newcommand*{\hyxmp@def@DocumentID}{%
755   \edef\hyxmp@seed@string{\jobname:\@pdftitle:\@pdfauthor}%
756   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
757   \edef\hyxmp@rand@num{\the\@tempcnta}%
758   \hyxmp@create@uuid\hyxmp@DocumentID
759 }

\hyxmp@def@InstanceID Seed the random-number generator with a function of the current filename,
  \hyxmp@InstanceID PDF document title, PDF author, and the current timestamp, then invoke
  \hyxmp@seed@string \hyxmp@create@uuid to define \hyxmp@InstanceID as a random UUID. For the
  current timestamp, we use both the document-specified timestamp from pdfdate
  and the TEX time. The former can be more precise (to sub-seconds) but may be
  less random (as it depends on manual document modifications) while the latter
  is typically less precise (to minutes) but may be more random (as it is updated
  automatically).
760 \newcommand*{\hyxmp@def@InstanceID}{%
761   \hyxmp@today@define{\hyxmp@seed@string}%
762   \edef\hyxmp@seed@string{%
763     \jobname:\@pdftitle:\@pdfauthor:\hyxmp@today:\hyxmp@seed@string
764   }%
765   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
766   \edef\hyxmp@rand@num{\the\@tempcnta}%
767   \hyxmp@create@uuid\hyxmp@InstanceID
768 }

```

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 3.5.9), and PDF/A Identification (Section 3.5.8). The `\hyxmp@construct@packet` macro (Section 3.5.12) 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.

```

769 \newcommand*{\hyxmp@add@to+xml}[1]{%
770   \bgroup
771   \@tempcnta=0
772   \ifhyxmp@unicodetex
773     \@tempcntb=65536%
774   \else
775     \@tempcntb=256%
776   \fi
777   \loop
778     \lccode\@tempcnta=\@tempcnta
779     \advance\@tempcnta by 1
780     \ifnum\@tempcnta<\@tempcntb
781       \repeat
782       \lccode'\_='\ \relax
783       \lccode'\^C='\,\relax
784       \lccode'\^U='\_\relax
785       \lowercase{\xdef\hyxmp@new+xml{#1}}%
786       \xdef\hyxmp+xml{\hyxmp+xml\hyxmp@new+xml}%
787   \egroup
788 }
```

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

```

789 \bgroup
790 \catcode'\#=11
791 \gdef\hyxmp@hash{#}
792 \egroup
```

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

```

793 \bgroup
794 \xdef\hyxmp+xml{%
795   \hyxmp@add@to+xml{%
796     -----^^J%
797   }
798   \xdef\hyxmp@padding{\hyxmp+xml}%
799 \egroup
800 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
801 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
802 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
```

```

803 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
804 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}

```

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

```

805 \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.

```

806 \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.

```

807 \hyxmp@add@simple@var{pdf:Keywords}{@pdfkeywords}%
808 \hyxmp@add@simple@var{pdf:Producer}{@pdfproducer}%
809 \@ifundefined{pdfvariable}{%
810 \@ifundefined{pdfminorversion}{%

```

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

```

811 }{%

```

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

```

812 \hyxmp@add@simple{pdf:PDFVersion}{1.\the\pdfminorversion}%
813 }%
814 }{%

```

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

```

815 \hyxmp@add@simple{pdf:PDFVersion}{1.\the\pdfvariable minorversion}%
816 }%
817 }

```

`\hyxmp@extra@indent` This macro is used internally to increase the amount of indentation when writing certain XML data. It is normally defined as empty but can temporarily be redefined to a sequence of `\space` characters.

```

818 \newcommand*{\hyxmp@extra@indent}{}

```

`\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.

```

819 \newcommand*{\hyxmp@add@simple}[2]{%
820 \@ifnotmtargexp{#2}{%

```

```

821 \hyxmp@xmllify{#2}%
822 \hyxmp@add@to@xml{%
823   \hyxmp@extra@indent_____<#1>\hyxmp@xmllified</#1>^^J%
824 }%
825 }%
826 }

```

`\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.

```

827 \newcommand*{\hyxmp@add@simple@var}[2]{%
828   \expandafter\ifx\csname#2\endcsname\relax
829   \else
830     \hyxmp@xmllify{\csname#2\endcsname}%
831     \hyxmp@add@to@xml{%
832       \hyxmp@extra@indent_____<#1>\hyxmp@xmllified</#1>^^J%
833     }%
834   \fi
835 }

```

`\hyxmp@add@simple@lang` 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. However, if the string begins with a language code in square brackets, specify that as the (sole) language for the tag.

```

836 \newcommand*{\hyxmp@add@simple@lang}[2]{%
837   \@ifnotmtarg{#2}{%
838     \hyxmp@xmllify{#2}%
839     \expandafter\hyxmp@add@simple@lang@i\hyxmp@xmllified\relax{#1}%
840   }%
841 }

```

`\hyxmp@add@simple@lang@i` This is a helper macro for `\hyxmp@add@simple@lang`. It takes an optional language code (in brackets), text up to `\relax`, and a tag, and typesets the text within the XML tag.

```

842 \newcommand*{\hyxmp@add@simple@lang@i}{%
843   \@ifnextchar[\hyxmp@add@simple@lang@ii{\hyxmp@add@simple@lang@ii[]}%
844 }

```

`\hyxmp@add@simple@lang@ii` This is another helper macro for `\hyxmp@add@simple@lang`. It takes a mandatory language code (in brackets; can be empty), text up to `\relax`, and a tag, and typesets the text within the XML tag.

```

845 \def\hyxmp@add@simple@lang@ii[#1]#2\relax#3{%
846   \@ifnotmtarg{#2}{%
847     \hyxmp@xmllify{#2}%

```

```

848     \@ifmtarg{#1}{%
849         \hyxmp@add@to@xml{%
850     -----<#3>\hyxmp@xmlified</#3>^^J%
851         }%
852     }{%
853         \hyxmp@add@to@xml{%
854     -----<#3 xml:lang="#1">\hyxmp@xmlified</#3>^^J%
855         }%
856     }%
857 }%
858 }

```

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.

```

859 \newcommand*{\hyxmp@rdf@dc}[3][\iffalse]{%
    Set \@tempswatrue only if the given text is nonempty or the provided conditional
    evaluates to TRUE.

860     \@ifmtargexp{#3}{\@tempswafalse}{\@tempswatrue}%
861     #1
862     \@tempswatrue
863     \fi

    Append the corresponding XML only if \@tempswatrue.

864     \if@tempswa
865         \hyxmp@xmlify{#3}%

```

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

```

866     \let\hyxmp@value=\hyxmp@xmlified
867     \hyxmp@add@to@xml{%
868     -----<dc:#2>^^J%
869     -----<rdf:Alt>^^J%
870         }%
871         \ifx\@pdfmetalang\hyxmp@x@default
872         \else
873             \hyxmp@xmlify{\@pdfmetalang}%
874             \hyxmp@add@to@xml{%
875     -----<rdf:li xml:lang="\hyxmp@xmlified">\hyxmp@value</rdf:li>^^J%
876                 }%
877         \fi
878         \hyxmp@add@to@xml{%
879     -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@value</rdf:li>^^J%
880                 }%

```

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

```

881 \def\do##1##2{
882 \hyxmp@xmlify{##2}%
883 \hyxmp@add@to@xml{%
884 -----<rdf:li xml:lang="##1">\hyxmp@xmlified</rdf:li>^^J%
885 }%
886 }%
887 \csname hyxmp@alt@#2\endcsname
Complete this XMP element.
888 \hyxmp@add@to@xml{%
889 -----</rdf:Alt>^^J%
890 -----</dc:#2>^^J%
891 }%
892 \fi
893 }%

```

`\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.

```

894 \newcommand*{\hyxmp@list@to@xml}[4][\iffalse]{%
Set \@tempswatrue only if the given list is nonempty or the provided conditional
evaluates to TRUE.
895 \@ifmtargexp{#4}{\@tempswafalse}{\@tempswatrue}%
896 #1
897 \@tempswatrue
898 \fi

```

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

```

899 \if@tempswa
900 \hyxmp@add@to@xml{%
901 -----<dc:#2>^^J%
902 -----<rdf:#3>^^J%
903 }%
904 \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`.

```

905 \hyxmp@xmlify{#4}%
906 \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
907 \def\@elt##1{%
908 \hyxmp@add@to@xml{%
909 -----<rdf:li>##1</rdf:li>^^J%
910 }%
911 }%
912 \hyxmp@list
913 \egroup
914 \hyxmp@add@to@xml{%
915 -----</rdf:#3>^^J%
916 -----</dc:#2>^^J%

```



```

917   }%
918   \fi
919 }

```

`\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.

```

920 \newcommand*{\hyxmp@dc@schema}{%
921   \hyxmp@add@simple{dc:format}{application/pdf}%
922   \hyxmp@rdf@dc[\ifHy@pdfa]{title}{\@pdftitle}%
923   \hyxmp@rdf@dc[\ifHy@pdfa]{description}{\@pdfsubject}%
924   \hyxmp@rdf@dc{rights}{\@pdfcopyright}%
925   \hyxmp@list@to@xml[\ifHy@pdfa]{creator}{Seq}{\hyxmp@pdfauthor}%
926   \hyxmp@list@to@xml{subject}{Bag}{\hyxmp@pdfkeywords}%
927   \hyxmp@list@to@xml{date}{Seq}{\hyxmp@today}%
928   \hyxmp@list@to@xml{language}{Bag}{\@pdflang}%
929   \hyxmp@list@to@xml{type}{Bag}{\@pdfstype}%
930   \ifx\@pdfsource\@empty
931   \else
932     \hyxmp@add@simple{dc:source}{\@pdfsource}%
933   \fi
934 }

```

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.

```

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

```

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

```

936   \let\hyxmp@rights=\@empty
937   \ifx\@pdflicenseurl\@empty
938   \else
939     \def\hyxmp@rights{YES}%
940   \fi
941   \ifx\@pdfcopyright\@empty
942   \else
943     \def\hyxmp@rights{YES}%
944   \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.

```

945 \ifx\hyxmp@rights\@empty
946 \else
947 \ifx\@pdfcopyright\@empty
948 \else
949 \hyxmp@add@simple{xmpRights:Marked}{True}%
950 \fi
951 \hyxmp@add@simple{xmpRights:WebStatement}{\@pdflicenseurl}%
952 \fi
953 }

```

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. We additionally support the `xmpMM:VersionID` property, whose value is supplied by the author using `pdfversionid`.

```

954 \gdef\hyxmp@mm@schema{%
955 \ifmtargexp{\hyxmp@DocumentID}{\hyxmp@def@DocumentID}{}%
956 \ifmtargexp{\hyxmp@InstanceID}{\hyxmp@def@InstanceID}{}%
957 \hyxmp@add@simple{xmpMM:DocumentID}{\hyxmp@DocumentID}%
958 \hyxmp@add@simple{xmpMM:InstanceID}{\hyxmp@InstanceID}%
959 \hyxmp@add@simple{xmpMM:VersionID}{\@pdfversionid}%
960 }

```

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 `xmp:CreateDate`, `xmp:ModifyDate`, and `xmp:MetadataDate` fields.

```

961 \newcommand*{\hyxmp@define@createdate}{%
962 \ifundefined{pdffeedback}{%
963 \ifundefined{pdfcreationdate}{%

```

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

```

964 \hyxmp@today@define\hyxmp@createdate
965 }{%

```

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

```

966 \edef\hyxmp@createdate{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
967 }%
968 }{%

```

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

```

969 \edef\hyxmp@createdate{\expandafter\hyxmp@pdf@to@xmp@date\pdffeedback creationdate}%
970 }%
971 }

```

`\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`.

```

972 \newcommand*{\hyxmp@xmp@basic@schema}{%
973 \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.

974 \ifundefined{\pdfcreationdate}{%
975 \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@createdate}%
976 }{%
977 \ifx\pdfcreationdate\@empty
978 \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@createdate}%
979 \else
980 \hyxmp@add@simple{xmp:CreateDate}{%
981 \expandafter\hyxmp@as@xmp@date\expandafter{\pdfcreationdate}}%
982 \fi
983 }%

```

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

```

984 \ifundefined{\pdfmoddate}{%
985 \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@createdate}%
986 }{%
987 \ifx\pdfmoddate\@empty
988 \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@createdate}%
989 \else
990 \hyxmp@add@simple{xmp:ModifyDate}{%
991 \expandafter\hyxmp@as@xmp@date\expandafter{\pdfmoddate}}%
992 \fi
993 }%

```

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

```

994 \ifx\pdfmetadatettime\@empty
995 \hyxmp@add@simple{xmp:MetadataDate}{\hyxmp@createdate}%
996 \else
997 \hyxmp@add@simple{xmp:MetadataDate}{\pdfmetadatettime}%
998 \fi

```

Define the creation tool and the base URL.

```

999 \hyxmp@add@simple{xmp:CreatorTool}{\pdfcreator}%
1000 \hyxmp@add@simple{xmp:BaseURL}{\@baseurl}%
1001 }

```

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.

```
1002 \gdef\hyxmp@photoshop@schema{%
1003   \edef\hyxmp@photoshop@data{\@pdfauthortitle\@pdfcaptionwriter}%
1004   \hyxmp@add@simple{photoshop:AuthorsPosition}{\@pdfauthortitle}%
1005   \hyxmp@add@simple{photoshop:CaptionWriter}{\@pdfcaptionwriter}%
1006 }
```

3.5.8 The PDF/A Identification schema

`\hyxmp@pdfa@id@schema` Add properties defined by the PDF/A Identification schema [12] 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`.

```
1007 \newcommand*{\hyxmp@pdfa@id@schema}{%
1008   \ifHy@pdfa
1009     \hyxmp@add@simple{pdfaid:part}{\@pdfapart}%
1010     \hyxmp@add@simple{pdfaid:conformance}{\@pdfaconformance}%
1011   \fi
1012 }
```

3.5.9 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.

```
1013 \begingroup
1014   \catcode'\&=12
1015   \catcode'\#=12
1016   \gdef\xmplinesep{&#xA;}
1017 \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.

```
1018 \newcommand*{\hyxmp@list@to@lines}[2]{%
1019   \@ifnotmtargexp{#2}{%
1020     \bgroup
1021     \hyxmp@add@to@xml{%
1022       \hyxmp@extra@indent_____<#1>%
1023     }%
1024   }
```

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

```
1024   \def\@elt@first##1{%
1025     \hyxmp@add@to@xml{##1}%
1026     \let\@elt=\@elt@rest
1027   }
```

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

```
1028     \def\@elt@rest##1{%
1029         \hyxmp@add@to@xml{\xmplinesep##1}%
1030     }%
```

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

```
1031     \let\@elt=\@elt@first
1032     \hyxmp@xmllify{#2}%
1033     \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmllified}%
1034     \hyxmp@list
1035     \hyxmp@add@to@xml{</#1>^^J}%
1036 \egroup
1037 }%
1038 }
```

`\hyxmp@iptc@schema` Add properties defined by the IPTC Photo Metadata schema [9] to the `\hyxmp@xml` macro. We currently support only the `lptc4xmpCore:CreatorContactInfo` property, although this is a structure containing multiple fields.

```
1039 \gdef\hyxmp@iptc@schema{%
```

Because we currently support only `lptc4xmpCore:CreatorContactInfo` it suffices to check if we have any relevant data. If so, we instantiate a `lptc4xmpCore:ContactInfo` structure with all available fields.

```
1040 \ifx\hyxmp@iptc@data\@empty
1041 \else
1042     \hyxmp@add@to@xml{%
1043 -----<Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">^^J%
1044     }%
```

We locally redefine `\hyxmp@extra@indent` to increase the indentation of the assignments to `lptc4xmpCore:CreatorContactInfo`'s fields.

```
1045     \bgroup
1046     \edef\hyxmp@extra@indent{\hyxmp@extra@indent\space\space}%
1047     \hyxmp@list@to@lines{Iptc4xmpCore:CiAdrExtadr}{\@pdfcontactaddress}%
1048     \hyxmp@add@simple{Iptc4xmpCore:CiAdrCity}{\@pdfcontactcity}%
1049     \hyxmp@add@simple{Iptc4xmpCore:CiAdrRegion}{\@pdfcontactregion}%
1050     \hyxmp@add@simple{Iptc4xmpCore:CiAdrPcode}{\@pdfcontactpostcode}%
1051     \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” [9]. 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.

```
1052     \def\xmplinesep{,}%
```

```

1053 \hyxmp@list@to@lines{Iptc4xmpCore:CTelWork}{\@pdfcontactphone}%
1054 \hyxmp@list@to@lines{Iptc4xmpCore:CiEmailWork}{\@pdfcontactemail}%
1055 \hyxmp@list@to@lines{Iptc4xmpCore:CiUrlWork}{\@pdfcontacturl}%
1056 \egroup
1057 \hyxmp@add@to@xml{%
1058 -----</Iptc4xmpCore:CreatorContactInfo>^^J%
1059 }%
1060 \fi
1061 }

```

3.5.10 The PRISM Basic Metadata schema

\hyxmp@prism@schema Add properties defined by the PRISM Basic Metadata schema [7].

```

1062 \newcommand*{\hyxmp@prism@schema}{%
1063 \ifx\hyxmp@prism@data\@empty
1064 \else
1065 \hyxmp@add@simple{prism:complianceProfile}{three}%
1066 \fi
1067 \hyxmp@add@simple@lang{prism:subtitle}{\@pdfsubtitle}%
1068 \hyxmp@add@simple@lang{prism:publicationName}{\@pdfpublication}%
1069 \hyxmp@add@simple{prism:aggregationType}{\@pdfpubtype}%
1070 \hyxmp@add@simple@lang{prism:bookEdition}{\@pdfbookedition}%
1071 \hyxmp@add@simple{prism:volume}{\@pdfvolumenum}%
1072 \hyxmp@add@simple{prism:number}{\@pdfissuenum}%
1073 \hyxmp@add@simple{prism:pageRange}{\@pdfpagerange}%
1074 \hyxmp@add@simple{prism:isbn}{\@pdfisbn}%
1075 \hyxmp@add@simple{prism:issn}{\@pdfissn}%
1076 \hyxmp@add@simple{prism:eIssn}{\@pdfeissn}%
1077 \hyxmp@add@simple{prism:doi}{\@pdfdoi}%
1078 \hyxmp@add@simple{prism:url}{\@pdfurl}%
1079 \hyxmp@add@simple{prism:byteCount}{\@pdfbytes}%
1080 \hyxmp@add@simple{prism:pageCount}{\@pdfnumpages}%
1081 }

```

3.5.11 XMP extension schemata

Not all of the schemata supported by hyperxmp are predefined by XMP. PDF/A conversion would normally fail for documents that employ “custom” schemata. However, this problem can be circumvented by declaring non-standard schemata in the XMP packet itself, following a technique described in a PDF Association technical note [13]. In this section, we declare only those schemata we actually use.

\hyxmp@check@iptc@data Define \hyxmp@iptc@data as the concatenation of all IPTC photo metadata supplied by the document.

```

1082 \newcommand*{\hyxmp@check@iptc@data}{%

\hyxmp@iptc@data
1083 \edef\hyxmp@iptc@data{%
1084 \@pdfcontactaddress

```

```

1085     \@pdfcontactcity
1086     \@pdfcontactregion
1087     \@pdfcontactpostcode
1088     \@pdfcontactcountry
1089     \@pdfcontactphone
1090     \@pdfcontactemail
1091     \@pdfcontacturl
1092 }%
1093 }%

\hyxmp@check@prism@data Define \hyxmp@prism@data as the concatenation of all PRISM metadata supplied
                        by the document.
1094 \newcommand*{\hyxmp@check@prism@data}{%

\hyxmp@prism@data
1095 \edef\hyxmp@prism@data{%
1096     \@pdfbookedition
1097     \@pdfbytes
1098     \@pdfdoi
1099     \@pdfeissn
1100     \@pdfisbn
1101     \@pdfissn
1102     \@pdfissuenum
1103     \@pdfnumpages
1104     \@pdfpagerange
1105     \@pdfpublication
1106     \@pdfpubtype
1107     \@pdfsubtitle
1108     \@pdfurl
1109     \@pdfvolumenum
1110 }%
1111 }%

\hyxmp@begin@extension@decls Begin a block of XML tags that indicates we're declaring one or more extension
                             schemata.
1112 \newcommand*{\hyxmp@begin@extension@decls}{%
1113     \hyxmp@add@to+xml{%
1114         <pdfaExtension:schemas>^^J%
1115         <rdf:Bag>^^J%
1116     }%
1117 }

\hyxmp@end@extension@decls End the block of XML tags begun by \hyxmp@begin@extension@decls.
1118 \newcommand*{\hyxmp@end@extension@decls}{%
1119     \hyxmp@add@to+xml{%
1120         </rdf:Bag>^^J%
1121         </pdfaExtension:schemas>^^J%
1122     }%
1123 }

```

`\hyxmp@begin@ext@decl` Begin the declaration of a single extension schema. `\hyxmp@begin@ext@decl` accepts the schema's name, prefix, and namespace URI.

```

1124 \newcommand*{\hyxmp@begin@ext@decl}[3]{%
1125   \hyxmp@add@to@xml{%
1126     <rdf:li rdf:parseType="Resource">^^J%
1127     <pdfaSchema:schema>#1</pdfaSchema:schema>^^J%
1128     <pdfaSchema:prefix>#2</pdfaSchema:prefix>^^J%
1129     <pdfaSchema:namespaceURI>#3</pdfaSchema:namespaceURI>^^J%
1130     <pdfaSchema:property>^^J%
1131     <rdf:Seq>^^J%
1132   }%
1133 }%

```

`\hyxmp@end@ext@decl` End the declaration of a single extension schema.

```

1134 \newcommand*{\hyxmp@end@ext@decl}{%
1135   \hyxmp@add@to@xml{%
1136     </rdf:Seq>^^J%
1137     </pdfaSchema:property>^^J%
1138     </rdf:li>^^J%
1139   }%
1140 }%

```

`\hyxmp@declare@property` Declare a single extension-schema property. `\hyxmp@declare@property` takes as input an optional type (defaults to Text) and a mandatory name, category, and description.

```

1141 \newcommand{\hyxmp@declare@property}[4][Text]{%
1142   \hyxmp@add@to@xml{%
1143     <rdf:li rdf:parseType="Resource">^^J%
1144     <pdfaProperty:name>#2</pdfaProperty:name>^^J%
1145     <pdfaProperty:valueType>#1</pdfaProperty:valueType>^^J%
1146     <pdfaProperty:category>#3</pdfaProperty:category>^^J%
1147     <pdfaProperty:description>#4</pdfaProperty:description>^^J%
1148     </rdf:li>^^J%
1149   }%
1150 }%

```

`\hyxmp@declare@field` Declare a single field in a custom datatype required by an extension schema. `\hyxmp@declare@field` takes as input an optional type (defaults to Text) and a mandatory name and description.

```

1151 \newcommand{\hyxmp@declare@field}[3][Text]{%
1152   \hyxmp@add@to@xml{%
1153     <rdf:li rdf:parseType="Resource">^^J%
1154     <pdfaField:name>#2</pdfaField:name>^^J%
1155     <pdfaField:valueType>#1</pdfaField:valueType>^^J%
1156     <pdfaField:description>#3</pdfaField:description>^^J%
1157     </rdf:li>^^J%
1158   }%
1159 }

```


`\hyxmp@mm@extensions` Declare the XMP Media Management schema.

```
1160 \newcommand*{\hyxmp@mm@extensions}{%
1161   \hyxmp@begin@ext@decl
1162     {XMP Media Management Schema}%
1163     {xmpMM}%
1164     {http://ns.adobe.com/xap/1.0/mm/}%
1165   \hyxmp@declare@property
1166     [URI]
1167     {DocumentID}%
1168     {internal}%
1169     {UUID based identifier for all versions and renditions of a document}%
1170   \hyxmp@declare@property
1171     [URI]
1172     {InstanceID}%
1173     {internal}%
1174     {UUID based identifier for specific incarnation of a document}%
1175   \hyxmp@declare@property
1176     {VersionID}%
1177     {internal}%
1178     {Document version identifier}%
1179   \hyxmp@end@ext@decl
1180 }%
```

`\hyxmp@pdfa@id@extensions` Declare the PDF/A Identification schema [12].

```
1181 \newcommand*{\hyxmp@pdfa@id@extensions}{%
1182   \hyxmp@begin@ext@decl
1183     {PDF/A Identification Schema}%
1184     {pdfaid}%
1185     {http://www.aiim.org/pdfa/ns/id/}%
1186   \hyxmp@declare@property
1187     [Integer]%
1188     {part}%
1189     {internal}%
1190     {Part of PDF/A standard}%
1191   \hyxmp@declare@property
1192     {conformance}%
1193     {internal}%
1194     {Conformance level of PDF/A standard}%
1195   \hyxmp@end@ext@decl
1196 }%
```

`\hyxmp@iptc@extensions` Because IPTC metadata are not recognized by the PDF/A standard, PDF/A conversion would normally fail for documents that utilize IPTC metadata. Declaring the IPTC metadata we support enables the document to be converted to PDF/A format.

```
1197 \newcommand*{\hyxmp@iptc@extensions}{%
1198   \hyxmp@begin@ext@decl
1199     {IPTC Core Schema}%
1200     {Iptc4xmpCore}%

```

```

1201      {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}%
1202 \hyxmp@declare@property
1203   [ContactInfo]
1204   {CreatorContactInfo}
1205   {external}
1206   {Document creator's contact information}

```

We can't call \hyxmp@end@ext@decl because we need first need to define the Iptc4xmpCore:ContactInfo structure.

```

1207 \hyxmp@add@to@xml{%
1208 -----</rdf:Seq>^^J%
1209 -----</pdfaSchema:property>^^J%
1210 -----<pdfaSchema:valueType>^^J%
1211 -----<rdf:Seq>^^J%
1212 -----<rdf:li rdf:parseType="Resource">^^J%
1213 -----<pdfaType:type>ContactInfo</pdfaType:type>^^J%
1214 -----<pdfaType:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaType:
1215 -----<pdfaType:prefix>Iptc4xmpCore</pdfaType:prefix>^^J%
1216 -----<pdfaType:description>%
1217         Basic set of information to get in contact with a person%
1218 -----</pdfaType:description>^^J%
1219 -----<pdfaType:field>^^J%
1220 -----<rdf:Seq>^^J%
1221 }%
1222 \hyxmp@declare@field
1223   {CiAdrCity}%
1224   {Contact information city}%
1225 \hyxmp@declare@field
1226   {CiAdrCtry}%
1227   {Contact information country}%
1228 \hyxmp@declare@field
1229   {CiAdrExtadr}%
1230   {Contact information address}%
1231 \hyxmp@declare@field
1232   {CiAdrPcode}%
1233   {Contact information local postal code}%
1234 \hyxmp@declare@field
1235   {CiAdrRegion}%
1236   {Contact information regional information such as state or province}%
1237 \hyxmp@declare@field
1238   {CiEmailWork}%
1239   {Contact information email address(es)}%
1240 \hyxmp@declare@field
1241   {CiTelWork}%
1242   {Contact information telephone number(s)}%
1243 \hyxmp@declare@field
1244   {CiUrlWork}%
1245   {Contact information Web URL(s)}%
1246 \hyxmp@add@to@xml{%
1247 -----</rdf:Seq>^^J%

```

```

1248 -----</pdfaType:field>^^J%
1249 -----</rdf:li>^^J%
1250 -----</rdf:Seq>^^J%
1251 -----</pdfaSchema:valueType>^^J%
1252 -----</rdf:li>^^J%
1253  }%
1254 }

```

`\hyxmp@prism@extensions` Because PRISM metadata are not recognized by the PDF/A standard, PDF/A conversion would normally fail for documents that utilize PRISM metadata. Declaring the PRISM metadata we support enables the document to be converted to PDF/A format.

```

1255 \newcommand*{\hyxmp@prism@extensions}{%
1256   \hyxmp@begin@ext@decl
1257     {PRISM Basic Metadata}%
1258     {prism}%
1259     {http://prismstandard.org/namespaces/basic/2.1/}%
1260   \hyxmp@declare@property
1261     {complianceProfile}%
1262     {internal}%
1263     {PRISM specification compliance profile to which this document adheres}%
1264   \hyxmp@declare@property
1265     {publicationName}%
1266     {external}%
1267     {Publication name}%
1268   \hyxmp@declare@property
1269     {aggregationType}%
1270     {external}%
1271     {Publication type}%
1272   \hyxmp@declare@property
1273     {bookEdition}%
1274     {external}%
1275     {Edition of the book in which the document was published}%
1276   \hyxmp@declare@property
1277     {volume}%
1278     {external}%
1279     {Publication volume number}%
1280   \hyxmp@declare@property
1281     {number}%
1282     {external}%
1283     {Publication issue number within a volume}%
1284   \hyxmp@declare@property
1285     {pageRange}%
1286     {external}%
1287     {Page range for the document within the print version of its publication}%
1288   \hyxmp@declare@property
1289     {issn}%
1290     {external}%
1291     {ISSN for the printed publication in which the document was published}%
1292   \hyxmp@declare@property

```

```

1293         {eIssn}%
1294         {external}%
1295         {ISSN for the electronic publication in which the document was published}%
1296 \hyxmp@declare@property
1297         {isbn}%
1298         {external}%
1299         {ISBN for the publication in which the document was published}%
1300 \hyxmp@declare@property
1301         {doi}%
1302         {external}%
1303         {Digital Object Identifier for the document}%
1304 \hyxmp@declare@property
1305         [URL]
1306         {url}%
1307         {external}%
1308         {URL at which the document can be found}%
1309 \hyxmp@declare@property
1310         [Integer]
1311         {byteCount}%
1312         {internal}%
1313         {Approximate file size in octets}%
1314 \hyxmp@declare@property
1315         [Integer]
1316         {pageCount}%
1317         {internal}%
1318         {Number of pages in the print version of the document}%
1319 \hyxmp@declare@property
1320         {subtitle}%
1321         {external}%
1322         {Document's subtitle}%
1323 \hyxmp@end@ext@decl
1324 }%

```

`\hyxmp@declare@extensions` Declare all XMP extension schemata. We'll always have at least one, the XMP Media Management extensions, because we automatically generate `xmpMM:DocumentID` and `xmpMM:InstanceID` values.

```
1325 \newcommand*{\hyxmp@declare@extensions}{%
```

```
1326 \hyxmp@begin@extension@decls
```

Declare the XMP Media Management extensions (always present).

```
1327 \hyxmp@mm@extensions
```

Declare the PDF/A Identification extensions, but only when generating a PDF/A document.

```
1328 \ifHy@pdfa
```

```
1329 \hyxmp@mm@extensions
```

```
1330 \fi
```

Conditionally declare IPTC photo metadata extensions.

```
1331 \ifx\hyxmp@iptc@data\@empty
```

```
1332 \else
```

```

1333     \hyxmp@iptc@extensions
1334 \fi

    Conditionally declare PRISM basic metadata extensions.
1335 \ifx\hyxmp@prism@data\@empty
1336 \else
1337     \hyxmp@prism@extensions
1338 \fi
1339 \hyxmp@end@extension@decls
1340 }

```

3.5.12 Combining schemata into an XMP packet

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

```

1341 \begingroup
1342 \ifhyxmp@unicodetex
1343     \lccode'\!="FEFF %
1344     \lowercase{%
1345         \gdef\hyxmp@bom{!}
1346     }%
1347 \else
1348     \catcode'\^^ef=12
1349     \catcode'\^^bb=12
1350     \catcode'\^^bf=12
1351     \gdef\hyxmp@bom{^^ef^^bb^^bf}%
1352 \fi
1353 \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.

```

1354 \def\hyxmp@construct@packet{%
1355     \gdef\hyxmp+xml{%
1356         \hyxmp@add@to+xml{<?xpacket begin="\hyxmp@bom" %
1357 id="W5MOMpCehiHzreSzNTczkc9d"?>^^J%
1358 <x:xmpmeta xmlns:x="adobe:ns:meta/">^^J%
1359 __<rdf:RDF %
1360 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\hyxmp@hash">^^J%
1361 ____<rdf:Description rdf:about="">^^J%

```

Specify every namespace we can potentially use, even the ones we end up not actually using.

```

1362 _____xmlns:pdf="http://ns.adobe.com/pdf/1.3/"^^J%
1363 _____xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/"^^J%
1364 _____xmlns:dc="http://purl.org/dc/elements/1.1/"^^J%
1365 _____xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/"^^J%
1366 _____xmlns:xmp="http://ns.adobe.com/xap/1.0/"^^J%
1367 _____xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"^^J%
1368 _____xmlns:stEvt="http://ns.adobe.com/xap/1.0/sType/ResourceEvent\hyxmp@hash"^^J%
1369 _____xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/"^^J%
1370 _____xmlns:prism="http://prismstandard.org/namespaces/basic/2.1/"^^J%

```

```

1371 -----xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"^^J%
1372 -----xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"^^J%
1373 -----xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema\hyxmp@hash"^^J%
1374 -----xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property\hyxmp@hash"^^J%
1375 -----xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type\hyxmp@hash"^^J%
1376 -----xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field\hyxmp@hash">^^J%
1377  }%

```

Declare non-standard schemata.

```

1378 \hyxmp@check@iptc@data
1379 \hyxmp@check@prism@data
1380 \hyxmp@declare@extensions

```

Insert all the metadata we know how to insert.

```

1381 \hyxmp@pdf@schema
1382 \hyxmp@xmpRights@schema
1383 \hyxmp@dc@schema
1384 \hyxmp@photoshop@schema
1385 \hyxmp@xmp@basic@schema
1386 \hyxmp@pdfa@id@schema
1387 \hyxmp@mm@schema
1388 \hyxmp@iptc@schema
1389 \hyxmp@prism@schema
1390 \hyxmp@add@to+xmll{%
1391 ____</rdf:Description>^^J%
1392 __</rdf:RDF>^^J%
1393 </x:xmpmeta>^^J%
1394 \hyxmp@padding
1395 <?xpacket end="w"?>^^J%
1396  }%
1397 }

```

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.

```

1398 \newcommand*{\hyxmp@embed@packet}{%
1399 \hyxmp@construct@packet
1400 \def\hyxmp@driver{hpdfTeX}%
1401 \ifx\hyxmp@driver\Hy@driver
1402 \hyxmp@embed@packet@pdfTeX
1403 \else
1404 \def\hyxmp@driver{hLaTeX}%
1405 \ifx\hyxmp@driver\Hy@driver
1406 \hyxmp@embed@packet@LaTeX
1407 \else

```

```

1408     \def\hyxmp@driver{hdvipdfm}%
1409     \ifx\hyxmp@driver\Hy@driver
1410         \hyxmp@embed@packet@dvipdfm
1411     \else
1412         \def\hyxmp@driver{hxdetex}%
1413         \ifx\hyxmp@driver\Hy@driver
1414             \hyxmp@embed@packet@xdetex
1415         \else
1416             \ifundefined{pdfmark}{%
1417                 \PackageWarningNoLine{hyperxmp}{%
1418                     Unrecognized hyperref driver ‘\Hy@driver’.\MessageBreak
1419                     \jobname.tex’s XMP metadata will *not* be\MessageBreak
1420                     embedded in the resulting file}%
1421             }{%
1422                 \hyxmp@embed@packet@pdfmark
1423             }%
1424         \fi
1425     \fi
1426 \fi
1427 \fi
1428 }

```

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.

```

1429 \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.

```

1430 \newcommand*{\hyxmp@embed@packet@pdfTeX}{%
1431     \bgroup
1432     \ifluatex
1433     \else
1434         \pdfcompresslevel=0
1435     \fi

```

```

1436 \immediate\pdfobj \ifluatex uncompressed\fi stream attr {%
1437 /Type /Metadata
1438 /Subtype /XML
1439 }{\hyxmp@xml}%
1440 \pdfcatalog {/Metadata \the\pdflastobj\space 0 R}%
1441 \egroup
1442 }

```

3.6.2 Embedding using LuaTeX 0.85+

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

```

1443 \newcommand*{\hyxmp@embed@packet@luatex}{%
1444 \immediate\pdfextension obj uncompressed stream attr {%
1445 /Type /Metadata
1446 /Subtype /XML
1447 }{\hyxmp@xml}%
1448 \pdfextension catalog {/Metadata \the\numexpr\pdffeedback lastobj\relax\space 0 R}%
1449 }

```

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.

```

1450 \newcommand*{\hyxmp@embed@packet@pdfmark}{%
1451 \pdfmark{%
1452 pdfmark=/NamespacePush
1453 }%
1454 \pdfmark{%
1455 pdfmark=/OBJ,
1456 Raw={/_objdef \string{hyxmp@Metadata\string} /type /stream}%
1457 }%
1458 \pdfmark{%
1459 pdfmark=/PUT,
1460 Raw={\string{hyxmp@Metadata\string}
1461 2 dict begin
1462 /Type /Metadata def
1463 /Subtype /XML def
1464 currentdict
1465 end
1466 }%
1467 }%
1468 \pdfmark{%
1469 pdfmark=/PUT,
1470 Raw={\string{hyxmp@Metadata\string} (\hyxmp@xml)}%
1471 }%
1472 \pdfmark{%
1473 pdfmark=/Metadata,
1474 Raw={\string{Catalog\string} \string{hyxmp@Metadata\string}}%

```



```

1475 }%
1476 \pdfmark{%
1477     pdfmark=/NamespacePop
1478 }%
1479 }

```

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.

```

1480 \newcommand*{\hyxmp@embed@packet@dvipdfm}{%
1481     \hyxmp@string@len{\hyxmp@xml}%
1482     \special{pdf: object @hyxmp@Metadata
1483         <<
1484             /Type /Metadata
1485             /Subtype /XML
1486             /Length \the\@tempcnta
1487         >>
1488         stream^^J\hyxmp@xml endstream%
1489     }%
1490     \special{pdf: docview
1491         <<
1492             /Metadata @hyxmp@Metadata
1493         >>
1494     }%
1495 }

```

`\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.

```

1496 \newcommand*{\hyxmp@string@len}[1]{%
1497     \@tempcnta=0
1498     \expandafter\hyxmp@count@spaces#1 {} %
1499     \expandafter\hyxmp@count@non@spaces#1{}%
1500 }

```

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

```

1501 \def\hyxmp@count@spaces#1 {%
1502     \def\hyxmp@one@token{#1}%
1503     \ifx\hyxmp@one@token\empty
1504         \advance\@tempcnta by -1
1505     \else
1506         \advance\@tempcnta by 1
1507         \expandafter\hyxmp@count@spaces

```

```

1508 \fi
1509 }

```

`\hyxmp@count@non@spaces` Count the number of non-spaces in a given string. Ideally, we'd count both spaces and non-spaces but T_EX 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.

```

1510 \newcommand*{\hyxmp@count@non@spaces}[1]{%
1511   \def\hyxmp@one@token{#1}%
1512   \ifx\hyxmp@one@token\empty
1513   \else
1514     \advance\@tempcnta by 1
1515     \expandafter\hyxmp@count@non@spaces
1516   \fi
1517 }

```

3.6.5 Embedding using X_YT_EX

`\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.

```

1518 \newcommand*{\hyxmp@embed@packet@xetex}{%
1519   \special{pdf:stream @hyxmp@Metadata (\hyxmp@xml)
1520     <<
1521       /Type /Metadata
1522       /Subtype /XML
1523     >>
1524   }%
1525   \special{pdf:put @catalog
1526     <<
1527       /Metadata @hyxmp@Metadata
1528     >>
1529   }%
1530 }

```

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.

```

1531 \catcode'\\"=\hyxmp@dq@code

```

4 Help Wanted

Comma handling Ideally, `\xmpquote` should automatically replace all commas with `\xmpcomma`. Unfortunately, my T_EX 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 T_EX engines (pdfT_EX, LuaT_EX, X_YT_EX, etc.), please send me a code patch.

A Sample XMP Packet

The following is an example of a complete XMP packet as may be produced by `hyperxmp`. This packet corresponds to the metadata included in the sample L^AT_EX document presented on pages 8–9. 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/">
  <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/"
      xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/"
      xmlns:xmp="http://ns.adobe.com/xap/1.0/"
      xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"
      xmlns:stEvt="http://ns.adobe.com/xap/1.0/sType/ResourceEvent#"
      xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/"
      xmlns:prism="http://prismstandard.org/namespaces/basic/3.0/"
      xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"
      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>
          :
          [over 100 lines of boilerplate definitions not shown]
          :
        </rdf:Bag>
      </pdfaExtension:schemas>
      <pdf:Keywords>
        energy quanta, Hertz effect, quantum physics
      </pdf:Keywords>
      <pdf:Producer>pdfTeX-1.40.19</pdf:Producer>
      <pdf:PDFVersion>1.5</pdf:PDFVersion>
      <xmpRights:Marked>True</xmpRights:Marked>
      <xmpRights:WebStatement>
        http://creativecommons.org/licenses/by-nc-nd/3.0/
      </xmpRights:WebStatement>
      <dc:format>application/pdf</dc:format>
```

```

<dc:title>
  <rdf:Alt>
    <rdf:li xml:lang="en">
      On a heuristic viewpoint concerning the production
      and transformation of light
    </rdf:li>
    <rdf:li xml:lang="x-default">
      On a heuristic viewpoint concerning the production
      and transformation of light
    </rdf:li>
    <rdf:li xml:lang="de">
      Über einen die Erzeugung und Verwandlung des Lichtes
      betreffenden heuristischen Gesichtspunkt
    </rdf:li>
  </rdf:Alt>
</dc:title>
<dc:description>
  <rdf:Alt>
    <rdf:li xml:lang="en">photoelectric effect</rdf:li>
    <rdf:li xml:lang="x-default">photoelectric effect</rdf:li>
  </rdf:Alt>
</dc:description>
<dc:rights>
  <rdf:Alt>
    <rdf:li xml:lang="en">
      Copyright (C) 1905, Albert Einstein
    </rdf:li>
    <rdf:li xml:lang="x-default">
      Copyright (C) 1905, Albert Einstein
    </rdf:li>
  </rdf:Alt>
</dc:rights>
<dc:creator>
  <rdf:Seq>
    <rdf:li>Albert Einstein</rdf:li>
  </rdf:Seq>
</dc:creator>
<dc:subject>
  <rdf:Bag>
    <rdf:li>energy quanta</rdf:li>
    <rdf:li>Hertz effect</rdf:li>
    <rdf:li>quantum physics</rdf:li>
  </rdf:Bag>
</dc:subject>
<dc:date>
  <rdf:Seq>

```

```

        <rdf:li>1905-03-17</rdf:li>
    </rdf:Seq>
</dc:date>
<dc:language>
    <rdf:Bag>
        <rdf:li>en</rdf:li>
    </rdf:Bag>
</dc:language>
<dc:type>
    <rdf:Bag>
        <rdf:li>Text</rdf:li>
    </rdf:Bag>
</dc:type>
<dc:source>einstein.tex</dc:source>
<photoshop:AuthorsPosition>
    Technical Assistant, Level III
</photoshop:AuthorsPosition>
<photoshop:CaptionWriter>Scott Pakin</photoshop:CaptionWriter>
<xmp:CreateDate>2019-03-16T23:07:38-06:00</xmp:CreateDate>
<xmp:ModifyDate>2019-03-16T23:07:38-06:00</xmp:ModifyDate>
<xmp:MetadataDate>2019-03-16T23:07:38-06:00</xmp:MetadataDate>
<xmp:CreatorTool>LaTeX with hyperref package</xmp:CreatorTool>
<xmpMM:DocumentID>
    uuid:6d1ac9ec-4ff2-515a-954b-648eeb4853b0
</xmpMM:DocumentID>
<xmpMM:InstanceID>
    uuid:3e4c4182-b182-46c9-995f-754c41d13390
</xmpMM:InstanceID>
<xmpMM:VersionID>2.998e8</xmpMM:VersionID>
<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>
<prism:complianceProfile>three</prism:complianceProfile>
<prism:subtitle xml:lang="en-US">
    Putting that bum Maxwell in his place
</prism:subtitle>
<prism:publicationName xml:lang="de">

```

```

        Annalen der Physik
    </prism:publicationName>
    <prism:aggregationType>journal</prism:aggregationType>
    <prism:volume>322</prism:volume>
    <prism:number>6</prism:number>
    <prism:pageRange>132-148</prism:pageRange>
    <prism:issn>0003-3804</prism:issn>
    <prism:eIssn>1521-3889</prism:eIssn>
    <prism:doi>10.1002/andp.19053220607</prism:doi>
    <prism:url>
        http://www.physik.uni-augsburg.de/annalen/history/einstein-papers/190517132-
    </prism:url>
    <prism:byteCount>56884</prism:byteCount>
    <prism:pageCount>17</prism:pageCount>
</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 Digital Enterprise Alliance, Inc. *Publishing Requirements for Industry Standard Metadata, Version 3.0: PRISM Basic Metadata Specification*, October 12, 2012. Available from http://www.prismstandard.org/specifications/3.0/PRISM_Basic_Metadata_3.0.htm.
- [8] International Digital Enterprise Alliance, Inc. *Publishing Requirements for Industry Standard Metadata, Version 3.0: PRISM Controlled Vocabularies Specification*, October 4, 2012. Available from http://www.prismstandard.org/specifications/3.0/PRISM_CV_Spec_3.0.pdf.
- [9] 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.
- [10] Internet Assigned Numbers Authority. Language subtag registry, January 11, 2011. Available from <http://www.iana.org/assignments/language-subtag-registry>.
- [11] 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>.
- [12] PDF/A Competence Center, Berlin, Germany. *TechNote 0008: Pre-defined XMP Properties in PDF/A-1*, March 20, 2008. Available from http://www.pdfa.org/wp-content/uploads/2011/08/tn0008_predefined_xmp_properties_in_pdfa-1_2008-03-20.pdf.
- [13] 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.
- [14] 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		and $\langle BF \rangle$ to “letter”. Thanks
General: Initial version	1	to Daniel Schömer for the bug
v1.1		report
\hyxmp@construct@packet:	v1.2	61
Explicitly set the category	General: Added support for the	
codes of characters $\langle EF \rangle$, $\langle BB \rangle$,	X _Y TeX backend (xdvipdfmx)	1

Added support for the Photoshop schema	1	\hyxmp@comma: Added this macro	27
Made the package compatible with <code>ngerman</code> . Thanks to Tobias Mueller for the bug report.	16	\hyxmp@construct@packet: Modified by Heiko Oberdiek to use an appropriate BOM representation via \hyxmp@bom	61
v1.3		\hyxmp@crap@convert: Added by Heiko Oberdiek	38
\hyxmp@reencode: 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	34	\hyxmp@crap@test: Added by Heiko Oberdiek	38
General: Introduced the <code>pdfmetalang</code> package option, which enables an author to specify the language in which he wrote the document's metadata	25	\hyxmp@dc@schema: Added support for <code>dc:language</code> and <code>dc:source</code> .	49
v1.4		\hyxmp@is@unicode: Added by Heiko Oberdiek	35
\hyxmp@mm@schema: Renamed the <code>xapMM</code> namespace prefix to <code>xmpMM</code>	50	\hyxmp@list@to+xml: Modified by Heiko Oberdiek to use the new Unicode-processing macros . .	48
\hyxmp@rdf@dc: Included metadata in the <code>x-default</code> language regardless of the specified metadata language	47	\hyxmp@photoshop@schema: Simplified using \hyxmp@add@simple	52
\hyxmp@xmpRights@schema: Renamed the <code>xapRights</code> namespace prefix to <code>xmpRights</code>	49	\hyxmp@reencode: Replaced with an empty macro by Heiko Oberdiek	34
v1.5		\hyxmp@skiptorelax: Added by Heiko Oberdiek	38
General: Made the XMP inclusion more robust. Thanks to Heiko Oberdiek for the bug report and suggested modifications. .	16	\hyxmp@skipzeros: Added by Heiko Oberdiek	37
v2.0		\hyxmp@toxml: Added by Heiko Oberdiek	36
\ProcessKeyvalOptions: Added this macro	23	Escaped parentheses written with <code>pdfmarks</code> to prevent <code>dvips</code> from line-wrapping the XMP packet	36
\XMPTruncateList: Added this macro	28	\hyxmp@toxml@unicodetex: Added by Heiko Oberdiek	36
\hyxmp@ProcessKeyvalOptions: Added this macro	23	\hyxmp@xetex@crap: Added by Heiko Oberdiek	37
\hyxmp@SpaceOther: Added by Heiko Oberdiek	38	\hyxmp+xmlify: Completely rewritten by Heiko Oberdiek to better support Unicode-enabled TeX programs	34
\hyxmp@add@simple: Added this macro	45	\hyxmp@xmp@basic@schema: Added this macro	51
\hyxmp@add@to+xml: Updated also to replace commas	44	\hyxmp@xmpRights@schema: 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 .	49
\hyxmp@bom: Added by Heiko Oberdiek	61	\hyxmp@zero: Added by Heiko Oberdiek	39

\ifhyxmp@unicodetex: Added by Heiko Oberdiek	34	\@pdflang as \@empty when hyperref has set it to \relax . .	25
\xmpcomma: Added this macro . . .	27		
\xmpquote: Added this macro . . .	28	v2.3b	
General: Added support for the XMP Basic schema and miscellaneous other bits of metadata	1	\XMPTruncateList: Made all definitions local to avoid spurious Too many unprocessed floats errors when running with memoir . .	28
Heiko Oberdiek's major rewrite of the code to better support native-Unicode T _E X implementations (X _Y T _E X and LuaT _E X)	1	v2.4	
New \AtBeginDocument code from Heiko Oberdiek to properly encode \@pdfmetalang	25	\hyxmp@add@simple@var: Added this macro	46
v2.1		\hyxmp@create@uuid: Modified this macro to produce a proper version 4 (random or pseudorandom) UUID	42
\hypersetup: Added this macro .	24	\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	49
\hyxmp@hypersetup: Added this macro	24	\hyxmp@parse@time: Added this macro	29
\hyxmp@redefine@Hyp: Added this macro	22	\hyxmp@parse@tz: Added this macro	29
General: Enabled hyperxmp and hyperref to be loaded in either order. This addresses a bug report by Yury Donskoy	22	\hyxmp@parse@tz@char: Added this macro	29
v2.2		\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	45
\hyxmp@iptc@extensions: Added this macro to support PDF/A generation	57	\hyxmp@pdf@to@xmp@date: Added this macro	29
\hyxmp@iptc@schema: Added this macro	53	\hyxmp@pdfa@id@schema: Added this macro	52
\hyxmp@list@to@lines: Added this macro	52	\hyxmp@today: Modified the code to parse the time and timezone from \pdfcreationdate, as proposed by Florian Breitwieser	32
\xmpcomma: Changed the default from \relax to an ordinary comma	27	\hyxmp@today@define: Added this macro	31
\xmplinesep: Added this macro .	52	\hyxmp@xmp@to@pdf@date: Added this macro	30
General: Added support for the IPTC Photo Metadata schema .	1	\xmptilde: Added this macro . .	28
v2.3		General: Added support for the PDF/A Identification schema, as	
\hyxmp@iptc@extensions: Gave the lptc4xmpCore:CreatorContactInfo fields a unique pdfaType:prefix to better support conversion of the document to PDF/A	57		
v2.3a			
General: Bug fix: Redefine			

requested by Florian Breitwieser	1	Introduced the <code>pdftype</code> package option, which enables an author to specify the type of document being produced	1
v2.5		v3.0	
<code>\hyxmp@add@to@xml</code> : Updated also to replace underscores and to modify only the text being added, not the already-modified text	44	<code>\hyxmp@embed@packet@luatex</code> : Added this macro	64
<code>\hyxmp@textunderscore</code> : Added this macro	17	<code>\hyxmp@today@define</code> : Modified to accept the name of a macro to define	31
<code>\hyxmp@uscore</code> : Added this macro	27	<code>\hyxmp@xmp@basic@schema</code> : Made the XMP <code>xmp:CreateDate</code> , <code>xmp:ModifyDate</code> , and <code>xmp:MetadataDate</code> match the PDF <code>CreationDate</code>	51
General: Enabled “_” to work within email addresses, as requested by Leonid Sinev	1	General: Made the code compatible with LuaTeX 0.85. Thanks to Robert Schlicht, Leonid Sinev, and David Carlisle for bug reports and to Leonid Sinev for helping test the new <code>hyperxmp</code> code	1
v2.6		v3.1	
General: Added support for a new <code>pdfdate</code> key to explicitly specify the document date (and optionally time)	1	<code>\hyxmp@embed@packet@luatex</code> : Updated to use <code>\pdfextension</code> <code>obj uncompressed</code> as suggested by Hans Hagen	64
v2.7		<code>\hyxmp@embed@packet@pdftex</code> : Leave the XMP packet—and only the XMP packet—uncompressed in both <code>pdfTeX</code> and pre-0.85 <code>LuaTeX</code>	63
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	26	v3.2	
v2.8		<code>\hyxmp@as@pdf@date</code> : Added this macro	30
<code>\hyxmp@add@to@xml</code> : Corrected inadvertent lowercasing of non-Latin characters when run under <code>X_qLaTeX</code> or <code>LuaLaTeX</code> (bug reported by Leonid Sinev)	44	<code>\hyxmp@as@xmp@date</code> : Added this macro	28
v2.9		<code>\hyxmp@today@define</code> : Modified to include hours and minutes . . .	31
<code>\hyxmp@iptc@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> . . .	53	<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	51
<code>\hyxmp@pdfa@id@schema</code> : Let the author specify the PDF/A part and conformance IDs, as requested by Leonid Sinev . . .	52	v3.3	
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>\@pdfsource</code> : Added this macro and the corresponding	

pdfsource option, at Niklas Beisert's request	19	\hyxmp@add@simple@lang: Added this macro	46
\XMPLangAlt: Added this macro based on a request—and some code—by Niklas Beisert to support metadata expressed in multiple languages	40	\hyxmp@begin@ext@decl: Added this macro	56
\hyxmp@rdf@dc: Bug fix: Output the metadata language as correct XML even when hyperref is loaded with the unicode option	47	\hyxmp@declare@field: Replaced \hyxmp@declare@resource with this macro	56
General: Don't overwrite an existing pdfmetalang with pdflang or x-default. This addresses a bug report by Niklas Beisert	25	\hyxmp@declare@property: Added this macro	56
v3.4		\hyxmp@end@ext@decl: Added this macro	56
\hyxmp@seed@string: Correctly handle an author field of all spaces. Bug reported by Gaëtan Leurent	43	\hyxmp@iptc@extensions: Moved the header code from here into \hyxmp@begin@extension@decls and the trailer code from here into \hyxmp@end@extension@decls	57
General: Use ifmtarg to test for empty arguments, including non-empty but all spaces	1	Rewrote to more closely honor the XMP specification	57
v3.5		\hyxmp@iptc@schema: Moved the definition of \hyxmp@iptc@data from here into \hyxmp@check@iptc@data	53
\hyxmp@DocumentID: Added the pdfdocumentid option, at Michael Osipov's request	19	Renamed this macro to \hyxmp@iptc@schema from \hyxmp@photometa@schema	53
\hyxmp@InstanceID: Added the pdfinstanceid option, at Michael Osipov's request	19	Rewrote this macro entirely to correct the use of fields within a structure	53
\hyxmp@mm@schema: Generate \hyxmp@DocumentID and \hyxmp@InstanceID only if the document does not already define these using the pdfdocumentid and pdfinstanceid options	50	\hyxmp@mm@extensions: Added this macro	57
\hyxmp@seed@string: Seed with the TeX timestamp in addition to the document-specified timestamp	43	\hyxmp@mm@schema: Include xmpMM:VersionID in the XMP packet	50
v4.0		\hyxmp@pdfa@id@extensions: Added this macro	57
\XMPTruncateList: Deprecated this macro	28	\hyxmp@prism@extensions: Added this macro	59
		\hyxmp@prism@schema: Added this macro	54
		\hyxmp@suppress@pdf@metadata: Added this macro	21
		General: Include all metadata within a single rdf:Description block	1

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		
\#	790, 1015	\@pdfcontactemail 116, 189, 1054, 1090
\&	520, 552, 1014	\@pdfcontactphone 114, 190, 1053, 1089
\,	783	\@pdfcontactpostcode 110, 191, 1050, 1087
\@author	251, 252	\@pdfcontactregion 108, 192, 1049, 1086
\@baseurl	180, 1000	\@pdfcontacturl 118, 193, 1055, 1091
\@elt 299, <u>905</u> , 1026, <u>1031</u>		\@pdfcopyright 46, 194, 924, 941, 947
\@elt@first	<u>1024</u>	\@pdfcreationdate 195, 977, 981
\@elt@rest	1026, <u>1028</u>	\@pdfcreator 999
\@firstofone	599	\@pdfdatetime 24, 196, 240, 242
\@firstoftwo	511	\@pdfdoi 92, 197, 1077, 1098
\@gobble	309, 596	\@pdfdoi 92, 197, 1077, 1098
\@gobbletwo 383, 398, 400		\@pdfdoi 92, 197, 1077, 1098
\@hyxmp@gobbletwo	383, 396	\@pdfdoi 92, 197, 1077, 1098
\@ifmtarg	16, 848	\@pdfdoi 92, 197, 1077, 1098
\@ifmtargexp 16, 244, 250, 860, 895, 955, 956	\@pdfdoi 92, 197, 1077, 1098
\@ifnextchar	843	\@pdfdoi 92, 197, 1077, 1098
\@ifnotmtarg 17, 837, 846		\@pdfdoi 92, 197, 1077, 1098
\@ifnotmtargexp 16, 245, 251, 820, 1019		\@pdfdoi 92, 197, 1077, 1098
\@pdfaformance 60, 1010	\@pdfdoi 92, 197, 1077, 1098
\@pdfapart	<u>58</u> , 1009	\@pdfdoi 92, 197, 1077, 1098
\@pdfauthor	<u>135</u> , 181, 250, 755, 763	\@pdfdoi 92, 197, 1077, 1098
\@pdfauthortitle	<u>52</u> , 182, 1003, 1004	\@pdfdoi 92, 197, 1077, 1098
\@pdfbookedition	<u>84</u> , 183, 1070, 1096	\@pdfdoi 92, 197, 1077, 1098
\@pdfbytes	<u>74</u> , 184, 1079, 1097	\@pdfdoi 92, 197, 1077, 1098
\@pdfcaptionwriter	<u>54</u> , 185, 1003, 1005	\@pdfdoi 92, 197, 1077, 1098
\@pdfcontactaddress	<u>98</u> , 186, 1047, 1084	\@pdfdoi 92, 197, 1077, 1098
\@pdfcontactcity	<u>106</u> , 187, 1048, 1085	\@pdfdoi 92, 197, 1077, 1098
\@pdfcontactcountry	<u>112</u> , 188, 1051, 1088	\@pdfdoi 92, 197, 1077, 1098
		\@pdfpublication 70, 209, 1068, 1105
		\@pdfpubtype 72, 210, 1069, 1106
		\@pdfsource 62, 930, 932
		\@pdfsubject 211, 923
		\@pdfsubtitle 96, 212, 1067, 1107
		\@pdftitle 213, 244, 755, 763, 922
		\@pdftype 48, 214, 929
		\@pdfurl 94, 215, 1078, 1108
		\@pdfversionid 68, 216, 959
		\@pdfvolumenum 86, 217, 1071, 1109
		\@secondoftwo 513
		\@tempwafalse 860, 895
		\@tempwattrue 860, 862, 895, 897
		\@title 245, 246
		\^ 287, 291, 456, 783, 784, 1348–1350
		_ 782, 784
		\~ 296, 590, 591
		_ 556, 591, 782
		A
		\and <u>135</u>
		ASCII 17, 35
		\AtBeginDocument 227
		\AtEndDocument 5
		\AtEndDvi 8
		atendddvi 16
		Author 9, 10, 21
		B
		Bag 73
		baseurl (option) 4, 6, 13, 17, 20, 51
		BOM 61, 72

C		G	\hyxmp@alt@description
CiAdrCity	2	Ghostscript	654, 664
CiAdrCtry	2	gitver	\hyxmp@alt@rights
CiAdrExtadr	2		654, 665
CiAdrPcode	2	H	\hyxmp@alt@title 654, 663
CiAdrRegion	2	\Hy@driver	\hyxmp@and 135
CiEmailWork	2	. 4, 1401, 1405,	\hyxmp@append@hex
CiTelWork	3	1409, 1413, 1418	. 703, 722–724, 728
CiUrlWork	3	\Hy@unicodetfalse . 27, 38	\hyxmp@append@hex@iii
CreationDate	74	hyperref 1, 4, 5, 7, 8, 10,	. 721, 727, 737, 748
		13, 17, 21, 22, 24–	\hyxmp@append@hex@iv
D		26, 45, 62–64, 73–75	. 726, 732,
Date	32	\hypersetup 173, 246, 252	733, 735, 750–752
\day	416, 417, 419	hyperxmp 1, 2, 4, 5, 7,	\hyxmp@as@pdf@date . 343
dc:creator	2, 10, 49, 74	8, 10, 12–18, 21,	\hyxmp@as@xmp@date
dc:date	2, 49	22, 25–28, 34, 41,	30, 41, 315, 981, 991
dc:description		45, 54, 63, 67, 73, 74	\hyxmp@at@end . 3, 256
. . . 3, 10, 39, 49, 74		\hyxmp@is@unicode . 500	\hyxmp@begin@ext@decl
dc:format	2	\hyxmp@add@simple 1124, 1161,
dc:language	3, 49, 72, 73 812, 815,	1182, 1198, 1256
dc:rights	2, 15, 39, 49	819, 921, 932,	\hyxmp@begin@extension@decls
dc:source	2, 49, 72	949, 951, 957– 1112, 1326
dc:subject	2, 49	959, 975, 978,	\hyxmp@big@prime . .
dc:title	3, 10, 39, 49, 74	980, 985, 988,	. 677, 680, 690, 700
dc:type	3	990, 995, 997,	\hyxmp@big@prime@ii
DCMI	7	999, 1000, 1004, 677, 699
\define@key	25,	1005, 1009, 1010,	\hyxmp@bom . . 1341, 1356
36, 47, 49, 51, 53,		1048–1051, 1065,	\hyxmp@check@iptc@data
55, 57, 59, 61, 63,		1069, 1071–1080 1082, 1378
65, 67, 69, 71, 73,		\hyxmp@add@simple@lang	\hyxmp@check@prism@data
75, 77, 79, 81, 83,	 836, 1094, 1379
85, 87, 89, 91, 93,		1067, 1068, 1070	\hyxmp@comma
95, 97, 99, 107,		\hyxmp@add@simple@lang@i	. 100, 136, 157, 286
109, 111, 113,	 839, 842	\hyxmp@commas@to@list
115, 117, 119,		\hyxmp@add@simple@lang@ii	. 270, 306, 906, 1033
123, 135, 156, 658	 843, 845	\hyxmp@commas@to@list@i
\do	660, 667, 881	\hyxmp@add@simple@var 272, 274
DOI	2, 6, 20 807, 808, 827	\hyxmp@concat@metadata
dvipdf (option)	64	\hyxmp@add@to@xml 178
dvipdfm	65	. 769, 795, 822,	\hyxmp@construct@packet
dvips (option)	64	831, 849, 853, 1354, 1399
dvips	9, 36, 72	867, 874, 878,	\hyxmp@count@non@spaces
dvipsone (option)	64	883, 888, 900, 1499, 1510
dviwindo (option)	64	908, 914, 1021,	\hyxmp@count@spaces
		1025, 1029, 1035, 1498, 1501
E		1042, 1057, 1113,	\hyxmp@crap@convert
\EdefEscapeHex	479, 492	1119, 1125, 1135, 582, 616
\EdefUnescapeHex	496	1142, 1152, 1207,	\hyxmp@crap@result .
\EdefUnescapeString	466	1246, 1356, 1390 572, 608
ETX	27, 34		\hyxmp@crap@test 579, 604

\hyxmp@create@uuid .	\hyxmp@end@extension@decls	\hyxmp@parse@time ..
.... 730, 758, 767 1118, 1339 324, 326
\hyxmp@createdate ..	\hyxmp@extra@indent	\hyxmp@parse@tz ...
..... 961, 975, 818, 333, 336, 340
978, 985, 988, 995	823, 832, 1022, 1046	\hyxmp@parse@tz@char
\hyxmp@cur@lang 660, 668	\hyxmp@find@metadata 328, 330
\hyxmp@dc@schema 178, 258	\hyxmp@pdf@schema ..
..... 920, 1383	\hyxmp@first@char .. 313 806, 1381
\hyxmp@declare@extensions	\hyxmp@first@char@i	\hyxmp@pdf@to@xmp@date
..... 1325, 1380 313, 316, 344 317, 322,
\hyxmp@declare@field	\hyxmp@gobbletwo ... 383	440, 443, 966, 969
..... 1151,	\hyxmp@hash 789, 1360,	\hyxmp@pdfa@id@extensions
1222, 1225, 1228,	1368, 1373–1376 1181
1231, 1234,	\hyxmp@Hyp@pdfauthor 129	\hyxmp@pdfa@id@schema
1237, 1240, 1243	\hyxmp@Hyp@pdfkeywords 1007, 1386
\hyxmp@declare@property 150	\hyxmp@pdfauthor ...
..... 1141,	\hyxmp@hypersetup .. 173 126, 135, 925
1165, 1170, 1175,	\hyxmp@InstanceID ..	\hyxmp@pdfkeywords .
1186, 1191, 1202,	. 66, 760, 956, 958 126, 156, 926
1260, 1264, 1268,	\hyxmp@iptc@data ...	\hyxmp@pdfstringdef
1272, 1276, 1280,	.. 1040, 1083, 1331 18, 29,
1284, 1288, 1292,	\hyxmp@iptc@extensions	40, 47, 49, 51, 53,
1296, 1300, 1304, 1197, 1333	55, 57, 59, 61, 63,
1309, 1314, 1319	\hyxmp@iptc@schema .	65, 67, 69, 71, 73,
\hyxmp@def@DocumentID 1039, 1388	75, 77, 79, 81, 83,
..... 754, 955	\hyxmp@is@unicode ..	85, 87, 89, 91, 93,
\hyxmp@def@InstanceID 468, 485, 500	95, 97, 102, 107,
..... 760, 956	\hyxmp@LA@accept ...	109, 111, 113,
\hyxmp@define@createdate 657, 663–665	115, 117, 119, 659
..... 961, 973	\hyxmp@legal 936	\hyxmp@photoshop@data
\hyxmp@DocumentID ..	\hyxmp@list 906, 912, 1033, 1034 1002
.. 64, 754, 955, 957	\hyxmp@list@to@lines	\hyxmp@photoshop@schema
\hyxmp@dq@code . 1, 1531 1018, 1002, 1384
\hyxmp@driver .. 3, 1398	1047, 1053–1055	\hyxmp@prism@data ..
\hyxmp@embed@packet 259, 1398	.. 1063, 1095, 1335
..... 1410, 1480	\hyxmp@list@to@xml .	\hyxmp@prism@extensions
\hyxmp@embed@packet@dvi 894, 925–929 1255, 1337
..... 1406, 1443	\hyxmp@mm@extensions	\hyxmp@prism@schema
\hyxmp@embed@packet@luatex	.. 1160, 1327, 1329 1062, 1389
..... 1422, 1450	\hyxmp@mm@schema ...	\hyxmp@ProcessKeyvalOptions
\hyxmp@embed@packet@pdfmark 954, 1387 168
..... 1402, 1430	\hyxmp@modulo@a 671,	\hyxmp@rand@num 696,
\hyxmp@embed@packet@pdftex	690, 700, 706, 741	705, 740, 757, 766
..... 1414, 1518	\hyxmp@new@xml 785, 786	\hyxmp@rdf@dc
\hyxmp@end@ext@decl	\hyxmp@num 616 859, 922–924
..... 1134,	\hyxmp@one@token ...	\hyxmp@redefine@Hyp
1179, 1195, 1323	679, 683, 1502, 128, 170, 175
\hyxmp@end@ext@decl	1503, 1511, 1512	\hyxmp@reencode ... 461
\hyxmp@end@ext@decl	\hyxmp@padding 793, 1394	\hyxmp@rights
		. 936, 939, 943, 945

<code>\hyxmp@seed@rng</code> . . .	1439, 1447, 1470,	<code>\intcalcDiv</code> 621, 628, 635
. 679, 756, 765	1481, 1488, 1519	<code>\intcalcMod</code> 623, 630, 637
<code>\hyxmp@seed@rng@i</code> . .	<code>\hyxmp@xmllified</code> 464,	IPTC 12, 20,
. 681, 683	823, 832, 839,	43, 53, 54, 57, 60, 73
<code>\hyxmp@seed@string</code> .	850, 854, 866,	<code>lptc4xmpCore:ContactInfo</code>
. 754, 760	875, 884, 906, 1033 53, 58
<code>\hyxmp@set@rand@num</code>	<code>\hyxmp@xmllify</code> . . 239,	<code>lptc4xmpCore:CreatorContactInfo</code>
. 696, 704, 739	464, 821, 830, 2, 3, 53, 73
<code>\hyxmp@skiptorelax</code> .	838, 847, 865,	ISBN 2, 6, 20
. 609, 615	873, 882, 905, 1032	ISO 5, 6, 14, 18, 39
<code>\hyxmp@skipzeros</code> . . . 567	<code>\hyxmp@xmp@basic@schema</code>	ISSN 2, 6, 19
<code>\hyxmp@SpaceOther</code> 972, 1385	
. 576, 589	<code>\hyxmp@xmp@to@pdf@date</code>	J
<code>\hyxmp@string@len</code> 347, 350	<code>\jobname</code> . . . 62, 221,
. 1481, 1496	<code>\hyxmp@xmp@to@pdf@date@i</code>	263, 755, 763, 1419
<code>\hyxmp@sublist</code> 351, 353	
. . 275, 276, 279, 280	<code>\hyxmp@xmp@to@pdf@date@ii</code>	K
<code>\hyxmp@suppress@pdf@metadata</code> 356, 359	<code>keeppdfinfo</code> (option) 12, 21
. 120, 257	<code>\hyxmp@xmp@to@pdf@date@iii</code>	<code>Keywords</code> 9, 10, 45
<code>\hyxmp@temp@list</code> . . . 299 362, 365	<code>\KV@Hyp@pdfauthor</code> . . 135
<code>\hyxmp@temp@str</code> . . . 299	<code>\hyxmp@xmp@to@pdf@date@iv</code>	<code>\KV@Hyp@pdfkeywords</code> 156
<code>\hyxmp@text</code> 368, 371	<code>kvoptions</code> 17, 23
. . 464, 542, 572, 616	<code>\hyxmp@xmp@to@pdf@date@v</code>	
<code>\hyxmp@textunderscore</code> 374, 377	L
. 18	<code>\hyxmp@xmp@to@pdf@date@vi</code>	<code>LF</code> 52
<code>\hyxmp@today</code> 380, 384	<code>LuaL^AT_EX</code> 9, 12,
. . 242, 436, 763, 927	<code>\hyxmp@xmp@to@pdf@date@vii</code>	32, 33, 45, 50, 51, 74
<code>\hyxmp@today@define</code> 387, 390, 400	<code>LuaT_EX</code> 34,
. . 409, 438, 761, 964	<code>\hyxmp@xmp@to@pdf@date@viii</code>	37, 63, 64, 66, 73, 74
<code>\hyxmp@toxml</code> . . . 494, 517 403, 406	
<code>\hyxmp@toxml@unicodetex</code>	<code>\hyxmp@xmpRights@schema</code>	M
. 482, 542 935, 1382	<code>memoir</code> 73
<code>\hyxmp@trimb</code> . . . 449, 452	<code>\hyxmp@zero</code> 625,	<code>Metadata</code> 9, 62, 66
<code>\hyxmp@trimc</code> . . . 452, 453	632, 639, 645, 650	<code>\month</code> 411, 412, 414
<code>\hyxmp@trimspaces</code> . .		
. 279, 445	I	N
<code>\hyxmp@try</code> 572	IETF 5	<code>NAK</code> 17, 27, 34
<code>\hyxmp@unicodetexfalse</code>	<code>\if@tempswa</code> . . . 864, 899	<code>nativepdf</code> (option) . . . 64
. 455	<code>\iffalse</code> 859, 894	<code>\newif</code> 455
<code>\hyxmp@unicodetextrue</code>	<code>\ifHy@pdfa</code> 922,	<code>\next</code> 28, 39, 274, 683
. 455	923, 925, 1008, 1328	<code>ngerman</code> 16, 72
<code>\hyxmp@uscore</code> . . . 20, 290	<code>\ifhyxmp@unicodetex</code>	<code>\number</code> 619, 621, 623,
<code>\hyxmp@value</code> . . . 659, 866 455, 467, 772, 1342	628, 630, 635, 637
<code>\hyxmp@x@default</code> . . .	<code>ifluatex</code> 63	<code>\numexpr</code> 1448
. . 234, 805, 871, 879	<code>\ifluatex</code> 1432, 1436	
<code>\hyxmp@xetex@crap</code> . .	<code>ifmtarg</code> 17, 75	O
. 473, 572	<code>ifxetex</code> 17	<code>options</code>
<code>\hyxmp+xml</code>	<code>\ifxetex</code> 472	<code>baseurl</code>
786, 793, 1354,	<code>Info</code> 9, 10, 12, 21	4, 6, 13, 17, 20, 51
	<code>intcalc</code> 17	<code>dvipdf</code> 64

dvips	64	pdfpagerange	5, 6, 15, 16	pdfcaptionwriter (option)	4, 5
dvipsone	64	pdfproducer	4, 17	\pdfcatalog	1440
dviwindo	64	pdfpublication	5, 6	\pdfcompresslevel	1434
keeppdfinfo	12, 21	pdfpubtype	5, 6	pdfcontactaddress (option)	4, 5, 12
nativepdf	64	pdfsource	5, 7, 75	pdfcontactcity (option)	4, 5
pdfa	7, 74	pdfsubject	4, 10, 17, 49	pdfcontactcountry (option)	4, 5
pdfaconformance	4, 7, 8, 52	pdfsubtitle	5	pdfcontactemail (option)	4, 5
pdfapart	4, 7, 8, 52	pdftitle	4, 5, 10, 14, 17, 26, 49, 74	pdfcontactphone (option)	4, 5
pdfauthor	4, 10, 13, 14, 17, 22, 26, 49, 74	pdftype	5, 7, 74	pdfcontactpostcode (option)	4, 5
pdfauthortitle	4, 5, 13	pdfurl	5, 6	pdfcontactregion (option)	4, 5
pdfbookedition	4, 6	pdfversionid	5, 6, 50	pdfcontacturl (option)	4, 5, 13
pdfbytes	4, 7	pdfvolumenum	5, 6	pdfcopyright (option)	4, 5, 49, 72
pdfcaptionwriter	4, 5	ps2pdf	64	pdfcreationdate (option)	4, 7, 74
pdfcontactaddress	4, 5, 12	textures	64	\pdfcreationdate	440, 966
pdfcontactcity	4, 5	unicode	13, 75	pdfdate (option)	4, 7, 14, 18, 43, 74
pdfcontactcountry	4, 5	vtexpdfmark	64	PDFDocEncoding	22, 34, 35
pdfcontactemail	4, 5			pdfdocumentid (option)	4, 6, 75
pdfcontactphone	4, 5			pdfdoi (option)	4, 6
pdfcontactpostcode	4, 5			pdf ISSN (option)	4, 6
pdfcontactregion	4, 5			pdfescape	17
pdfcontacturl	4, 5, 13			\pdfextension	1444, 1448
pdfcopyright	4, 5, 49, 72			\pdffeedback	443, 969, 1448
pdfcreationdate	4, 7, 74			pdfinstanceid (option)	4, 6, 75
pdfdate	4, 7, 14, 18, 43, 74			pdfisbn (option)	4, 6
pdfdocumentid	4, 6, 75			pdf ISSN (option)	4, 6
pdfdoi	4, 6			pdfissuenum (option)	4, 6
pdf ISSN	4, 6			pdfkeywords (option)	4, 10, 13, 17, 22, 49
pdfisbn	4, 6			pdflang (option)	4–6, 14, 17, 25, 39, 49, 75
pdf ISSN	4, 6			\pdflastobj	1440
pdfissuenum	4, 6			pdf ^A T _E X	4, 9, 12, 32, 45, 50
pdfkeywords	4, 10, 13, 17, 22, 49				
pdflang	4–6, 14, 17, 25, 39, 49, 75				
pdflicenseurl	4, 5, 13, 49, 72				
pdfmark	64				
pdfmetadate	4, 7, 18, 74				
pdfmetalang	4, 5, 14, 39, 72, 75				
pdfmoddate	4, 7, 74				
pdfnumpages	6, 7, 15, 16				
pdfpagecount	4				

pdflicenseurl (option) .	prism:issn 2	U
... 4, 5, 13, 49, 72	prism:number 2	Unicode . . 13, 17, 34–
pdfmark (option) 64	prism:pageCount 3	38, 48, 53, 61, 66, 72
\pdfmark 1451,	prism:pageRange 3	unicode (option) . . 13, 75
1454, 1458,	prism:publicationName . 3	URL 2, 3, 5, 6, 13,
1468, 1472, 1476	prism:subtitle 3	18, 20, 21, 49–51, 53
pdfmetadate (option) .	prism:url 3	UTF-16BE 35
... 4, 7, 18, 74	prism:volume 3	UTF-32BE 34, 35
pdfmetalang (option) .	\ProcessKeyvalOptions	UTF-8 35
4, 5, 14, 39, 72, 75 168	UUID 3, 5,
\pdfminorversion . . 812	Producer 45	6, 19, 40, 42, 43, 73
pdfmoddate (option) .	ps2pdf (option) 64	
... 4, 7, 74		V
pdfnumpages (option)	Q	\vfuzz 453
... 6, 7, 15, 16	\Q 445, 454	vtexpdfmark (option) . 64
\pdfobj 1436		
pdfpagecount (option) . 4	R	X
pdfpagerange (option)	RDF 48	\x 572
... 5, 6, 15, 16	rdf:Description 75	xdvipdfmx 12, 66
pdfproducer (option) 4, 17	rdf:li 2	X _q LaTeX 9,
pdfpublication (option)	rdf:Seq 2	12, 32, 45, 50, 74
... 5, 6	\renewcommand 169	X _q TeX 17, 34, 37, 66, 71, 73
pdfpubtype (option) . 5, 6	\RequirePackage . . .	XML 1, 2,
pdfsource (option) 5, 7, 75	... 7, 10–15, 1429	12, 26, 34–37, 43–
\pdfstringdef 21		48, 52, 53, 55, 61, 75
pdfsubject (option) . .	S	XMP 1, 2, 4, 5, 7,
... 4, 10, 17, 49	\SE->pdfdoc@03 462	9–18, 21, 25–30,
pdfsubtitle (option) . . 5	\SE->pdfdoc@15 463	32, 33, 36, 39, 40,
pdfTeX 7, 16, 36, 63, 66, 74	\setkeys 669	43–46, 48–51, 54,
pdftitle (option) 4, 5,	\special 1482,	57, 60, 63–67, 72–75
10, 14, 17, 26, 49, 74	1490, 1519, 1525	xmp:BaseURL 2
pdftype (option) . 5, 7, 74	stringenc 17	xmp:CreateDate . 2, 50, 74
pdfurl (option) 5, 6	\StringEncodingConvert	xmp:CreatorTool 3
\pdfvariable 815 469,	xmp:MetadataDate 2, 50, 74
pdfversionid (option) .	475, 486, 489, 584	xmp:ModifyDate 2, 50, 74
... 5, 6, 50	Subject 9, 10, 21	\xmpcomma 100,
pdfvolumenum (option)		103, 135, 156, 285
... 5, 6	T	xmpincl 4
photoshop:AuthorsPosition	TeX 14, 15,	\XMLLangAlt 666
... 3, 52	31, 32, 34, 36, 37,	\xmplinesep
photoshop:CaptionWriter	40, 41, 43, 50, 65, 66	... 1013, 1029, 1052
... 3, 52	texdate 14	xmpMM:DocumentID .
PI 43	Text 56	... 3, 40, 50, 60
PRISM . . . 6, 54, 55, 59, 61	\textunderscore . . .	xmpMM:InstanceId . .
prism:aggregationType . 3	... 19, 20, 22	... 3, 40, 50, 60
prism:bookEdition 2	textures (option) 64	xmpMM:VersionID 3, 50, 75
prism:byteCount 2	\time 421, 429	\xmpquote 101,
prism:doi 2	Title 9, 10, 21	104, 135, 156, 294
prism:elssn 2	totpages 15, 16	xmpRights:Marked 2, 49, 72
prism:isbn 2		

xmpRights:WebStatement	\xmptilde	<u>295</u>		Y
..... 3, 49, 72	\XMPTruncateList	...	<u>299</u>	\year 410