

# The hyperxmp package\*

Scott Pakin  
scott+hyxmp@pakin.org

November 18, 2020

## Abstract

`hyperxmp` makes it easy for an author to include XMP metadata in a PDF document produced by  $\text{\LaTeX}$ . `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 [5]—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  $\text{\LaTeX}$  document with three authors—Jack Napier, Edward Nigma, and Harvey Dent—named in the  $\text{\LaTeX}$  source in the usual way: “`\author{Jack Napier \and Edward Nigma \and Harvey Dent}`”. With `hyperxmp`, 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>
```

---

\*This document corresponds to `hyperxmp` v5.8, dated 2020/11/18.

```

    <rdf:li>Harvey Dent</rdf:li>
  </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 computer applications to identify and categorize the document.

## 1.1 Supported metadata

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  $\text{\LaTeX}$  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`)

- journal article version (jav:journal\_article\_version)
- 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 version and conformance level (pdfaid:part and pdfaid:conformance)
- PDF/UA version (pdfuaid:part)
- PDF/X standard compliance (pdfxid:GTS\_PDFXVersion)
- position/title of primary author (photoshop:AuthorsPosition)
- publication name of parent publication (prism:publicationName)
- publisher of the document (dc:publisher)
- rendition variation of the document (xmpMM:RenditionClass)
- summary (dc:description)
- subtitle (prism:subtitle)
- telephone number(s) of primary author  
(Iptc4xmpCore:CreatorContactInfo.CiTelWork)
- title (dc:title)
- trapping of colors (pdf:trapped)
- type of document (dc:type)
- type of parent publication (prism:aggregationType)
- unique identifier for the document (dc:identifier)
- URL of the document (prism:url)
- URL(s) of the primary author (Iptc4xmpCore:CreatorContactInfo.CiUrlWork)
- UUID for the document (xmpMM:DocumentID)

<pre> \Title{Baking through the ages} \Author{A. Baker\sep C. Kneader} \Language{en-GB} \Keywords{cookies\sep muffins\sep cakes} \Publisher{Baking International} </pre>	<pre> \hypersetup{% pdftitle={Baking through the ages}, pdfauthor={A. Baker, C. Kneader}, pdflang={en-GB}, pdfkeywords={cookies, muffins, cakes}, pdfpublisher={Baking International} } </pre>
(a) pdfx (separate .xmpdata file)	(b) hyperxmp (main document)

Figure 1: Comparison of pdfx and hyperxmp

- 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.

## 1.2 Comparisons with similar packages

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

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

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

**pdfx** The main difference between `hyperxmp` and `pdfx` is that `hyperxmp` tries to integrate as seamlessly as possible into an existing document. It leverages `hyperref`'s `\hypersetup` command and many of `\hypersetup`'s options and defines its own options in a compatible manner. In contrast, `pdfx` requires the user to create a separate `\jobname.xmpdata` file containing `pdfx`-defined commands for each metadata element.

Figure 1 adapts an example appearing in the `pdfx` manual to `hyperxmp`. The two are comparable line-by-line in terms of how one specifies the title, author, document language, keywords, and publisher. However, `hyperxmp` implicitly writes a wealth of additional metadata into the XMP packet such as the document date, creation date, creator tool, file format, PDF version, and unique document and

instance IDs. In fact, if a document omits all of the code shown in Figure 1(b), it will still store the `\title` and `\author` data in the XMP packet.

One can therefore summarize the difference between `hyperxmp` and `pdfx` as follows: `pdfx` requires the author to be fully explicit about the document’s metadata while `hyperxmp` allows some metadata to be specified implicitly, automatically inferring it when possible. In general, `hyperxmp` tries to simplify the author’s task as much as possible.

## 2 Usage

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

- `baseurl`
- `pdfauthor`
- `pdfcreationdate`
- `pdfkeywords`
- `pdflang`
- `pdfmoddate`
- `pdfproducer`
- `pdfsubject`
- `pdftitle`
- `pdftrapped`

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

- `pdfaconformance`
- `pdfapart`
- `pdfauthortitle`
- `pdfbookedition`
- `pdfbytes`
- `pdfcaptionwriter`
- `pdfcontactaddress`
- `pdfcontactcity`
- `pdfcontactcountry`
- `pdfcontactemail`
- `pdfcontactphone`
- `pdfcontactpostcode`
- `pdfcontactregion`
- `pdfcontacturl`
- `pdfcopyright`
- `pdfdate`
- `pdfdocumentid`
- `pdfdoi`
- `pdfeissn`
- `pdfidentifier`
- `pdfinstanceid`
- `pdfisbn`
- `pdfissn`
- `pdfissuenum`
- `pdflicenseurl`
- `pdfmetadate`
- `pdfmetalang`
- `pdfnumpages`
- `pdfpagerange`
- `pdfpublication`
- `pdfpublisher`
- `pdfpubstatus`
- `pdfpubtype`
- `pdfrendition`
- `pdfsource`
- `pdfsubtitle`

- `pdftype`
- `pdfurl`
- `pdfvolumenum`
- `pdfuapart`
- `pdfversionid`
- `pdfxstandard`

## 2.1 Option descriptions

`pdftitle` The document title is specified as normal for `hyperref` with `pdftitle`, but see Note 7 on page 16 for instructions on how to specify a title in multiple languages. If `pdftitle` is not specified it will inherit its value from the document’s `\title`. `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`.

`pdfauthor` `hyperref`’s `pdfauthor` option specifies the document’s author(s). See Note 4 on page 15 for a discussion of the correct syntax. If `pdfauthor` is not specified it will inherit its value from the document’s `\author`. `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` 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 [11], 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.

`pdfdocumentid` 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 [12] for each of these. However, a document can

pdfinstanceid	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:xxxxxx-xxx-xxx-xxx-xxxxxxxxxx</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 [12]. A more freeform mechanism than <code>pdfinstanceid</code>
pdfversionid	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 5.8 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 17) version of the current Git hash that can suitably be passed to <code>pdfversionid</code> . If not specified, <code>pdfversionid</code> defaults to 1.
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	
pdfidentifier	<code>pdfidentifier</code> provides an alternative mechanism to uniquely identify a document. Its advantage relative to <code>pdfisbn</code> , <code>pdfissn</code> , <code>pdfdoi</code> , etc. is its flexibility; any of a wide variety of identification types can be used. <sup>1</sup> <code>pdfidentifier</code> ’s disadvantage is that it allows only a single identifier per document. For example, a document could use <code>pdfidentifier=urn:iso:std:32000:ed-1:v1:en</code> to identify itself as version 1 of English-language ISO standard 32000-1, but then this same document could not also use <code>pdfidentifier</code> to identify itself by DOI ( <code>info:doi/...</code> ), ISBN ( <code>urn:ISSN:...</code> ), etc. (It can still use the options described in the previous paragraph, though.) If <code>pdfidentifier</code> is not specified explicitly, <code>hyperxmp</code> will use the first non-empty value out of the DOI, electronic ISSN, print ISSN, and ISBN or skip the identifier entirely if all of those are empty.
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> ). The publisher itself can be

<sup>1</sup>See, for example, <https://www.iana.org/assignments/urn-namespaces/urn-namespaces.xhtml> for the `urn:` URI scheme and <http://info-uri.info/registry/> for the `info:` URI scheme.

Table 1: Valid arguments for `pdfpubstatus`

Value	Meaning
AO	Author’s Original
SMUR	Submitted Manuscript Under Review
AM	Accepted Manuscript
P	Proof
VoR	Version of Record
CVoR	Corrected Version of Record
EVOR	Enhanced Version of Record

<code>pdfpublisher</code>	named using <code>pdfpublisher</code> .
<code>pdfpubtype</code>	<code>pdfpubtype</code> indicates the type of publication in which the document was published. This should be one of the PRISM aggregation types [9] such as <code>book</code> , <code>journal</code> , <code>magazine</code> , <code>manual</code> , <code>report</code> , or <code>whitepaper</code> .
<code>pdfvolumenum</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>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 17 for advice on how to assign <code>pdfpagerange</code> semi-automatically. A
<code>pdfpagerange</code>	journal article’s publication status can be indicated with <code>pdfpubstatus</code> . This option expects to take one of the values listed in Table 1. See the NISO/ALPSP Journal Article Versions recommendation [1] for an explanation of each of those values and when to use them.
<code>pdfpubstatus</code>	
<code>pdfbookedition</code>	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> .
<code>pdfdate</code>	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 $\text{\LaTeX}$ <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 [5] or PDF format [4]. XMP dates are written in the form <code>YYYY-MM-DDThh:mm:ss+TT:tt</code> . <sup>2</sup> A W3C recommendation [15] 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.

The document’s creation date, modification date, and metadata date are

<sup>2</sup>Although allowed by XMP, `hyperxmp` does not currently accept fractions of a second in timestamps.



pdfcreationdate	normally set automatically, but pdfcreationdate, pdfmoddate, and pdfmetadate can
pdfmoddate	be used to override the defaults. Like pdfdate, pdfmetadate can be specified in
pdfmetadate	either XMP or PDF format. However, because hyperref defines pdfcreationdate and
	pdfmoddate and expects these to be written as PDF dates, hyperxmp concomitantly
	accepts these two dates only in PDF format as well. Note that it's rare that a
	document would need to specify any of pdfcreationdate, pdfmoddate, or pdfmetadate.
pdftype	pdftype describes the type of document being produced. This refers to “the
	nature or genre of the resource” [5] such as poem, novel or working paper, as
	opposed to the file format (always application/pdf when generated by hyperxmp).
	Although pdftype can be assigned an arbitrary piece of text, the XMP specification
	recommends selecting types from a “controlled vocabulary” such as the DCMI Type
	Vocabulary [6]. The DCMI Type Vocabulary currently consists of only Collection,
	Dataset, Event, Image, InteractiveResource, MovingImage, PhysicalObject,
	Service, Software, Sound, StillImage, and Text. pdftype defaults to Text,
	which refers to “books, letters, dissertations, poems, newspapers, articles, archives
	of mailing lists,” [6] and other forms of text—all things L <sup>A</sup> T <sub>E</sub> X is commonly used
	to typeset.
pdfrendition	Sometimes a base document is rendered in different forms. pdfrendition indicates
	the particular rendition the current document instance represents. The value
	should come from the following controlled vocabulary [5]: default, draft, low-
	res, proof, screen, and thumbnail. hyperxmp's default value is default, which
	indicates the master document, unless the draft option is passed to \documentclass,
	in which case hyperxmp defaults to draft.
pdftrapped	hyperxmp honors hyperref's pdftrapped option. A document can indicate whether
	it employs color trapping by specifying pdftrapped=True or pdftrapped=False.
	(pdftrapped=Unknown is also allowed.)
pdfapart	pdfapart and pdfaconformance, are used in conjunction with hyperref's pdfa
pdfaconformance	option to claim a particular PDF/A standard by which the document abides. They
	default to pdfapart=1 and pdfaconformance=B, indicating the PDF/A-1b standard.
	These can be changed (with caution) to assert that the document abides by a
	different standard (e.g., PDF/A-2u). A document that conforms to the PDF/UA
pdfuapart	standard can use pdfuapart to indicate the PDF/UA conformance level. For example,
pdfxstandard	pdfuapart=1 asserts that the document respects PDF/UA-1. pdfxstandard indicates
	the particular PDF/X standard by which the document abides. Unlike pdfapart and
	pdfaconformance, which accept a number and a letter, respectively, pdfxstandard
	expects a textual identification of a standard name. The following are the acceptable
	PDF/X standard names as of at the time of this writing.

- |                 |                |             |
|-----------------|----------------|-------------|
| • PDF/X-1a:2001 | • PDF/X-3:2003 | • PDF/X-5g  |
| • PDF/X-1a:2003 | • PDF/X-4      | • PDF/X-5n  |
| • PDF/X-3:2002  | • PDF/X-4p     | • PDF/X-5pg |

For example, one can specify pdfxstandard={PDF/X-4} or pdfxstandard={PDF/X-3:2003}, but specifying pdfxstandard={PDF/X-3} will not pass PDF/X validation. Note that at the time of this writing the use of the PDF/X-4p, PDF/X-5n, and PDF/X-5pg standards has not been tested.

### Rarely needed options

pdfsource	pdfsource overrides the name of the L <sup>A</sup> T <sub>E</sub> X source file. It defaults to <code>\jobname.tex</code> but can be replaced by any other string. If pdfsource is given an empty argument, no document source will be specified at all.
pdfnumpages	The number of pages in the published, print version of the document can be expressed with pdfnumpages. This is computed automatically when the document is built using either pdfL <sup>A</sup> T <sub>E</sub> X or LuaL <sup>A</sup> T <sub>E</sub> X.
pdfbytes	The pdfbytes 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. pdfbytes is computed automatically by both pdfL <sup>A</sup> T <sub>E</sub> X and LuaL <sup>A</sup> T <sub>E</sub> X, using the file size from the previous build of the document.

It is usually more convenient to provide values for all of the options presented in this section 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 L<sup>A</sup>T<sub>E</sub>X document that provides values for most of the metadata options that hyperxmp recognizes:

```
\documentclass{article}
\usepackage[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},
pdfpublisher={Wiley-VCH},
pdfpubtype={journal},
pdfvolumenum={322},
pdfissuenum={6},
pdfpagerange={132-148},
pdfissn={0003-3804},
pdfeissn={1521-3889},
pdfpubstatus={VoR},
pdflang={en},
pdfmetalang={en},
pdfurl={http://www.physik.uni-augsburg.de/annalen/history/einstein-
papers/1905_17_132-148.pdf},
pdfdoi={10.1002/andp.19053220607},
pdfidentifier={info:lcnn/50013519}
}
\XMLLangAlt{de}{pdftitle={Über einen die Erzeugung und Verwandlung des
  Lichtes betreffenden heuristischen Gesichtspunkt}}

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

```

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

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

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

Once the document is compiled, the resulting PDF file will contain an XMP packet that looks something like that shown in Appendix A. Figure 2 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 3.

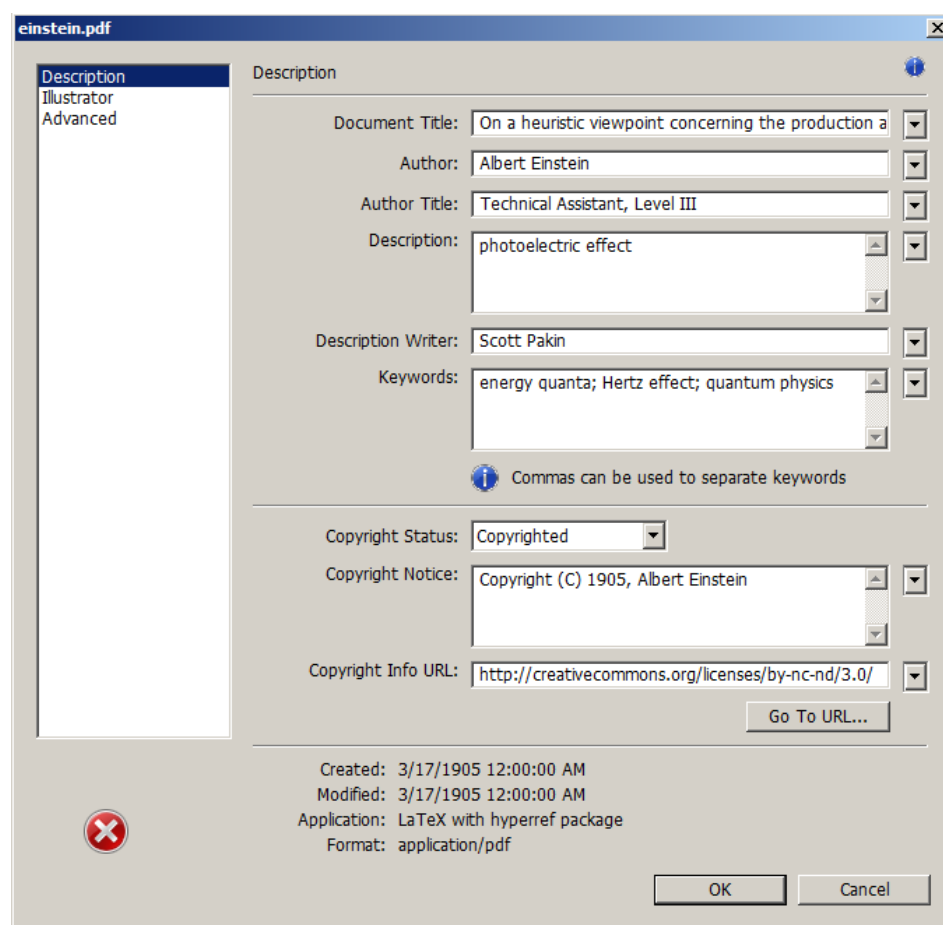


Figure 2: XMP metadata as it appears in Adobe Acrobat

## 2.3 Usage notes

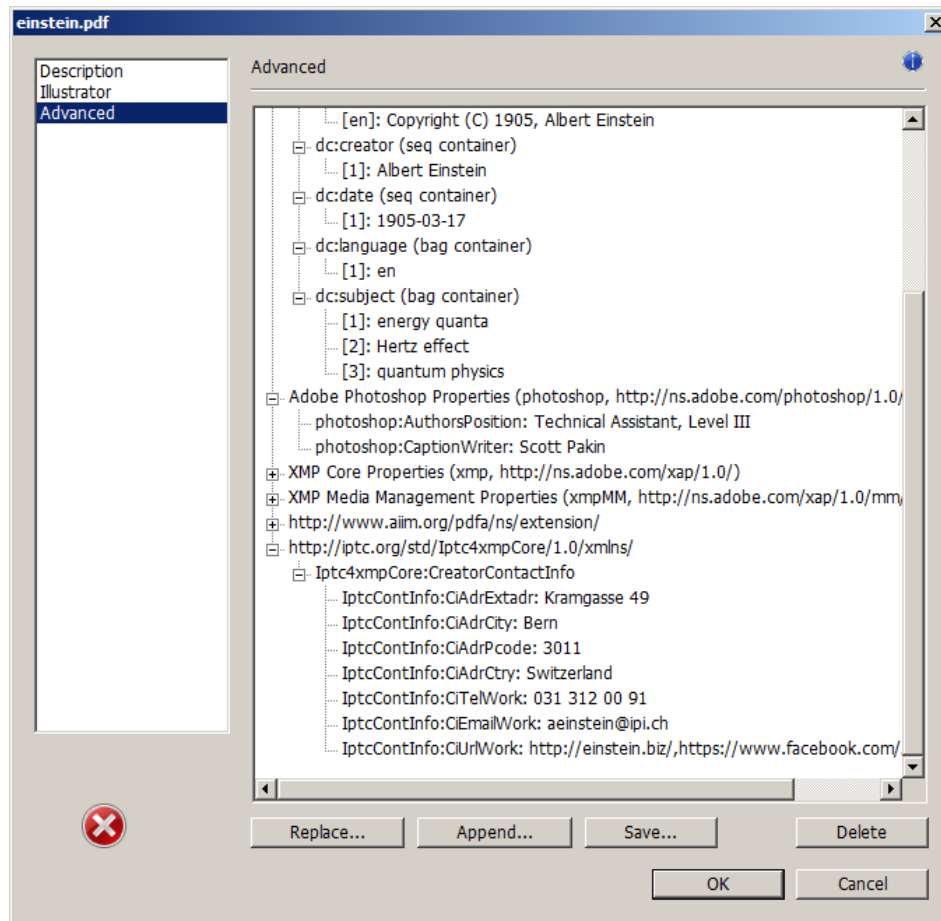


Figure 3: Additional XMP metadata as it appears in Adobe Acrobat

**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

```

<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. (`hyperxmp` v5.0+ is more sophisticated. It suppresses only the author and keyword lists.) 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 `\hyperxmpsetup`.

**Note 2: Acrobat multiline-field bug** The IPTC Photo Metadata schema states that “the [contact] address is a multiline field” [10]. `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:

```
\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 `LuaLTeX` earlier than 0.85, `hyperxmp` leaves all PDF objects uncompressed. This is due to `LuaLTeX` treating object compression as a global parameter, unlike `pdfLTeX`, which treats it as a local parameter. Hence, when `hyperxmp` requests that the XMP packet be left uncompressed, `LuaLTeX` in fact leaves *all* PDF streams uncompressed. Beginning with

version 3.0, hyperxmp includes a workaround that correctly leaves only the XMP metadata uncompressed, but this workaround is implemented only for Lua<sup>A</sup>T<sub>E</sub>X v0.85 onwards.

2. X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X (or, more precisely, the xdvipdfmx back end) exhibits the opposite problem. It compresses *all* PDF objects, including the ones containing XMP metadata. While Adobe Acrobat can still detect and utilize the XMP metadata, non-PDF-aware applications are unlikely to see the metadata. Three options to consider are to (1) use a different program (e.g., Lua<sup>A</sup>T<sub>E</sub>X), (2) pass the `--output-driver="xdvipdfmx -z0"` option to X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X 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}

If you need to include a literal comma within an author or keyword list (where 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** `hyperxmp` attempts to identify certain metadata automatically. The hope is that in many cases, an author can simply include `\usepackage{hyperxmp}` in a document’s preamble and benefit from a modicum of XMP metadata with no additional effort.

Currently, `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. `hyperxmp` recognizes some class-specific metadata as well, such as that provided via the Koma letter classes (e.g., `scrlettr2`) and the ACM article class (`acmart`).

If a document uses either the `babel` or `polyglossia` packages it is recommended that it *not* explicitly set `pdflang`. `pdflang` accepts only a single language name while `hyperxmp` can automatically query `babel` and `polyglossia` for a list of all languages used in the document and include this list in an XMP `dc:language` element.

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

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

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

```
\hypersetup{%
```



```

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

```

**Note 8: Expandable arguments** All arguments passed to `hyperxmp` options must be expandable, in  $\text{\TeX}$  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 `\printfddate{Y}` in the following code snippet is not replaced by the current year, as one might expect:

```

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

```

Rather, it generates a `dc:rights` tag of the form “Copyright © =2=0=by-1by=02020, Scott Pakin”. The garbage in that line corresponds to the remnants of the `\printfddate` code after expanding all of the  $\text{\TeX}$  primitives and certain other control sequences it uses to the empty string. For example, “`\global\advance\texd@yr 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: Semi-automatic page ranges** Although `pdfpagerange` is 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 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 the 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 `pdfpagerange` after the `\begin{document}` or to ask L<sup>A</sup>T<sub>E</sub>X to do that on your behalf:

```

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

```

**Note 10: Automatic computation of the PDF byte count** The PRISM Basic Metadata schema [8] defines a `prism:byteCount` property that indicates the PDF file size in bytes. `hyperxmp` computes this value automatically when the document is built using LuaL<sup>A</sup>T<sub>E</sub>X but not when using any other T<sub>E</sub>X engine. Note that `hyperxmp` uses the file size from the *previous* run of LuaL<sup>A</sup>T<sub>E</sub>X because the new PDF file is not yet complete. Consequently, one extra compilation is needed for the byte count to converge relative to the the number of compilations that would otherwise be required.

Starting with `hyperxmp` v5.8, the `hyperxmp` distribution includes a Perl script called `add_byteCount` that edits a PDF file in place, adding or replacing the `prism:byteCount` property with one that specifies the final file size. Run the script as “`add_byteCount <filename.pdf>`”.

```

foreach my $cmd ( "latex", "lualatex", "pdflatex", "xelatex",
                  "dvipdf", "xdvipdfmx", "ps2pdf" ) {
    ${cmd} = "internal mycmd4 ${cmd}";
}

sub mycmd4 {
    my $retval = system @_;
    if ( $$Pdest =~ /\.pdf$/ ) {
        system 'add_byteCount', $$Pdest;
    }
    return $retval;
}

```

Figure 4: `latexmk` configuration-file code for automatically invoking `add_byteCount` every time a PDF file is generated

The `latexmk` build tool can be configured to run `add_byteCount` automatically every time a PDF file is generated. Simply add the code shown in Figure 4 to your `latexmk` configuration file. See the `latexmk` manual for information on configuration-file naming on different operating systems and explanations of the hook functions used in Figure 4.

Even though `hyperxmp` can compute the byte count automatically when run from Lua $\text{\LaTeX}$ , users of `latexmk` need to use configuration-file code like that shown in Figure 4. Otherwise, `latexmk` would compile the document one time too few for the byte count to converge. It is recommended that those who use both `latexmk` and `hyperxmp` configure `latexmk` to be `hyperxmp`-aware.

### 3 Implementation

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

One thing to bear in mind when reading the `hyperxmp` source code is that different actions occur at different times throughout document processing:

1. `\usepackage{hyperxmp}`: `hyperxmp` parses package options, defines a number of commands, loads various helper packages, and assigns default values to most XMP fields.
2. `\begin{document}`: `hyperxmp` loads certain packages such as `hyperref` and `ifdraft` and queries natural-language information from `babel` and `polyglossia` that becomes available only at the end of the preamble.
3. `\end{document}`: `hyperxmp` finalizes certain data that are known only at the end of the document, such as the page count, and writes the XMP packet to the PDF file.

### 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 3.8).

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

`\hyxmp@at@end` The `\hyxmp@at@end` macro includes code at the end of the document. When available (as is the case in most modern `TeX` backends), `\AtEndDocument` works well enough. Otherwise, we invoke `\AtEndDvi` from the `atenddvi` package, which is robust but requires an additional `LaTeX` run.

```
3 \ifundefined{AtEndDocument}{%
4   \RequirePackage{atenddvi}
5   \let\hyxmp@at@end=\AtEndDvi
6 }{%
7   \let\hyxmp@at@end=\AtEndDocument
8 }
```

`\hyxmp@set@jobname` Given an expanded `\jobname` followed by `\relax`, invoke the `\hyxmp@set@jobname@dbl` macro if the job name is surrounded by double quotes and the `\hyxmp@set@jobname@plain` macro otherwise.

```
9 \def\hyxmp@set@jobname#1\relax{%
10   \ifnextchar"\hyxmp@set@jobname@dbl}{\hyxmp@set@jobname@plain}#1\relax
11 }
```

`\hyxmp@set@jobname@dbl` Set `\hyxmp@jobname` to to #1, discarding the surrounding double quotes.

```
\hyxmp@jobname 12 \def\hyxmp@set@jobname@dbl"#1"\relax{\xdef\hyxmp@jobname{#1}}
```

`\hyxmp@set@jobname@plain` Set `\hyxmp@jobname` to to #1.

```
\hyxmp@jobname 13 \def\hyxmp@set@jobname@plain#1\relax{\xdef\hyxmp@jobname{#1}}
```

Define `\hyxmp@jobname` as a sanitized version of `\jobname`. The problem with using `\jobname` directly is that it surrounds the filename with double quotes if it contains a space character. For example, a source file named `my-file.tex` results in a `\jobname` of “my-file”, but a source file named `my file.tex` results in a `\jobname` of “my file”. Trying to access “my file”.log (as is done on page 51) will fail because the filename does not in fact contain literal double quotes.

```
14 \expandafter\hyxmp@set@jobname\jobname\relax
```

`\hyxmp@aep@toks` In order for `hyperxmp` to be loaded safely during `\AtEndPreamble` we need to ensure that we perform no `\AtEndPreamble` actions until all top-level macro definitions have been made. The most straightforward approach would be to move all of `hyperxmp`’s `\AtEndPreamble` stanzas to the end of the package. However, this degrades readability of the source code. For instance, an `\AtEndPreamble` stanza

related to integration with `hyperref` could no longer appear in the “Integration with `hyperref`” section (Section 3.2). Hence, we instead store in a token list, `\hyxmp@aep@toks`, each `\AtEndPreamble` stanza as we encounter it. This token list is evaluated as one of the package’s final actions (Section 3.8).

```
15 \newtoks{\hyxmp@aep@toks}
```

## 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 XMP metadata from `hyperref`’s `baseurl`, `pdfauthor`, `pdfkeywords`, `pdflang`, `pdfproducer`, `pdfsubject`, `pdftrapped`, and `pdftitle` options. It also introduces a number of new options, which are listed on pages 5–6. For consistency with `hyperref`’s document-metadata naming conventions (which are in turn based on L<sup>A</sup>T<sub>E</sub>X’s document-metadata naming conventions), we do not prefix metadata-related macro names with our package-specific `\hyxmp@` prefix. That is, we use names like `\@pdfcopyright` instead of `\hyxmp@pdfcopyright`.

We load a bunch of helper packages: `kvoptions` for package-option processing, `pdfescape` and `stringenc` for re-encoding Unicode strings, `intcalc` for performing integer calculations (division and modulo), `iftex` for determining which T<sub>E</sub>X engine is being used, `ifmtarg` for testing if a macro argument is empty or all spaces, `etoolbox` for dynamically patching existing commands (specifically, `hyperref`’s `\PDF@FinishDoc`), and `ifthen` for convenient string comparisons.

```
16 \RequirePackage{kvoptions}
17 \RequirePackage{pdfescape}
18 \RequirePackage{stringenc}
19 \RequirePackage{intcalc}
20 \RequirePackage{iftex}
21 \RequirePackage{ifmtarg}
22 \RequirePackage{etoolbox}
23 \RequirePackage{ifthen}
```

There are a few places where `hyperxmp` can take advantage of LuaT<sub>E</sub>X features. To simplify the use of LuaT<sub>E</sub>X we load the `luacode` package.

```
24 \ifLuaTeX
25   \RequirePackage{luacode}
26 \fi
```

```
\@ifmtargexp \@ifnotmtargexp
```

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

```
27 \def\@ifmtargexp#1{\expandafter\@ifmtarg\expandafter{#1}}
28 \def\@ifnotmtargexp#1{\expandafter\@ifnotmtarg\expandafter{#1}}
```

```
\@if@def@and@nonempty
```

This macro combines `\@ifundefined` and `\@ifmtargexp`. If the macro named `#1` is both defined and non-empty, evaluate `#2`. Otherwise, evaluate `#3`.

```
29 \newcommand*{\@if@def@and@nonempty}[3]{%
```

```

30 \@ifundefined{#1}{#3}{%
31   \expandafter\ifmtargexp\expandafter{\csname#1\endcsname}{#3}{#2}%
32 }%
33 }

```

`\hyxmp@pdfstringdef` 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.

```

34 \newcommand{\hyxmp@pdfstringdef}[2]{%
35   \let\hyxmp@textunderscore=\textunderscore
36   \let\textunderscore=\hyxmp@uscore
37   \pdfstringdef{#1}{#2}%
38   \let\textunderscore=\hyxmp@textunderscore
39 }

```

`\@pdfdatetime` Prepare to store the document’s date and (optionally) time. Whether specified by the author in XMP format or PDF format (see Section 3.4.2) we always store `\@pdfdatetime` as an XMP-format string.

```

40 \def\@pdfdatetime{}
41 \define@key{Hyp}{pdfdate}{%
42   \begingroup
43     \Hy@unicodetofalse

```

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

```

44   \edef\next{%
45     \noexpand\hyxmp@pdfstringdef\noexpand\@pdfdatetime{%
46       \noexpand\hyxmp@as@xmp@date{#1}}}%
47   }%
48   \next
49 \endgroup
50 }

```

`\@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 3.4.2) we always store `\@pdfmetadatettime` as an XMP-format string.

```

51 \def\@pdfmetadatettime{}
52 \define@key{Hyp}{pdfmetadate}{%
53   \begingroup
54     \Hy@unicodetofalse

```

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

```

55   \edef\next{%
56     \noexpand\hyxmp@pdfstringdef\noexpand\@pdfmetadatettime{%
57       \noexpand\hyxmp@as@xmp@date{#1}}}%
58   }%

```

```

59     \next
60   \endgroup
61 }

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

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

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

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

\@pdfcaptionwriter Prepare to store the name of the person who inserted the hyperxmp metadata.
70 \def\@pdfcaptionwriter{}
71 \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.
72 \def\@pdfmetalang{}
73 \define@key{Hyp}{pdfmetalang}{\hyxmp@pdfstringdef\@pdfmetalang{#1}}

\hyxmp@no@bad@parts Complain about a bad pdfapart or pdfuapart if given trailing non-digits after a part
number.
74 \def\hyxmp@no@bad@parts#1\relax{%
75   \ifnotmtarg{#1}{%
76     \PackageWarning{hyperxmp}{pdfapart and pdfuapart must be numeric}%
77   }%
78 }

\@pdfapart Prepare to store the PDF/A part ID, which defaults to "1" if pdfa is passed to
hyperref.
79 \def\@pdfapart{}
80 \define@key{Hyp}{pdfapart}{%
81   \afterassignment\hyxmp@no@bad@parts\@tempcnta=0#1\relax
82   \hyxmp@pdfstringdef\@pdfapart{\the\@tempcnta}%
83 }

\@pdfaconformance Prepare to store the PDF/A conformance ID, which defaults to "b" if pdfa is passed
to hyperref and \@pdfapart is empty.
84 \def\@pdfaconformance{}

```

```

85 \define@key{Hyp}{pdfaconformance}{%
86   \uppercase{\hyxmp@pdfstringdef\@pdfaconformance{#1}}%
87 }

\@pdfuapart Prepare to store the PDF/UA part ID.
88 \def\@pdfuapart{}
89 \define@key{Hyp}{pdfuapart}{%
90   \afterassignment\hyxmp@no@bad@parts\@tempcnta=0#1\relax
91   \hyxmp@pdfstringdef\@pdfuapart{\the\@tempcnta}%
92 }

\hyxmp@set@pdfx@major Parse pdfxstandard as “PDF/X-⟨major⟩⟨other⟩”, setting \hyxmp@pdfx@major to
⟨major⟩.
93 \newcommand*{\hyxmp@set@pdfx@major}[1]{\hyxmp@set@pdfx@major@i#1!}

\hyxmp@set@pdfx@major@i This is the first helper macro for \hyxmp@set@pdfx@major. It stores the PDF/X
major version in \@tempcnta.
94 \def\hyxmp@set@pdfx@major@i PDF/X-{%
95   \afterassignment\hyxmp@set@pdfx@major@ii
96   \@tempcnta=%
97 }

\hyxmp@set@pdfx@major@ii This is the second helper macro for \hyxmp@set@pdfx@major. It copies the PDF/X
\hyxmp@pdfx@major major version from \@tempcnta to \@hyxmp@pdfx@major and discards the rest of
the PDF/X standard string.
98 \def\hyxmp@set@pdfx@major@ii#1!{%
99   \edef\hyxmp@pdfx@major{\the\@tempcnta}%
100 }

\hyxmp@check@std Compare a user-provided string to a fixed string. (Assumption: Both are names of
PDF/X standard versions.) If they match, undefine \next, which we assume was
previously defined to issue an “unrecognized standard” warning message.
101 \newcommand*{\hyxmp@check@std}[2]{%
102   \ifthenelse{\equal{#1}{#2}}{%
103     {\global\let\next=\relax}%
104     {}%
105 }%

\@pdfxstandard Prepare to store the PDF/X standard.
106 \def\@pdfxstandard{}
107 \def\hyxmp@pdfx@major{}
108 \define@key{Hyp}{pdfxstandard}{%
109   \hyxmp@pdfstringdef\@pdfxstandard{#1}%

\next Issue a warning message if the PDF/X standard named by the user does not appear
in a list of known PDF/X standards. This is to caution the user that hyperxmp
generates standard-specific XMP metadata and it can only guess at the correct

```



format for new standard versions. (See the comments on page 74 above the definition of `\hyxmp@pdfx@id@schema`, for example.)

```

110 \gdef\next{%
111   \PackageWarning{hyperxmp}{Unrecognized PDF/X standard ‘#1’}%
112 }%
113 \hyxmp@check@std{#1}{PDF/X-1a:2001}%
114 \hyxmp@check@std{#1}{PDF/X-1a:2003}%
115 \hyxmp@check@std{#1}{PDF/X-3:2002}%
116 \hyxmp@check@std{#1}{PDF/X-3:2003}%
117 \hyxmp@check@std{#1}{PDF/X-4}%
118 \hyxmp@check@std{#1}{PDF/X-4p}%
119 \hyxmp@check@std{#1}{PDF/X-5g}%
120 \hyxmp@check@std{#1}{PDF/X-5n}%
121 \hyxmp@check@std{#1}{PDF/X-5pg}%
122 \next

```

`\hyxmp@pdfx@major` Parse the PDF/X major version number from `pdfxstandard` and assign it to `\hyxmp@pdfx@major`.

```

123 \hyxmp@set@pdfx@major{#1}%
124 }

```

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

```

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

```

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

```

127 \def\hyxmp@DocumentID{}
128 \define@key{Hyp}{pdfdocumentid}{\hyxmp@pdfstringdef\hyxmp@DocumentID{#1}}

```

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

```

129 \def\hyxmp@InstanceID{}
130 \define@key{Hyp}{pdfinstanceid}{\hyxmp@pdfstringdef\hyxmp@InstanceID{#1}}

```

`\@pdfversionid` Prepare to store a string that represents the current version of the document. It defaults to “1”.

```

131 \def\@pdfversionid{1}
132 \define@key{Hyp}{pdfversionid}{\hyxmp@pdfstringdef\@pdfversionid{#1}}

```

`\ifdraft` Use the `ifdraft` package to determine if this is a draft or final document. The challenge here is that we want to use `ifdraft` if it’s already loaded, load it if not, and not break any incompatible, author-defined `\ifdraft` macros that may occur either before or after the `\usepackage{hyperxmp}`. Our solution begins by defining a new group. Then, if `ifdraft` is not yet loaded, we locally undefine `\ifdraft` and load the package. In this case, we later “unload” the package by setting `\ver@ifdraft.sty` to `\relax`.

```

133 \begingroup
134 \@ifpackageloaded{ifdraft}{%
135   \let\next=\relax

```

```

136 }{%
137   \let\ifdraft=\relax
138   \RequirePackage{ifdraft}%
139   \def\next{%
140     \expandafter\global\expandafter\let\csname ver@ifdraft.sty\endcsname=\relax
141   }%
142 }%

\@pdfrendition Prepare to store a tag describing how this rendition of the document differs from
the master. The default value is default, which indicates the master document,
except in the case of \documentclass[draft], for which \@pdfrendition defaults
to draft.

143 \ifdraft{%
144   \gdef\@pdfrendition{draft}%
145 }{%
146   \gdef\@pdfrendition{default}%
147 }
148 \next
149 \endgroup
150 \define@key{Hyp}{pdfrendition}{\hyxmp@pdfstringdef\@pdfrendition{#1}}

\@pdfpublication Prepare to store the name of the publication in which the document was published.

151 \def\@pdfpublication{}
152 \define@key{Hyp}{pdfpublication}{\hyxmp@pdfstringdef\@pdfpublication{#1}}

\@pdfpubtype Prepare to store the type of the publication in which the document was published.

153 \def\@pdfpubtype{}
154 \define@key{Hyp}{pdfpubtype}{\hyxmp@pdfstringdef\@pdfpubtype{#1}}

\@pdfbytes Prepare to store the size of the file in bytes.

155 \def\@pdfbytes{}
156 \define@key{Hyp}{pdfbytes}{\hyxmp@pdfstringdef\@pdfbytes{#1}}

\@pdfnumpages Prepare to store the number of pages in the file.

157 \def\@pdfnumpages{}
158 \define@key{Hyp}{pdfnumpages}{\hyxmp@pdfstringdef\@pdfnumpages{#1}}

\@pdfissn Prepare to store the ISSN of the publication in which the document was published.

159 \def\@pdfissn{}
160 \define@key{Hyp}{pdfissn}{\hyxmp@pdfstringdef\@pdfissn{#1}}

\@pdfeissn Prepare to store the ISSN of the electronic version of the publication in which the
document was published.

161 \def\@pdfeissn{}
162 \define@key{Hyp}{pdfeissn}{\hyxmp@pdfstringdef\@pdfeissn{#1}}

\@pdfisbn Prepare to store the ISBN of the publication in which the document was published.

163 \def\@pdfisbn{}
164 \define@key{Hyp}{pdfisbn}{\hyxmp@pdfstringdef\@pdfisbn{#1}}

```

`\@pdfbookedition` Prepare to store the edition of the book in which the document was published.

```

165 \def\@pdfbookedition{}
166 \define@key{Hyp}{pdfbookedition}{\hyxmp@pdfstringdef\@pdfbookedition{#1}}

```

`\@pdfpublisher` Prepare to store the name of the document’s publisher.

```

167 \def\@pdfpublisher{}
168 \define@key{Hyp}{pdfpublisher}{\hyxmp@pdfstringdef\@pdfpublisher{#1}}

```

`\@pdfvolumenum` Prepare to store the volume identifier of the publication in which the document was published.

```

169 \def\@pdfvolumenum{}
170 \define@key{Hyp}{pdfvolumenum}{\hyxmp@pdfstringdef\@pdfvolumenum{#1}}

```

`\@pdfissuenum` Prepare to store the identifier of the issue within a volume of the publication in which the document was published.

```

171 \def\@pdfissuenum{}
172 \define@key{Hyp}{pdfissuenum}{\hyxmp@pdfstringdef\@pdfissuenum{#1}}

```

`\@pdfpagerange` Prepare to store the document’s range of pages within the publication in which the document was published.

```

173 \def\@pdfpagerange{}
174 \define@key{Hyp}{pdfpagerange}{\hyxmp@pdfstringdef\@pdfpagerange{#1}}

```

`\@pdfdoi` Prepare to store a DOI that represents the current instance of the document.

```

175 \def\@pdfdoi{}
176 \define@key{Hyp}{pdfdoi}{\hyxmp@pdfstringdef\@pdfdoi{#1}}

```

`\@pdfurl` Prepare to store a URL that represents where the document can be found. Note that we do not prepend `baseurl` to the value provided.

```

177 \def\@pdfurl{}
178 \define@key{Hyp}{pdfurl}{\hyxmp@pdfstringdef\@pdfurl{#1}}

```

`\@pdfidentifier` Prepare to store an identifier that uniquely represents the document.

```

179 \def\@pdfidentifier{}
180 \define@key{Hyp}{pdfidentifier}{\hyxmp@pdfstringdef\@pdfidentifier{#1}}

```

`\@pdfsubtitle` Prepare to store the document’s subtitle.

```

181 \def\@pdfsubtitle{}
182 \define@key{Hyp}{pdfsubtitle}{\hyxmp@pdfstringdef\@pdfsubtitle{#1}}

```

`\@pdfpubstatus` Prepare to store the document’s journal article version.

```

183 \def\@pdfpubstatus{}
184 \define@key{Hyp}{pdfpubstatus}{\hyxmp@pdfstringdef\@pdfpubstatus{#1}}

```

The following eight macros—`\@pdfcontactaddress`, `\@pdfcontactcity`, `\@pdfcontactregion`, `\@pdfcontactpostcode`, `\@pdfcontactcountry`, `\@pdfcontactphone`, `\@pdfcontactemail`, and `\@pdfcontacturl`—together specify how to contact the person or institution responsible for the document.

`\@pdfcontactaddress` Prepare to store a street address for the document's contact person/institution. The IPTC standard defines this as follows:

The contact information address part. Comprises an optional company name and all required information to locate the building or postbox to which mail should be sent. To that end, the address is a multiline field.

For consistency with the rest of `hyperxmp`, we use commas to separate terms, in this case, lines of the address. The author can use `\xmpquote` and `\xmpcomma` to include literal commas.

```
185 \def\@pdfcontactaddress{}
186 \define@key{Hyp}{pdfcontactaddress}{%
187   \let\xmpcomma=\hyxmp@comma
188   \def\xmpquote##1{##1}%
189   \hyxmp@pdfstringdef\@pdfcontactaddress{#1}%
190   \def\xmpcomma{,}%
191   \let\xmpquote=\relax
192 }
```

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

```
193 \def\@pdfcontactcity{}
194 \define@key{Hyp}{pdfcontactcity}{\hyxmp@pdfstringdef\@pdfcontactcity{#1}}
```

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

```
195 \def\@pdfcontactregion{}
196 \define@key{Hyp}{pdfcontactregion}{\hyxmp@pdfstringdef\@pdfcontactregion{#1}}
```

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

```
197 \def\@pdfcontactpostcode{}
198 \define@key{Hyp}{pdfcontactpostcode}{\hyxmp@pdfstringdef\@pdfcontactpostcode{#1}}
```

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

```
199 \def\@pdfcontactcountry{}
200 \define@key{Hyp}{pdfcontactcountry}{\hyxmp@pdfstringdef\@pdfcontactcountry{#1}}
```

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

```
201 \def\@pdfcontactphone{}
202 \define@key{Hyp}{pdfcontactphone}{\hyxmp@pdfstringdef\@pdfcontactphone{#1}}
```

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

```
203 \def\@pdfcontactemail{}
204 \define@key{Hyp}{pdfcontactemail}{\hyxmp@pdfstringdef\@pdfcontactemail{#1}}
```

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

```
205 \def\@pdfcontacturl{}
206 \define@key{Hyp}{pdfcontacturl}{\hyxmp@pdfstringdef\@pdfcontacturl{#1}}
```

```

\hyxmp@no@info@lists Suppress hyperref from writing Author and Keywords into the Info dictionary. This
                        prevents conflicts between the PDF metadata and the XMP metadata that cause
                        PDF/A validation to fail. The PDF metadata can be restored by passing the
                        keeppdfinfo option to \hypersetup.
207 \def\hyxmp@no@info@lists{%

\hyxmp@suppress@pdf@info If \patchcmd fails for any reason—most likely, a modification to the hyperref
\next package—our fallback is to prevent hyperref from writing any data to the PDF Info
                        dictionary.
208 \def\hyxmp@suppress@pdf@info{%
209   \global\let\PDF@FinishDoc=\@empty
210   \PackageWarningNoLine{hyperxmp}{%
211     Suppressing the _entire_ PDF Info dictionary.\MessageBreak
212     Please notify the hyperxmp maintainer%
213   }%
214 }%
215 \let\next=\relax
216 \patchcmd
217   {\PDF@FinishDoc}%
218   {/Author(\@pdfauthor)}%
219   {}%
220   {}%
221   {\let\next=\hyxmp@suppress@pdf@info}%
222 \patchcmd
223   {\PDF@FinishDoc}%
224   {/Keywords(\@pdfkeywords)}%
225   {}%
226   {}%
227   {\let\next=\hyxmp@suppress@pdf@info}%
228 \next
229 }

230 \define@key{Hyp}{keeppdfinfo}[true]{%
231   \gdef\hyxmp@no@info@lists{%
232 }

```

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

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

<code>\hyxmp@pdfauthor</code> <code>\hyxmp@pdfkeywords</code>	Prepare to store the name of the author and a list of keywords. 233 <code>\def\hyxmp@pdfauthor{}</code> 234 <code>\def\hyxmp@pdfkeywords{}</code>
<code>\hyxmp@redefine@Hyp</code>	If not already redefined, redefine <code>hyperref</code> 's <code>pdfauthor</code> and <code>pdfkeywords</code> options to properly handle <code>\xmpcomma</code> and <code>\xmpquote</code> . 235 <code>\newcommand*{\hyxmp@redefine@Hyp}{%</code>
<code>\hyxmp@Hyp@pdfauthor</code>	Store the old definition of <code>\KV@Hyp@pdfauthor</code> in <code>\hyxmp@Hyp@pdfauthor</code> , but only if we see that <code>\KV@Hyp@pdfauthor</code> is defined and <code>\hyxmp@Hyp@pdfauthor</code> isn't. Otherwise, we'd be defining <code>\hyxmp@Hyp@pdfauthor</code> in terms of itself and creating an infinite loop. 236 <code>\@ifundefined{KV@Hyp@pdfauthor}{}{%</code> 237 <code>\@ifundefined{hyxmp@Hyp@pdfauthor}{%</code> 238 <code>\expandafter\let\expandafter\hyxmp@Hyp@pdfauthor</code> 239 <code>\csname KV@Hyp@pdfauthor\endcsname</code> 240 <code>}{}}%</code> 241 <code>}%</code>
<code>\KV@Hyp@pdfauthor</code> <code>\xmpcomma</code> <code>\xmpquote</code> <code>\hyxmp@and</code> <code>\and</code> <code>\hyxmp@pdfauthor</code> <code>\@pdfauthor</code>	Redefine <code>\KV@Hyp@pdfauthor</code> to process its argument twice. The first time, <code>\xmpcomma</code> is defined as a placeholder character ( <code>\hyxmp@comma</code> ) and <code>\xmpquote</code> as the identity function. The result is stored in <code>\hyxmp@pdfauthor</code> for use in structured lists (those surrounding each entry with <code>&lt;rdf:li&gt;</code> ). The second time, <code>\xmpcomma</code> is defined as an ordinary comma, and <code>\xmpquote</code> is defined as a macro that puts its argument within double quotes. The result is stored in <code>\@pdfauthor</code> for use in unstructured lists (those in which the entire list appears within a single pair of tags). In case <code>pdfauthor</code> is left unspecified and we copy <code>\author</code> 's argument to <code>pdfauthor</code> , we temporarily redefine <code>\and</code> as the list separator when producing a structured list and as "and" when producing an unstructured list. 242 <code>\define@key{Hyp}{pdfauthor}{%</code> 243 <code>\let\xmpcomma=\hyxmp@comma</code> 244 <code>\def\xmpquote####1{####1}%</code> 245 <code>\let\hyxmp@and=\and</code> 246 <code>\def\and{,}%</code> 247 <code>\hyxmp@Hyp@pdfauthor{##1}%</code> 248 <code>\global\let\hyxmp@pdfauthor=\@pdfauthor</code> 249 <code>\def\and{and\space}%</code> 250 <code>\def\xmpcomma{,}%</code> 251 <code>\def\xmpquote####1{"####1"%</code> 252 <code>\hyxmp@Hyp@pdfauthor{##1}%</code> 253 <code>\def\xmpcomma{,}%</code> 254 <code>\let\xmpquote=\relax</code> 255 <code>\let\and=\hyxmp@and</code> 256 <code>}%</code>
<code>\hyxmp@Hyp@pdfkeywords</code>	The previous block of code now repeats for the keyword list, starting by storing the old definition of <code>\KV@Hyp@pdfkeywords</code> in <code>\hyxmp@Hyp@pdfkeywords</code> .

```

257 \ifundefined{KV@Hyp@pdfkeywords}{}{%
258   \ifundefined{hyxmp@Hyp@pdfkeywords}{%
259     \expandafter\let\expandafter\hyxmp@Hyp@pdfkeywords
260     \csname KV@Hyp@pdfkeywords\endcsname
261   }{}%
262 }%

```

`\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).

```

263 \define@key{Hyp}{pdfkeywords}{%
264   \let\xmpcomma=\hyxmp@comma
265   \def\xmpquote####1{####1}%
266   \hyxmp@Hyp@pdfkeywords{##1}%
267   \global\let\hyxmp@pdfkeywords=\@pdfkeywords
268   \def\xmpcomma{,%}
269   \def\xmpquote####1{"####1"%
270   \hyxmp@Hyp@pdfkeywords{##1}%
271   \def\xmpcomma{,%}
272   \let\xmpquote=\relax
273 }%
274 }

```

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

```

275 \let\hyxmp@ProcessKeyvalOptions=\ProcessKeyvalOptions
276 \renewcommand*\ProcessKeyvalOptions{%
277   \hyxmp@redefine@Hyp
278   \hyxmp@ProcessKeyvalOptions
279 }

```

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

```

280 \let\hyxmp@hypersetup=\hypersetup
281 \def\hypersetup{%
282   \hyxmp@redefine@Hyp
283   \hyxmp@hypersetup
284 }

```

`\hyxmp@concat@metadata` Assume that if the document loaded either `babel` or `polyglossia` it will eventually define one or more languages that `hyperxmp` can list within a `dc:language` element. As explained in Section 3.1, we defer the invocation of `\AtEndPreamble` to the end of the file.

```

285 \edef\hyxmp@concat@metadata{}
286 \expandafter\hyxmp@aep@toks\expandafter=\expandafter{%
287   \the\hyxmp@aep@toks
288   \AtEndPreamble{%
289     \ifpackageloaded{babel}{%
290       \edef\hyxmp@concat@metadata{babel}%
291     }{%
292       \ifpackageloaded{polyglossia}{%
293         \edef\hyxmp@concat@metadata{polyglossia}%
294       }{%
295       }%
296     }%
297   }%
298 }

```

`\hyxmp@warn@if@no@metadata` Issue a warning message if the author failed to specify any metadata at all. This excludes metadata that is included automatically such as the current timestamp. Note that we don't consider `\@pdfmetalang` as metadata as that value is meaningful only when used in conjunction with other information. We also don't examine `\@pdfapart` or `\@pdfaconformance` because those have nonempty default values.

```

299 \newcommand*{\hyxmp@warn@if@no@metadata}{%
300   \edef\hyxmp@concat@metadata{%
301     \hyxmp@concat@metadata
302     \@baseurl
303     \@pdfauthor
304     \@pdfauthortitle
305     \@pdfbookedition
306     \@pdfbytes
307     \@pdfcaptionwriter
308     \@pdfcontactaddress
309     \@pdfcontactcity
310     \@pdfcontactcountry
311     \@pdfcontactemail
312     \@pdfcontactphone
313     \@pdfcontactpostcode
314     \@pdfcontactregion
315     \@pdfcontacturl
316     \@pdfcopyright
317     \@pdfcreationdate
318     \@pdfdatetime
319     \@pdfdoi
320     \@pdfeissn
321     \@pdfidentifier
322     \@pdfisbn
323     \@pdfissn
324     \@pdfissuenum
325     \@pdfkeywords
326     \@pdflang
327     \@pdflicenseurl

```



```

328 \pdfmetadatetitle
329 \pdfmoddate
330 \pdfnumpages
331 \pdfpagerange
332 \pdfpublication
333 \pdfpubtype
334 \pdfsubject
335 \pdfsubtitle
336 \pdftitle
337 \pdfuapart
338 \pdfurl
339 \pdfvolumenum
340 \pdfxstandard
341 }%
342 \ifx\hyxmp@concat@metadata\@empty
343 \PackageWarningNoLine{hyperxmp}{%
344 \hyxmp@jobname.tex did not specify any metadata to\MessageBreak
345 include in the XMP packet.\space\space Please see the\MessageBreak
346 hyperxmp documentation for instructions on how to\MessageBreak
347 provide metadata values to hyperxmp}%
348 \fi
349 }

```

`\hyxmp@check@standards` Most PDF standards require that certain metadata be present. If compliance with a PDF standard is claimed but any of the metadata it requires are absent, issue a warning message.

```

350 \newcommand*{\hyxmp@check@standards}{%
    If the pdfa option was passed to hyperref but \pdfapart is not set, set it to 1 and
    \pdfaconformance to B.
351 \ifHy@pdfa
352 \ifmtargexp{\pdfapart}{%
353 \PackageWarningNoLine{hyperxmp}{%
354 'pdfa' was passed to hyperref, but 'pdfapart' was\MessageBreak
355 not specified.\space\space Setting pdfapart to '1' and\MessageBreak
356 pdfaconformance to 'B'%
357 }%
358 \gdef\pdfapart{1}%
359 \gdef\pdfaconformance{B}%
360 }%
361 {}%
362 \fi

```

`\hyxmp@standards` We define `\hyxmp@standards` to be non-empty if *any* PDF standard is claimed (currently, PDF/A, PDF/X, or PDF/UA).

```

363 \edef\hyxmp@standards{%
364 \pdfapart
365 \pdfxstandard
366 \pdfuapart
367 }%

```

Check that a document title was provided and is non-empty.

```

368 \ifnotmtargexp{\hyxmp@standards}{%
369 \ifmtargexp{\@pdftitle}{%
370 \PackageWarningNoLine{hyperxmp}{%
371 Missing pdftitle (required for PDF standards\MessageBreak
372 compliance)%
373 }%
374 }%
375 {}%
376 }%
377 }

```

`\hyxmp@aep@toks` Right before we reach the `\begin{document}` we check if `hyperref` was loaded. In normal usage, the document will already have done a `\usepackage{hyperref}` because otherwise, `\hypersetup` will not have been defined, and only a limited amount of metadata will be included. However, in case the author is relying exclusively on `hyperxmp`'s automatically detected metadata, we'll need to load `hyperref` now. As explained in Section 3.1, we defer the invocation of `\AtEndPreamble` to the end of the file.

```

378 \expandafter\hyxmp@aep@toks\expandafter=\expandafter{%
379 \the\hyxmp@aep@toks
380 \AtEndPreamble{%
381 \RequirePackage{hyperref}%

```

Older versions of `hyperref` write the `Info` dictionary to the PDF file at the end of the document. Newer versions of `hyperref` write the `Info` dictionary to the PDF file at the *beginning* of the document. For compatibility with both old and new `hyperref` implementations we suppress writing the `Info` dictionary here, at the beginning of the document.

```

382 \hyxmp@no@info@lists

```

If `pdftitle` is undefined but the author invoked `\title`, we copy the latter to the former. This addresses two problems: (1) handling `LATEX` classes in which `\maketitle` clears `\title` and (2) ensuring that `hyperref` writes the same title to the PDF `Info` dictionary that `hyperxmp` writes to the XMP packet. We do likewise for `\author`  $\rightarrow$  `pdfauthor`.

One tricky bit is that the standard `LATEX` classes do not define `\@title` and `\@author` as empty strings but rather as calls to `\@latex@warning@no@line` that complain about a missing title/author. Hence, we can't simply test if the title and author are empty because they're not. Instead, we first locally redefine `\@latex@warning@no@line` to discard its argument then test if any text remains.

```

383 \begingroup
384 \let\@latex@warning@no@line=\gobble
385 \hyxmp@use@first@valid{pdftitle}{\@pdftitle}{%
386 \scr@subject@var,%
387 \@title
388 }%
389 \hyxmp@use@first@valid{pdfauthor}{\@pdfauthor}{%

```

```

390         \scr@fromname@var,%
391         \@author
392     }%
393 \endgroup
394 }%
395 }

```

When we reach the `\end{document}` we need to gather up the metadata specified explicitly by the user, infer additional metadata where possible, and write the XMP packet to the PDF file.

```

396 \hyxmp@at@end{%

```

Fill in any missing metadata we can using values provided by the author via mechanisms other than the `\hypersetup` command.

```

397 \hyxmp@auto@assign@data

```

If the document claims to comply with one or more PDF standards, check that all of the requisite metadata are present.

```

398 \hyxmp@check@standards

```

We can finally construct the XMP packet and write it to the PDF document catalog.

```

399 \hyxmp@warn@if@no@metadata
400 \hyxmp@embed@packet
401 }

```

### 3.3 Advanced metadata detection

`hyperxmp` strives to be as convenient and user-friendly as possible. To that end, we try to automatically detect as much metadata as possible. The author can of course augment or override autodetected metadata by explicitly providing values to `\hypersetup`, but the hope is that we can save the author some effort in many cases.

In this section, we identify additional metadata we can use. Most of the functionality is class- or package-specific. For example, we check for phone numbers provided to the Koma letter classes via `\setkomavar{fromphone}{...}` and/or `\setkomavar{frommobilephone}{...}`, street addresses provided to the ACM article class via `\affiliation`, and languages the `polyglossia` package is instructed to load via `\setdefaultlanguage` and `\setotherlanguage`.

```

\hyxmp@set@koma@phones Define \hyxmp@koma@phones as a comma-separated list of the phone numbers
\hyxmp@koma@phones provided to a Koma letter class (mobile and landline).
402 \newcommand*{\hyxmp@set@koma@phones}{%
403 \begingroup
404 \Hy@unicodedefalse
405 \@if@def@and@nonempty{scr@frommobilephone@var}{%
406 \@if@def@and@nonempty{scr@fromphone@var}{%
407 \hyxmp@pdfstringdef\hyxmp@koma@phones{\scr@frommobilephone@var,\scr@fromphone@var}%
408 }{%
409 \hyxmp@pdfstringdef\hyxmp@koma@phones{\scr@frommobilephone@var}%

```

```

410     }%
411   }{%
412     \if@def@and@nonempty{scr@fromphone@var}{%
413       \hyxmp@pdfstringdef\hyxmp@koma@phones{\scr@fromphone@var}%
414     }{%
415     }%
416   }%
417 \endgroup
418 }

```

`\hyxmp@use@first@valid` Given a hyperxmp option (#1), its current value (#2), and a comma-separated list of option names (#3), if the current value is empty, invoke `\hypersetup` to set the option to the first non-empty item in the list. If all items in the list are empty, do nothing.

```

419 \newcommand*{\hyxmp@use@first@valid}[3]{%
420   \ifmtargexp{#2}{%
421     \hyxmp@use@first@valid@i{#1}#3,!,%
422   }%
423   {}%
424 }

```

`\hyxmp@use@first@valid@i` This macro performs all the work for `\hyxmp@use@first@valid`. It loops over a comma-separated list of macros (#2), stopping when it encounters an end-of-list marker (“!”). The first list element that is neither undefined nor empty is assigned to a given option name (#1) using `\hypersetup`.

```

425 \def\hyxmp@use@first@valid@i#1#2,{%
426   \def\next{\hyxmp@use@first@valid@i{#1}}%
427   \ifx#2!%
428     \let\next=\relax
429   \else
430     \ifx#2\undefined
431     \else
432       \ifnotmtargexp{#2}{%
433         \hypersetup{#1={#2}}%
434         \def\next##1!,{%
435           }%
436       \fi
437     \fi
438     \next
439 }

```

`\hyxmp@auto@assign@data` If certain metadata are unspecified, try to specify meaningful values using data provided by author via other means (e.g., `\title` for the document’s title).

```

440 \newcommand*{\hyxmp@auto@assign@data}{%

```

If `\@pdflang` is not set, see if we can detect the document language via either the `babel` or `polyglossia` packages.

```

441   \if@def@and@nonempty{@pdflang}{%
442     \let\hyxmp@dc@lang=\@pdflang

```

```

443 }{%
444   \hyxmp@detect@langs
445 }%

```

Replace an empty `\@pdfmetalang`. If `\@pdflang` is defined, use that as the metadata language. Otherwise, use x-default.

```

446 \ifx\@pdfmetalang\@empty
447   \ifx\@pdflang\@empty
448     \let\@pdfmetalang=\hyxmp@x@default
449   \else
450     \edef\@pdfmetalang{\@pdflang}%
451   \fi
452 \fi

```

Identify various author-provided information that can be co-opted for use as XMP metadata.

```

453 \hyxmp@use@first@valid{pdfcontactemail}{\@pdfcontactemail}{%
454   \scr@fromemail@var
455 }%
456 \hyxmp@set@koma@phones
457 \hyxmp@use@first@valid{pdfcontactphone}{\@pdfcontactphone}{%
458   \hyxmp@koma@phones
459 }%
460 \hyxmp@use@first@valid{pdfcontacturl}{\@pdfcontacturl}{%
461   \scr@fromurl@var
462 }%
463 \hyxmp@use@first@valid{pdfsubtitle}{\@pdfsubtitle}{%
464   \@subtitle
465 }%
466 \hyxmp@use@first@valid{pdfpublisher}{\@pdfpublisher}{%
467   \@publishers
468 }%

```

We handle the `acmart` class specially. `acmart` stores author-provided contact information in a structured format that we can process fairly easily. Note that if the author is not using the `acmart` class, `\hyxmp@parse@acmart` will have been redefined to do nothing.

```

469 \hyxmp@parse@acmart

```

Most PDF standards dictate that if the same metadata appear in both the XMP packet and the PDF Info dictionary, the metadata must match. This requirement poses a problem for a user-unspecified `pdfcreationdate` in the context of  $\text{\LaTeX}$ . In this case we explicitly define `\@pdfcreationdate` as `\hyxmp@today@pdf` to prevent the `xdvipdfmx` back-end processor from detecting a missing `CreationDate` in the Info dictionary and adding its own—typically a few seconds after `hyperxmp` has constructed an `xmp:CreateDate` for the XMP metadata and leading to a metadata mismatch.

```

470 \@ifundefined{XeTeXversion}{}{%
471   \ifmtargexp{\@pdfcreationdate}{%
472     \let\@pdfcreationdate=\hyxmp@today@pdf

```

```

473     }%
474     {}%
475 }%

```

Query the document currently being built for page and byte counts.

```

476 \hyxmp@query@self
477 }

```

`hyperxmp` can directly query the page count using Lua<sub>TEX</sub> features. When any other <sub>TEX</sub> engine is used, `hyperxmp` employs the `totpages` package to help tally the total number of pages.

```

478 \ifLuaTeX
479 \else
480 \RequirePackage{totpages}
481 \fi

```

Determine the size of the output file from the *previous* run of Lua<sub>TEX</sub>. This action has to be performed before the `\begin{document}` because at that point the size of the output file is reset to zero. We use `\jobname.pdf` as the name of the output file because `status.output_file_name` is not defined at this point.

It's possible to use pdf<sub>TEX</sub>'s `\pdffilesize` primitive to query the size of `\jobname.pdf` under pdf<sub>TEX</sub>. Unfortunately, doing so has a side effect of making `latexmk` view the PDF file as an input file, which puts `latexmk` in an infinite build loop. (This was the case for `hyperxmp` v5.5 and v5.6.) See the discussion at <https://github.com/borisveytsman/acmart/issues/413> for more information.

```

482 \ifLuaTeX

```

Now that we know we're running Lua<sub>TEX</sub> we define a Lua function, `get_pdf_size`, that takes the base name of the output file and returns the number of bytes in the corresponding PDF file. One difficulty is that, at the time of this writing, Lua<sub>TEX</sub> lacks a mechanism for querying the full name of the output file. Our workaround is a tad kludgy but seems to work. We walk the list of command-line arguments for `--output-directory=<dir>`. (We in fact accept either one or two initial dashes and abbreviations as terse as `--output-d`.) Then, we concatenate the output directory (or `.` if unspecified), a path separator, the given base name of the job, and a `.pdf` extension. Alas, different operating systems use different path separators so we have to query the operating-system type to select an appropriate separator: `\` on Windows/DOS and `/` on everything else.

`get_pdf_size` is called regardless of whether we're producing PDF or DVI output. We assume that even if the user specified `--output-format=dvi`, the user's intention is eventually to convert the document to PDF.

```

483 \begin{luacode*}
484 function get_pdf_size (bname)

```

Search the list of command-line arguments for the output directory.

```

485     local odir = ""
486     for _, opt in ipairs(arg) do
487         local m = string.match(opt, "%-output%-d.-==(.*)")

```

```

488     if m then
489         odir = m
490     end
491 end

```

Set the path separator to either “/” or “\”, depending on the operating system.

```

492 local sep = "/"
493 if os.type == "windows" or os.type == "msdos" then
494     sep = "\\\"
495 end

```

Concatenate the output directory, path separator, base name, and .pdf extension. Do not insert a path separator if either (1) no output directory was specified, (2) the output directory already ends with the path separator, or (3) the output directory ends in a colon (and is therefore a relative directory) on Windows/DOS. As a few examples,

- “” + “/” + “myfile” + “.pdf” = “myfile.pdf”
- “/docs” + “/” + “myfile” + “.pdf” = “/docs/myfile.pdf”
- “/docs/” + “/” + “myfile” + “.pdf” = “/docs/myfile.pdf”
- “C:\docs” + “\” + “myfile” + “.pdf” = “C:\docs\myfile.pdf”
- “C:\docs\” + “\” + “myfile” + “.pdf” = “C:\docs\myfile.pdf”
- “C:\” + “\” + “myfile” + “.pdf” = “C:\myfile.pdf”
- “C:” + “\” + “myfile” + “.pdf” = “C:myfile.pdf”

```

496 local dlast = string.sub(odir, -1)
497 if odir == "" or dlast == sep or (dlast == ":" and sep == "\\\" then
498     sep = ""
499 end
500 local fname = odir .. sep .. bname .. ".pdf"

```

Query the file size and return it.

```

501 local nbytes = lfs.attributes(fname, "size")
502 return nbytes
503 end
504 \end{luacode*}

```

Now that we’ve defined `get_pdf_size` we invoke it, passing it `\hyxmp@jobname` as the base name of the job. (Recall that `\hyxmp@jobname` is the same as `\jobname` but with any surrounding double quotes removed.) We store `get_pdf_size`’s output—which will be empty if the PDF file doesn’t yet exist—in `\hyxmp@prev@pdf@size`.

```

505 \xdef\hyxmp@prev@pdf@size{%
506     \luadirect{
507 nbytes = get_pdf_size("\hyxmp@jobname")
508 if nbytes then
509     tex.write(nbytes)
510 end
511     }%
512 }%
513 \fi

```

`\hyxmp@query@self` Query the document currently being built to acquire page and byte counts.

```

514 \newcommand*{\hyxmp@query@self}{%
    LuaTeX exposes via status.total_pages the number of pages written. We use
    this mechanism when available to assign \@pdfnumpages. To finalize the page
    count we first issue a \clearpage.
515 \ifLuaTeX
516     \@if@def@and@nonempty{\@pdfnumpages}{%
517     }{%
518         \clearpage
519         \xdef\@pdfnumpages{\luadirect{tex.print(status.total_pages)}}%
520     }%
521 \else
    Without LuaTeX we rely on the totpages package to help count the number of
    pages. This may require an additional run of LATEX, but the user will be notified
    in that case.
522     \pdfstringdef\@pdfnumpages{\ref*{TotPages}}%
523 \fi
    If pdfbytes hasn't been set, set it to the output file's size from the previous run.
524 \hyxmp@use@first@valid{pdfbytes}{\@pdfbytes}{%
525     \hyxmp@prev@pdf@size
526 }%
527 }

```

`\hyxmp@parse@acmart` The `acmart` class stores a rich set of author metadata in its `\addresses` macro. `\hyxmp@parse@acmart` extracts the contact information for the first author and converts that to XMP metadata.

```

528 \newcommand*{\hyxmp@parse@acmart}{%
529     \begingroup

```

`\@author` `acmart` has already invoked `\hypersetup{pdfauthor=...}` to specify the complete list of authors. At this point, `\@author` is defined to produce a warning message. We locally redefine it to do nothing.

```

530     \let\@author=\@gobble

```

`\email` Within `\addresses`, `\email` is defined to accept two arguments, the second of which is the author's email address.

```

\hyxmp@address@val
531     \def\email##1##2{%
532         \def\hyxmp@address@val{##2}%
533         \hyxmp@use@first@valid{pdfcontactemail}{\@pdfcontactemail}{%
534             \hyxmp@address@val
535         }%
536     }%

```

`\streetaddress` `\streetaddress` wraps the author's street address.

```

\hyxmp@address@val
537     \def\streetaddress##1{%
538         \def\hyxmp@address@val{##1}%

```



```

539      \hyxmp@use@first@valid{pdfcontactaddress}{\@pdfcontactaddress}{%
540      \hyxmp@address@val
541      }%
542  }%

\city    \city wraps the author's city name.
\hyxmp@address@val 543      \def\city##1{%
544      \def\hyxmp@address@val{##1}%
545      \hyxmp@use@first@valid{pdfcontactcity}{\@pdfcontactcity}{%
546      \hyxmp@address@val
547      }%
548  }%

\state    \state wraps the author's state or region name.
\hyxmp@address@val 549      \def\state##1{%
550      \def\hyxmp@address@val{##1}%
551      \hyxmp@use@first@valid{pdfcontactregion}{\@pdfcontactregion}{%
552      \hyxmp@address@val
553      }%
554  }%

\country    \country wraps the author's country name.
\hyxmp@address@val 555      \def\country##1{%
556      \def\hyxmp@address@val{##1}%
557      \hyxmp@use@first@valid{pdfcontactcountry}{\@pdfcontactcountry}{%
558      \hyxmp@address@val
559      }%
560  }%

\postcode    \postcode wraps the author's postal code.
\hyxmp@address@val 561      \def\postcode##1{%
562      \def\hyxmp@address@val{##1}%
563      \hyxmp@use@first@valid{pdfcontactpostcode}{\@pdfcontactpostcode}{%
564      \hyxmp@address@val
565      }%
566  }%

\affiliation    We want to produce XMP metadata for only a single affiliation.  Although
\hyxmp@use@first@valid will ensure that only the first email, city, country, etc.
encountered is considered, we run the first of one affiliation defining, say, a city
and state but no country and a subsequent affiliation defining a country. In that
case, the XMP would include the first author's city and state and the subsequent
author's country. Hence, we define \affiliation to “self destruct” after its first
use, discarding all further affiliations.

567      \def\affiliation##1##2{%
568      ##2%
569      \let\affiliation=\@gobbletwo
570  }%

```

We want to evaluate `\addresses` with the preceding local definitions in effect, but we don't want to typeset any text appearing in the string. Hence, we “typeset” `\addresses` within a box that is subsequently discarded.

```
571 \setbox0=\hbox{\addresses}%
572 \endgroup
```

`acmart` supports other relevant metadata in addition to the authors' mailing addresses. For instance, papers accepted for publication indicate their DOI number. However, papers under review will contain either a placeholder DOI, “10.1145/nnnnnnn.nnnnnnn”, or the example DOI specified in the `acmart` example document, “10.1145/1122445.1122456”. We ignore both of those DOIs.

```
573 \if@def@and@nonempty{@acmDOI}{%
574 \IfSubStr{\@acmDOI}{10.1145/1122445.1122456}{}%
575 \IfSubStr{\@acmDOI}{10.1145/nnnnnnn.nnnnnnn}{}%
576 \hyxmp@use@first@valid{pdfdoi}{\@pdfdoi}{%
577 \@acmDOI
578 }%
579 }%
580 }%
581 }%
582 {}%
```

`\hyxmp@strip@isbn@date` Papers appearing in conference proceedings specify the proceedings' ISBN. As  
`\hyxmp@acm@isbn` with `\@acmDOI` above, we ignore both the placeholder ISBN, “978-x-xxxx-xxxx-x/YY/MM”, and the example ISBN, “978-1-4503-XXXX-X/18/06”. We also strip off the “/*year*”/“/*month*” suffix so as to include a true ISBN in the XMP metadata.

```
583 \if@def@and@nonempty{@acmISBN}{%
584 \IfSubStr{\@acmISBN}{XXXX}{}%
585 \IfSubStr{\@acmISBN}{xxxx}{}%
586 \def\hyxmp@strip@isbn@date##1/##2!{##1}%
587 \edef\hyxmp@acm@isbn{%
588 \expandafter\hyxmp@strip@isbn@date\@acmISBN/!%
589 }%
590 \hyxmp@use@first@valid{pdfisbn}{\@pdfisbn}{%
591 \hyxmp@acm@isbn
592 }%
593 }%
594 }%
595 }%
596 {}%
```

`\hyxmp@acm@publisher` The publisher is of course ACM.

```
597 \def\hyxmp@acm@publisher{Association for Computing Machinery}%
598 \hyxmp@use@first@valid{pdfpublisher}{\@pdfpublisher}{%
599 \hyxmp@acm@publisher
600 }%
```

Use the journal name if defined, otherwise the book name (for conference proceedings).

```

601 \hyxmp@use@first@valid{pdfpublication}{\@pdfpublication}{%
602   \@journalName,%
603   \@acmBooktitle,%
604   \@acmConference
605 }%

```

`\hyxmp@acm@pubtype` `acmart` makes clear whether it's typesetting a journal article. If it's not a journal, we assume it's a book (conference proceedings).

```

606 \if@ACM@journal
607   \def\hyxmp@acm@pubtype{journal}%
608 \else
609   \def\hyxmp@acm@pubtype{book}%
610 \fi
611 \hyxmp@use@first@valid{pdfpubtype}{\@pdfpubtype}{%
612   \hyxmp@acm@pubtype
613 }%

```

Journal articles have a volume and issue number.

```

614 \hyxmp@use@first@valid{pdfvolumenum}{\@pdfvolumenum}{%
615   \@acmVolume
616 }%
617 \hyxmp@use@first@valid{pdfissuenum}{\@pdfissuenum}{%
618   \@acmNumber
619 }%
620 }

```

Nullify `\hyxmp@parse@acmart` if the author is not using the `acmart` class.

```

621 \@ifclassloaded{acmart}{\let\hyxmp@parse@acmart=\relax}

```

`\hyxmp@dc@lang` `\hyxmp@dc@lang` is a comma-separated list of all languages used in the document.

```

622 \let\hyxmp@dc@lang=\@empty

```

`\hyxmp@detect@langs` If `pdflang` was not specified, try to determine the document language(s) using either `babel`'s or `polyglossia`'s definitions.

```

623 \newcommand*{\hyxmp@detect@langs}{%
624   \@ifundefined{mainbcp47id}{%
625     \@ifundefined{LocaleForEach}{%

```

The document doesn't appear to have loaded either `babel` or `polyglossia`. In this case we have one small task to do. 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 explicitly set `\@pdflang` to `\@empty` to avoid problems with non-expandable symbols.

```

626   \let\@pdflang=\@empty
627 }%

```

```

\hyxmp@dc@lang Use babel's \LocaleForEach and \getlocaleproperty to set \@pdflang to the
\hyxmp@lang@tag document's main language and \hyxmp@dc@lang to a comma-separated list of all
\hyxmp@lang@name languages used.
\@pdflang 628 \BabelEnsureInfo
629 \LocaleForEach{%
630 \getlocaleproperty\hyxmp@lang@tag{##1}{identification/tag.bcp47}%
631 \ifx\hyxmp@dc@lang\@empty
632 \xdef\hyxmp@dc@lang{\hyxmp@lang@tag}%
633 \else
634 \xdef\hyxmp@dc@lang{\hyxmp@dc@lang,\hyxmp@lang@tag}%
635 \fi
636 \def\hyxmp@lang@name{##1}%
637 \ifx\hyxmp@lang@name\bbl@main@language
638 \edef\@pdflang{\hyxmp@lang@tag}%
639 \fi
640 }%
641 }%
642 }{%
```

Use polyglossia's `\mainbcp47id` as the document's main language and its `\xpg@bcp@loaded` as a comma-separated list of all document languages.

```

643 \xdef\@pdflang{\csname mainbcp47id\endcsname}%
644 \edef\hyxmp@dc@lang{\xpg@bcp@loaded}%
645 }%
646 }
```

### 3.4 Manipulating author-supplied data

The author provides metadata information to `hyperxmp` via package options to `hyperref` or via `hyperref`'s `\hypersetup` command. The functions in this section convert author-supplied lists (e.g., `pdfkeywords={foo, bar, baz}`) into L<sup>A</sup>T<sub>E</sub>X lists (e.g., `\@elt {foo} \@elt {bar} \@elt {baz}`) that can be more easily manipulated (Section 3.4.1); parse dates in both PDF and XMP formats (Section 3.4.2); trim spaces off the ends of strings (Section 3.4.3); convert text to XML (e.g., from `<scott+hyxmp@pakin.org>` to `&lt;scott+hyxmp@pakin.org&gt;`) (Section 3.4.4); simplify the pretty-printing of a begin tag, XML text, and end tag (Section 3.4.5); and provide metadata in multiple languages (Section 3.4.6).

#### 3.4.1 List manipulation

We define a macro for converting a list of comma-separated elements (e.g., the list of PDF keywords) to a list of L<sup>A</sup>T<sub>E</sub>X `\@elt`-separated elements.

```

\hyxmp@commas@to@list Given a macro name (#1) and a comma-separated list (#2), define the macro name
as the elements of the list, each preceded by \@elt. (Executing the macro therefore
applies \@elt to each element in turn.)
647 \newcommand*{\hyxmp@commas@to@list}[2]{%
648 \gdef#1{%
```

```

649 \expandafter\hyxmp@commas@to@list@i\expandafter#1#2,,%
650 }

\hyxmp@commas@to@list@i Recursively construct macro #1 from comma-separated list #2. Stop if #2 is empty.
\next 651 \def\hyxmp@commas@to@list@i#1#2,{%
652 \gdef\hyxmp@sublist{#2}%
653 \ifx\hyxmp@sublist\@empty
654 \let\next=\relax
655 \else
656 \hyxmp@trimspaces\hyxmp@sublist
657 \@cons{#1}{\hyxmp@sublist}%
658 \def\next{\hyxmp@commas@to@list@i{#1}}%
659 \fi
660 \next
661 }

\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.
662 \def\xmpcomma{,}%

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

663 \bgroup
664 \catcode'\^C=11
665 \gdef\hyxmp@comma{^C}
666 \egroup

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

667 \bgroup
668 \catcode'\^U=11
669 \gdef\hyxmp@uscore{^U}
670 \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.
671 \let\xmpquote=\relax

```

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

```

672 \bgroup
673 \catcode'\~ =12%
674 \gdef\xmptilde{~}%
675 \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` One can then write “`\XMPTruncateList{pdfauthor}`” and have Adobe Acrobat

`\hyxmp@temp@list` display the author list correctly.

```

676 \newcommand{\XMPTruncateList}[1]{%
677   \PackageWarning{hyperxmp}{%
678     \noexpand\XMPTruncateList has been deprecated since\MessageBreak
679     hyperxmp 4.0 and may be removed in future\MessageBreak
680     versions of the package. \noexpand\XMPTruncateList\MessageBreak
681     was found}%
682   \edef\hyxmp@temp@str{\csname hyxmp@#1\endcsname}%
683   \hyxmp@commas@to@list{\hyxmp@temp@list}{\hyxmp@temp@str}%
684   \def\@elt##1{%
685     \expandafter\gdef\csname @#1\endcsname{##1}%
686     \let\@elt=\@gobble
687   }
688   \hyxmp@temp@list
689 }%

```

### 3.4.2 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:20201118225924-07’00’) [4], while XMP timestamps are of the form “YYYY-MM-DDThh:mm:ss+TT’tt’” (e.g., 2020-11-18T22:59:24-07:00) [5]. 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`

```

690 \def\hyxmp@first@char#1{\hyxmp@first@char@i#1\relax}
691 \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.

```

692 \def\hyxmp@as@xmp@date#1{%
693   \expandafter\ifnum\expandafter'\hyxmp@first@char@i#1\relax='D
694   \hyxmp@pdf@to@xmp@date{#1}%
695   \else
696     #1%

```

```

697  \fi
698  }

\hyxmp@pdf@to@xmp@date  Convert a timestamp from PDF format to XMP format. This macro is fully expand-
                        able.
699  \def\hyxmp@pdf@to@xmp@date#1:#2#3#4#5#6#7#8#9{%
700    #2#3#4#5-#6#7-#8#9%
701    \hyxmp@parse@time
702  }

\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.
703  \def\hyxmp@parse@time#1#2#3#4#5#6{%
704    T#1#2:#3#4:#5#6%
705    \hyxmp@parse@tz@char
706  }

\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, includ-
                        ing Asia, Oceania, and most of Europe), “-” for western timezones (UTC-x, pri-
                        marily 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.
707  \def\hyxmp@parse@tz@char#1{%
708    #1%
709    \ifx#1-%
710      \expandafter\hyxmp@parse@tz
711    \else
712      \ifx#1+%
713        \expandafter\hyxmp@parse@tz
714      \fi
715    \fi
716  }

\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.
717  \def\hyxmp@parse@tz#1'#2' {%
718    #1:#2%
719  }

\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.
720  \def\hyxmp@as@pdf@date#1{%

```

```

721 \expandafter\ifx\hyxmp@first@char@i#1\relax D%
722   #1%
723 \else
724   \hyxmp@xmp@to@pdf@date{#1}%
725 \fi
726 }

```

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

```

727 \def\hyxmp@xmp@to@pdf@date#1{%
728   D:\hyxmp@xmp@to@pdf@date@i#1\relax\relax
729 }

```

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

```

730 \def\hyxmp@xmp@to@pdf@date@i#1#2#3#4#5#6{%
731   #1#2#3#4%
732   \ifx#5-%
733     \expandafter\hyxmp@xmp@to@pdf@date@ii\expandafter#6%
734   \fi
735 }

```

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

```

736 \def\hyxmp@xmp@to@pdf@date@ii#1#2#3#4{%
737   #1#2%
738   \ifx#3-%
739     \expandafter\hyxmp@xmp@to@pdf@date@iii\expandafter#4%
740   \fi
741 }

```

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

```

742 \def\hyxmp@xmp@to@pdf@date@iii#1#2#3#4{%
743   #1#2%
744   \ifx#3T%
745     \expandafter\hyxmp@xmp@to@pdf@date@iv\expandafter#4%
746   \fi
747 }

```

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

```

748 \def\hyxmp@xmp@to@pdf@date@iv#1#2#3#4{%
749   #1#2%
750   \ifx#3:%
751     \expandafter\hyxmp@xmp@to@pdf@date@v\expandafter#4%
752   \fi
753 }

```

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

```

754 \def\hyxmp@xmp@to@pdf@date@v#1#2#3#4{%
755   #1#2%
756   \ifx#3:%

```



```

757 \expandafter\hyxmp@xmp@to@pdf@date@vi\expandafter#4%
758 \fi
759 }

```

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

```

760 \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).

```

761 \def\hyxmp@xmp@to@pdf@date@vi#1#2#3#4{%
762   #1#2%
763   \ifx#3+%
764     \expandafter\hyxmp@xmp@to@pdf@date@vii
765   \fi
766   \ifx#3-%
767     -\expandafter\hyxmp@xmp@to@pdf@date@vii
768   \fi
769   \ifx#3Z%
770     Z%
771   \fi
772   \ifx#3\relax
773     \expandafter\hyxmp@gobbletwo
774   \fi
775   \@gobbletwo #4%
776 }

```

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

```

777 \def\hyxmp@xmp@to@pdf@date@vii#1\@gobbletwo#2#3#4#5{%
778   #2#3%
779   \ifx#4:%
780     \expandafter\hyxmp@xmp@to@pdf@date@viii\expandafter#5%
781   \fi
782 }

```

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

```

783 \def\hyxmp@xmp@to@pdf@date@viii#1#2#3#4{%
784   '#1#2'%
785 }

```

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

```

786 \def\hyxmp@today@xmp@define#1{%

```

The date is a straightforward representation of T<sub>E</sub>X's `\year`, `\month`, and `\day` primitives, with the latter two zero-padded to two digits apiece.

```

787 \xdef#1{\the\year}%
788 \ifnum\month<10
789   \xdef#1{#1-0\the\month}%
790 \else
791   \xdef#1{#1-\the\month}%
792 \fi
793 \ifnum\day<10
794   \xdef#1{#1-0\the\day}%
795 \else
796   \xdef#1{#1-\the\day}%
797 \fi

```

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

```

798 \@tempcnta=\time
799 \divide\@tempcnta by 60
800 \ifnum\@tempcnta<10
801   \xdef#1{#1T0\the\@tempcnta}%
802 \else
803   \xdef#1{#1T\the\@tempcnta}%
804 \fi
805 \multiply\@tempcnta by -60
806 \advance\@tempcnta by \time
807 \ifnum\@tempcnta<10
808   \xdef#1{#1:0\the\@tempcnta}%
809 \else
810   \xdef#1{#1:\the\@tempcnta}%
811 \fi
812 \xdef#1{#1Z}%
813 }

```

`\hyxmp@try@today` If `\hyxmp@today@xmp` is still empty and #1 is defined, evaluate #2. Otherwise, do nothing.

```

814 \def\hyxmp@try@today#1#2{%
815   \ifmtargexp{\hyxmp@today@xmp}{%
816     \ifundefined{#1}{#2}%
817   }%
818   {}%
819 }

```

`\hyxmp@today@xmp` Define `\hyxmp@today@xmp` as the current date and (if available) time and timezone in XMP Date format [5].

```

820 \def\hyxmp@today@xmp{}

```

Case 1: `\pdfcreationdate` is defined (pdfL<sup>A</sup>T<sub>E</sub>X and pre-0.85 LuaL<sup>A</sup>T<sub>E</sub>X).

```

821 \hyxmp@try@today{\pdfcreationdate}{%
822   \edef\hyxmp@today@xmp{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
823 }

```

Case 2: `\pdffeedback` is defined (Lua<sup>A</sup>T<sub>E</sub>X 0.85+).

```
824 \hyxmp@try@today{\pdffeedback}{%
825   \edef\hyxmp@today@xmp{\expandafter\hyxmp@pdf@to@xmp@date\pdffeedback creationdate}%
826 }
```

`\hyxmp@timestamp` Case 3: `\filemoddate` is defined (X<sub>Y</sub>-L<sup>A</sup>T<sub>E</sub>X). In this case, we treat the timestamp of the job's `.log` file as the current date/time.

```
827 \hyxmp@try@today{\filemoddate}{%
828   \edef\hyxmp@today@xmp{\filemoddate{\hyxmp@jobname.log}}%
829   \edef\next{%
830     \edef\noexpand\hyxmp@today@xmp{\noexpand\hyxmp@as@xmp@date{\hyxmp@today@xmp}}%
831   }%
832   \next
833 }
```

Case 4: None of the above. Do the best we can using the available T<sub>E</sub>X primitives (`\year`, `\month`, `\day`, and `\time`).

```
834 \hyxmp@try@today{\year}{%
835   \hyxmp@today@xmp@define\hyxmp@today@xmp
836 }
```

`\hyxmp@today@pdf` Define `\hyxmp@today@pdf` as the current date and (if available) time and timezone in PDF date format [4]. To do so we simply convert `\hyxmp@today@xmp`, defined above, from XMP to PDF using `\hyxmp@xmp@to@pdf@date`.

```
837 \expandafter\edef\expandafter\hyxmp@today@pdf\expandafter{%
838   \expandafter\hyxmp@xmp@to@pdf@date\expandafter{\hyxmp@today@xmp}%
839 }
```

### 3.4.3 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 [7]. Inline comments are also taken from the solution text.

```
840 \catcode'\Q=3
\hyxmp@trimspaces\x redefines \x to have the same replacement text sans leading
and trailing space tokens.
841 \newcommand{\hyxmp@trimspaces}[1]{%
  Use grouping to emulate a multi-token afterassignment queue.
842   \begingroup
  Put “\toks 0 {” into the afterassignment queue.
843   \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.

```
844 \expandafter\hyxmp@trimb\expandafter\noexpand#1Q Q}%
```

Transfer the trimmed text back into #1.

```
845 \edef#1{\the\toks0}%
```

```
846 }
```

`\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.

```
847 \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.

```
848 \def\hyxmp@trimc#1Q#2{\afterassignment\endgroup \vfuzz\the\vfuzz#1}
```

```
849 \catcode'\Q=11
```

### 3.4.4 Converting text to XML

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

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

```
850 \newif\ifhyxmp@unicodetex
```

```
851 \ifnum64='\^^^0040\relax
```

```
852 \hyxmp@unicodetextrue
```

```
853 \else
```

```
854 \hyxmp@unicodetexfalse
```

```
855 \fi
```

`\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.

```
856 \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.

```
857 \expandafter\def\csname SE->pdfdoc@15\endcsname{0015}
```

`\hyxmp@xmlify` Given a piece of text defined using `\pdfstringdef` (i.e., with many special characters redefined to have category code 11), set `\hyxmp@xmlified` to the same text  
`\hyxmp@xmlified` but with all occurrences of “<” replaced with `&lt;`; all occurrences of “>” replaced  
`\hyxmp@text` with `&gt;`; and all occurrences of “&” replaced with `&amp;`;.

```

858 \newcommand*{\hyxmp@xmlify}[1]{%
859   \gdef\hyxmp@xmlified{%
      Escaped PDF string → PDFDocEncoding/Unicode
860   \EdefUnescapeString\hyxmp@text{#1}%
861   \ifhyxmp@unicodetex
      PDFDocEncoding/Unicode → UTF-32BE
862     \hyxmp@is@unicode\hyxmp@text{%
863       \StringEncodingConvert
864       \hyxmp@text\hyxmp@text{utf16be}{utf32be}%
865     }{%
866       \ifXeTeX
867         \hyxmp@xetex@crap
868       \else
869         \StringEncodingConvert
870         \hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
871       \fi
872     }%
      UTF-32BE → UTF-32BE as hex string
873     \EdefEscapeHex\hyxmp@text{\hyxmp@text}%
      UTF-32BE → XML in ASCII
874     \edef\hyxmp@text{%
875       \expandafter
876     }\expandafter\hyxmp@toxml@unicodetex\hyxmp@text
877     \relax\relax\relax\relax\relax\relax\relax\relax
878   \else
      PDFDocEncoding/Unicode → UTF-8
879     \hyxmp@is@unicode\hyxmp@text{%
880       \StringEncodingConvert
881       \hyxmp@text\hyxmp@text{utf16be}{utf8}%
882     }{%
883       \StringEncodingConvert
884       \hyxmp@text\hyxmp@text{pdfdoc}{utf8}%
885     }%
      UTF-8 → UTF-8 as hex string
886     \EdefEscapeHex\hyxmp@text{\hyxmp@text}%
      UTF-8 as hex string → XML in UTF-8 as hex string
887     \edef\hyxmp@text{%
888       \expandafter\hyxmp@toxml\hyxmp@text\@empty\@empty
889     }%
  
```

XML in UTF-8 as hex string  $\rightarrow$  XML in UTF-8

```
890 \EdefUnescapeHex\hyxmp@text{\hyxmp@text}%
891 \fi
892 \global\let\hyxmp@xmlified\hyxmp@text
893 }
```

`\hyxmp@is@unicode` Given a string and two expressions, evaluate the first expression if the string is UTF-16BE-encoded and the second expression if not.

```
894 \begingroup
895 \lccode'\<=254 %
896 \lccode'\>=255 %
897 \catcode254=12 %
898 \catcode255=12 %
899 \lowercase{\endgroup
900 \def\hyxmp@is@unicode#1{%
901 \expandafter\hyxmp@@is@unicode#1<>\@nil
902 }%
903 \def\hyxmp@@is@unicode#1<>#2\@nil{%
904 \ifx\#1\%
905 \expandafter\@firstoftwo
906 \else
907 \expandafter\@secondoftwo
908 \fi
909 }%
910 }
```

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

```
911 \def\hyxmp@toxml#1#2{%
912 \ifx#1\@empty
913 \else
914 \ifnum"#1#2='\& %
915 26616D703B% &#2;
916 \else\ifnum"#1#2='\< %
917 266C743B% <#2;
918 \else\ifnum"#1#2='\> %
919 2667743B% >#2;
920 \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” [3]. 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.

```

921     \@ifundefined{pdfmark}{%
922         #1#2%
923     }{%
924         \ifnum"#1#2='\( %
925             5C28% \((
926         \else\ifnum"#1#2='\) %
927             5C29% \)
928         \else
929             #1#2%
930         \fi\fi
931     }%
932     \fi\fi\fi
933     \expandafter\hyxmp@toxml
934 \fi
935 }
```

`\hyxmp@toxml@unicodetex` Replace the characters “<”, “&”, and “>” with XML entities when using a native-Unicode T<sub>E</sub>X (X<sub>Y</sub>T<sub>E</sub>X or LuaT<sub>E</sub>X).

`\hyxmp@text`

```

936 \def\hyxmp@toxml@unicodetex#1#2#3#4#5#6#7#8{%
937     \ifx#1\relax
938     \else
939         \ifnum"#1#2#3#4#5#6#7#8>127 %
940             \uccode'\*="#1#2#3#4#5#6#7#8\relax
941             \uppercase{%
942                 \edef\hyxmp@text{\hyxmp@text *}%
943             }%
944         \else\ifnum"#7#8='\< %
945             \edef\hyxmp@text{\hyxmp@text &lt;}%
946         \else\ifnum"#7#8='\& %
947             \edef\hyxmp@text{\hyxmp@text &amp;}%
948         \else\ifnum"#7#8='\> %
949             \edef\hyxmp@text{\hyxmp@text &gt;}%
950         \else\ifnum"#7#8='\ %
951             \edef\hyxmp@text{\hyxmp@text\space}%
952         \else
953             \uccode'\*="#7#8\relax
954             \uppercase{%
955                 \edef\hyxmp@text{\hyxmp@text *}%
956             }%
957         \fi\fi\fi\fi\fi
958         \expandafter\hyxmp@toxml@unicodetex
959     \fi
960 }
```

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

```

961 \def\hyxmp@skipzeros#1{%
962     \ifx#10%
```

```

963     \expandafter\hyxmp@skipzeros
964   \fi
965 }

\hyxmp@xetex@crap  \x In the case of XeTeX, the strings defined by \pdfstringdef can contain big
                    characters. In this case, the string is treated as Unicode.
\hyxmp@try 966 \begingroup
\hyxmp@crap@result 967 \def\x#1{\endgroup
\hyxmp@text 968   \def\hyxmp@xetex@crap{%
969     \edef\hyxmp@try{%
970       \expandafter\hyxmp@SpaceOther\hyxmp@text#1\@nil
971     }%
972     \let\hyxmp@crap@result=N%
973     \expandafter\hyxmp@crap@test\hyxmp@try\relax
974     \ifx\hyxmp@crap@result Y%
975       \let\hyxmp@text\@empty
976       \expandafter\hyxmp@crap@convert\hyxmp@try\relax
977     \else
978       \StringEncodingConvert\hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
979     \fi
980   }%
981 }
982 \x{ }

\hyxmp@SpaceOther Re-encode all spaces in a string with category code 12 (“other”).
983 \begingroup
984   \catcode'\~ =12 %
985   \lccode'\~ ='\ %
986   \lowercase{\endgroup
987   \def\hyxmp@SpaceOther#1 #2\@nil{%
988     #1%
989     \ifx\relax#2\relax
990       \expandafter\@gobble
991     \else
992       ~%
993       \expandafter\@firstofone
994     \fi
995     {\hyxmp@SpaceOther#2\@nil}%
996   }%
997 }

\hyxmp@crap@test Determine if we need to treat a string as Unicode.
998 \def\hyxmp@crap@test#1{%
999   \ifx#1\relax
1000   \else
1001     \ifnum'#1>127 %
1002       \let\hyxmp@crap@result=Y%
1003       \expandafter\expandafter\expandafter\hyxmp@skiptorelax
1004     \else

```



```

1005     \expandafter\expandafter\expandafter\hyxmp@crap@test
1006     \fi
1007 \fi
1008 }

```

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

```

1009 \def\hyxmp@skiptorelax#1\relax{}

```

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

```

\hyxmp@num 1010 \def\hyxmp@crap@convert#1{%
\hyxmp@text 1011 \ifx#1\relax
1012 \else
1013 \edef\hyxmp@num{\number'#1}%
1014 \ifnum\hyxmp@num>"FFFFFF %
1015 \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"1000000}\relax
1016 \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
1017 \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"1000000}}%
1018 \else
1019 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
1020 \fi
1021 \ifnum\hyxmp@num>"FFFF %
1022 \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"10000}\relax
1023 \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
1024 \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"10000}}%
1025 \else
1026 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
1027 \fi
1028 \ifnum\hyxmp@num>"FF %
1029 \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"100}\relax
1030 \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
1031 \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"100}}%
1032 \else
1033 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
1034 \fi
1035 \ifnum\hyxmp@num>0 %
1036 \lccode'\!=\hyxmp@num\relax
1037 \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
1038 \else
1039 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
1040 \fi
1041 \expandafter\hyxmp@crap@convert
1042 \fi
1043 }

```

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

```

1044 \begingroup
1045 \catcode0=12 %
1046 \gdef\hyxmp@zero{^^00}%
1047 \endgroup

```

### 3.4.5 Outputting structured XML

An XMP packet consists of structured XML data. We define some helper routines to handle the repetitive tasks of indenting a consistent number of spaces, inserting begin and end tags, and escaping arbitrary text as necessary for XML compatibility.

`\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.

```
1048 \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.

```
1049 \newcommand*{\hyxmp@add@simple}[2]{%
1050   \ifnotmtargexp{#2}{%
1051     \hyxmp@xmllify{#2}%
1052     \hyxmp@add@to@xml{\hyxmp@extra@indent_____<}%
1053     \xdef\hyxmp@xml{\hyxmp@xml#1}%
1054     \hyxmp@add@to@xml{>\hyxmp@xmllified</>%
1055     \xdef\hyxmp@xml{\hyxmp@xml#1>^^J}%
1056   }%
1057 }
```

`\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.

```
1058 \newcommand*{\hyxmp@add@simple@var}[2]{%
1059   \expandafter\ifx\csname#2\endcsname\relax
1060   \else
1061     \hyxmp@xmllify{\csname#2\endcsname}%
1062     \hyxmp@add@to@xml{%
1063       \hyxmp@extra@indent_____<#1>\hyxmp@xmllified</#1>^^J%
1064     }%
1065   \fi
1066 }
```

`\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.

```
1067 \newcommand*{\hyxmp@add@simple@lang}[2]{%
1068   \ifnotmtarg{#2}{%
1069     \hyxmp@xmllify{#2}%
1070   }
```

```

1070     \expandafter\hyxmp@add@simple@lang@i\hyxmp@xmllified\relax{#1}%
1071   }%
1072 }

```

`\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.

```

1073 \newcommand*{\hyxmp@add@simple@lang@i}{%
1074   \ifnextchar[\hyxmp@add@simple@lang@iif\hyxmp@add@simple@lang@i[\@pdfmetalang]}%
1075 }

```

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

```

1076 \def\hyxmp@add@simple@lang@iif[#1]#2\relax#3{%
1077   \ifnotmtarg{#2}{%
1078     \hyxmp@xmllify{#2}%
1079     \ifmtarg{#1}{%
1080       \hyxmp@add@to@xml{%
1081         <#3>\hyxmp@xmllified</#3>^^J%
1082       }%
1083     }{%
1084       \hyxmp@add@to@xml{%
1085         <#3 xml:lang="#1">\hyxmp@xmllified</#3>^^J%
1086       }%
1087     }%
1088   }%
1089 }

```

`\hyxmp@add@simple@pfx` Given an XMP tag (`#1`), a—typically hard-wired—prefix string (`#2`), and a main string (`#3`), if the main string is nonempty, add a begin tag, both strings, and an end tag to the packet. The “simple” in the macro name indicates that the string is output without variations for different languages.

```

1090 \newcommand*{\hyxmp@add@simple@pfx}[3]{%
1091   \ifnotmtargexp{#3}{%
1092     \hyxmp@add@to@xml{\hyxmp@extra@indent_____<}%
1093     \xdef\hyxmp@xml{\hyxmp@xml#1}%
1094     \hyxmp@pdfstringdef\hyxmp@iprefix{#2}%
1095     \hyxmp@xmllify{\hyxmp@iprefix}%
1096     \hyxmp@add@to@xml{>\hyxmp@xmllified}%
1097     \hyxmp@xmllify{#3}%
1098     \hyxmp@add@to@xml{\hyxmp@xmllified</}%
1099     \xdef\hyxmp@xml{\hyxmp@xml#1>^^J}%
1100   }%
1101 }

```

### 3.4.6 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}`”.

```
1102 \def\hyxmp@alt@title{}
1103 \def\hyxmp@alt@description{}
1104 \def\hyxmp@alt@rights{}
```

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

```
1105 \newcommand{\hyxmp@LA@accept}[2]{%
1106   \define@key{hyxmp@LA}{#1}{%
```

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

```
1107   \hyxmp@pdfstringdef\hyxmp@value{##1}%
1108   \xdef#2{#2\noexpand\do {\hyxmp@cur@lang} {\hyxmp@value}}%
1109 }
1110 }
```

Define `<key>=<value>` options for appending to each of the `\hyxmp@alt<tag>`  
lists.

```
1111 \hyxmp@LA@accept{pdftitle}{\hyxmp@alt@title}
1112 \hyxmp@LA@accept{pdfsubject}{\hyxmp@alt@description}
1113 \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 `<key>=<value>` 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.

```
1114 \newcommand{\XMPLangAlt}[2]{%
1115   \let\do=\relax
```

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

```
1116   \edef\hyxmp@cur@lang{#1}%
1117   \setkeys{hyxmp@LA}{#2}%
1118 }
```

### 3.5 UUID generation

We use a linear congruential generator to produce pseudorandom version 4 UUIDs [12]. True, this method has its flaws but it's simple to implement in T<sub>E</sub>X and is good enough for producing the XMP xmpMM:DocumentID and xmpMM:InstanceID fields.

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

```
1119 \def\hyxmp@modulo@a#1{%
1120   \@tempcntb=\@tempcnta
1121   \divide\@tempcntb by #1
1122   \multiply\@tempcntb by #1
1123   \advance\@tempcnta by -\@tempcntb
1124 }
```

`\hyxmp@big@prime` Define a couple of large prime numbers that can still be stored in a T<sub>E</sub>X counter.

```
\hyxmp@big@prime@ii 1125 \def\hyxmp@big@prime{536870923}
1126 \def\hyxmp@big@prime@ii{536870027}
```

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

```
\hyxmp@one@token 1127 \def\hyxmp@seed@rng#1{%
1128   \@tempcnta=\hyxmp@big@prime
1129   \futurelet\hyxmp@one@token\hyxmp@seed@rng@i#1\@empty
1130 }
```

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

```
\next 1131 \def\hyxmp@seed@rng@i{%
1132   \ifx\hyxmp@one@token\@empty
1133     \let\next=\relax
1134   \else
1135     \def\next##1{%
1136       \multiply\@tempcnta by 3
1137       \advance\@tempcnta by '##1
1138       \hyxmp@modulo@a{\hyxmp@big@prime}%
1139       \futurelet\hyxmp@one@token\hyxmp@seed@rng@i
1140     }%
1141   \fi
1142   \next
1143 }
```

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

```
1144 \def\hyxmp@set@rand@num{%
1145   \@tempcnta=\hyxmp@rand@num
1146   \multiply\@tempcnta by 3
```

```

1147 \advance\@tempcnta by \hyxmp@big@prime@ii
1148 \hyxmp@modulo@a{\hyxmp@big@prime}%
1149 \xdef\hyxmp@rand@num{\the\@tempcnta}%
1150 }

```

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

```

1151 \def\hyxmp@append@hex#1{%
1152 \hyxmp@set@rand@num
1153 \@tempcnta=\hyxmp@rand@num
1154 \hyxmp@modulo@a{16}%
1155 \ifnum\@tempcnta<10
1156 \xdef#1{#1\the\@tempcnta}%
1157 \else

```

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

```

1158 \advance\@tempcnta by -10
1159 \ifcase\@tempcnta
1160 \xdef#1{#1a}%
1161 \or\xdef#1{#1b}%
1162 \or\xdef#1{#1c}%
1163 \or\xdef#1{#1d}%
1164 \or\xdef#1{#1e}%
1165 \or\xdef#1{#1f}%
1166 \fi
1167 \fi
1168 }

```

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

```

1169 \def\hyxmp@append@hex@iii#1{%
1170 \hyxmp@append@hex#1%
1171 \hyxmp@append@hex#1%
1172 \hyxmp@append@hex#1%
1173 }

```

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

```

1174 \def\hyxmp@append@hex@iv#1{%
1175 \hyxmp@append@hex@iii#1%
1176 \hyxmp@append@hex#1%
1177 }

```

`\hyxmp@create@uuid` As per the definition of a version 4 UUID [12], define macro #1 as a UUID of the form “`uuid:xxxxxxxx-xxxx-4xxx-yxxx-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`.

```

1178 \def\hyxmp@create@uuid#1{%
1179 \def#1{uuid:}%
1180 \hyxmp@append@hex@iv#1%

```

```

1181 \hyxmp@append@hex@iv#1%
1182 \g@addto@macro#1{-}%
1183 \hyxmp@append@hex@iv#1%
1184 \g@addto@macro#1{-4}%
1185 \hyxmp@append@hex@iii#1%
1186 \g@addto@macro#1{-}%

```

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

```

1187 \hyxmp@set@rand@num
1188 \@tempcnta=\hyxmp@rand@num
1189 \hyxmp@modulo@a{4}%
1190 \ifcase\@tempcnta
1191   \g@addto@macro#1{8}%
1192   \or\g@addto@macro#1{9}%
1193   \or\g@addto@macro#1{a}%
1194   \or\g@addto@macro#1{b}%
1195 \fi
1196 \hyxmp@append@hex@iii#1%
1197 \g@addto@macro#1{-}%
1198 \hyxmp@append@hex@iv#1%
1199 \hyxmp@append@hex@iv#1%
1200 \hyxmp@append@hex@iv#1%
1201 }

```

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

```

1202 \newcommand*{\hyxmp@def@DocumentID}{%
1203   \edef\hyxmp@seed@string{\hyxmp@jobname:\@pdftitle:\@pdfauthor}%
1204   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
1205   \edef\hyxmp@rand@num{\the\@tempcnta}%
1206   \hyxmp@create@uuid\hyxmp@DocumentID
1207 }

```

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

```

1208 \newcommand*{\hyxmp@def@InstanceID}{%
1209   \hyxmp@today@xmp@define{\hyxmp@seed@string}%
1210   \edef\hyxmp@seed@string{%
1211     \hyxmp@jobname:\@pdftitle:\@pdfauthor:\hyxmp@today@xmp:\hyxmp@seed@string
1212   }%
1213   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
1214   \edef\hyxmp@rand@num{\the\@tempcnta}%

```

```

1215 \hyxmp@create@uuid\hyxmp@InstanceID
1216 }

```

## 3.6 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” [5]. (“PI” is an abbreviation for “processing instructions”). The serialized XMP includes blocks of XML for various XMP schemata: Adobe PDF (Section 3.6.2), Dublin Core (Section 3.6.3), XMP Rights Management (Section 3.6.4), XMP Media Management (Section 3.6.5), XMP Basic (Section 3.6.6), Photoshop (Section 3.6.7), PDF/\* Identification (Section 3.6.8), IPTC Photo Metadata (Section 3.6.9), PRISM Basic Metadata (Section 3.6.10), Journal Article Versions (Section 3.6.11), and XMP Paged-Text (Section 3.6.12). The `\hyxmp@construct@packet` macro (Section 3.6.14) 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.

### 3.6.1 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.

```

1217 \newcommand*{\hyxmp@add@to+xml}[1]{%
1218   \bgroup
1219   \@tempcnta=0
1220   \ifhyxmp@unicodetex
1221     \@tempcntb=65536%
1222   \else
1223     \@tempcntb=256%
1224   \fi
1225   \loop
1226     \lccode\@tempcnta=\@tempcnta
1227     \advance\@tempcnta by 1
1228     \ifnum\@tempcnta<\@tempcntb
1229       \repeat
1230     \lccode'\_='\ \relax
1231     \lccode'\^C='\,\relax
1232     \lccode'\^U='\_\relax
1233     \lowercase{\xdef\hyxmp@new+xml{#1}}%
1234     \xdef\hyxmp+xml{\hyxmp+xml\hyxmp@new+xml}%
1235   \egroup
1236 }

```

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

```

1237 \bgroup
1238 \catcode'\#=11
1239 \gdef\hyxmp@hash{#}

```



```

1240 \egroup

\hyxmp@padding  The XMP specification recommends leaving approximately 2000 bytes of whites-
\hyxmp@xml      pace at the end of each XMP packet to facilitate editing the packet in place [5].
                \hyxmp@padding is defined to contain 32 lines of 63 spaces and a newline apiece
                for a total of 2048 characters of whitespace.

1241 \bgroup
1242   \xdef\hyxmp@xml{%
1243     \hyxmp@add@to@xml{%
1244       -----^^J%
1245     }
1246     \xdef\hyxmp@padding{\hyxmp@xml}%
1247 \egroup
1248 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
1249 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
1250 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
1251 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
1252 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}

\hyxmp@x@default  Define an x-default string that we can use in comparisons with \@pdfmetalang.
1253 \newcommand*{\hyxmp@x@default}{x-default}

```

### 3.6.2 The Adobe PDF schema

Older versions of `hyperref` defined a default producer; newer versions do not. Instead, they let the  $\text{\TeX}$  engine define the producer itself. This poses a problem for PDF/A compliance because `hyperxmp` sees an empty producer and therefore omits writing a `pdf:Producer` to the XMP packet, causing a mismatch between the data in the XMP packet and the data in the PDF `Info` dictionary. To ensure consistency between XMP and `Info`, we explicitly define our own default `\@pdfproducer` here.

```

\@pdfproducer  Define \@pdfproducer using the banner string if available or the \TeX engine's
\hyxmp@define@pdfproducer  version number if not.

1254 \newcommand*{\hyxmp@define@pdfproducer}{%
1255   \gdef\@pdfproducer{\TeX}
1256   \ifLuaTeX
1257     \expandafter\hyxmp@banner@to@producer\expandafter{\luatexbanner}%
1258   \else
1259     \ifPDFTeX
1260       \expandafter\hyxmp@banner@to@producer\expandafter{\pdftexbanner}%
1261     \else
1262       \ifXeTeX
1263         \edef\@pdfproducer{XeTeX version \the\XeTeXversion\XeTeXrevision}%
1264       \fi
1265     \fi
1266   \fi
1267 }

```

`\pdfproducer` Define `\pdfproducer` as the TeX engine’s banner string (e.g., “This is LuaHBTeX, Version 1.12.0 (TeX Live 2020)”), removing the initial “This is” if possible (specifically, when  $\varepsilon$ -TeX’s `\scantokens` primitive is available).

```

1268 \def\hyxmp@banner@to@producer#1{%
1269   \ifx\scantokens\relax
1270     \gdef\pdfproducer{#1}%
1271   \else
1272     {\scantokens{\makeatletter\hyxmp@remove@this#1\relax}}%
1273   \fi
1274 }

```

`\pdfproducer` Define `\pdfproducer` as a given banner string with the initial “This is” stripped off the beginning.

```

1275 \def\hyxmp@remove@this This is #1\relax{\gdef\pdfproducer{#1}}

```

If `pdfproducer` wasn’t specified and `hyperref` didn’t already define `\pdfproducer`—old versions of `hyperref` did; newer ones don’t—try to assign a meaningful producer string and use that.

```

1276 \AtBeginDocument{%
1277   \ifx\pdfproducer\relax
1278     \hyxmp@define@pdfproducer
1279   \fi
1280 }

```

`\hyxmp@assign@major@minor` Assign `\hyxmp@major@minor` to be the PDF version targeted by the running TeX engine.

`\hyxmp@major@minor`

```

1281 \newcommand*{\hyxmp@assign@major@minor}{%
1282   \@ifundefined{pdfvariable}{%
1283     \@ifundefined{pdfminorversion}{%

```

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

```

1284       }{%

```

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

```

1285         \xdef\hyxmp@major@minor{\the\pdfminorversion}%
1286         \@ifundefined{pdfmajorversion}{%

```

Case 2(a): `\pdfmajorversion` is not defined (older versions of pdfLaTeX and LuaLaTeX).

```

1287           \xdef\hyxmp@major@minor{1.\hyxmp@major@minor}%
1288           }{%

```

Case 2(b): `\pdfmajorversion` is defined (pdfLaTeX 1.40.21+).

```

1289             \xdef\hyxmp@major@minor{\the\pdfmajorversion.\hyxmp@major@minor}%
1290             }%
1291           }%
1292         }%

```

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

```
1293 \xdef\hyxmp@major@minor{\the\pdfvariable majorversion.\the\pdfvariable minorversion}%
1294 }%
1295 }
```

`\hyxmp@pdf@schema` Add properties defined by the Adobe PDF schema to the `\hyxmp@xml` macro.

```
1296 \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.
1297 \hyxmp@add@simple@var{pdf:Producer}{@pdfproducer}%
1298 \hyxmp@add@simple@var{pdf:Keywords}{@pdfkeywords}%
1299 \hyxmp@add@simple{pdf:Trapped}{@pdftrapped}%
1300 \hyxmp@assign@major@minor
1301 \hyxmp@add@simple@var{pdf:PDFVersion}{hyxmp@major@minor}%
1302 }
```

### 3.6.3 The Dublin Core schema

`\ifhyxmp@multi@langs` These macros are used locally to `\hyxmp@rdf@dc`. If the property being processed has values in different languages, `\ifhyxmp@multi@langs` evaluates to TRUE. If it has a value in only a single language, `\ifhyxmp@multi@langs` evaluates to FALSE.

```
1303 \newif\ifhyxmp@multi@langs
```

`\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.

```
1304 \newcommand*{\hyxmp@rdf@dc}[3][\iffalse]{%
    Set \@tempswattrue only if the given text is nonempty or the provided conditional
    evaluates to TRUE.
1305 \@ifmtargexp{#3}{\@tempswafalse}{\@tempswattrue}%
1306 #1
1307 \@tempswattrue
1308 \fi
    Append the corresponding XML only if \@tempswattrue.
1309 \if@tempswa
1310 \hyxmp@xmlify{#3}%
1311 }
```

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

```

1311 \let\hyxmp@value=\hyxmp@xmlified
1312 \hyxmp@add@to@xml{%
1313 -----<dc:#2>^^J%
1314 -----<rdf:Alt>^^J%
1315 }%

```

Record whether property #2 has definitions in multiple languages.

```

1316 \@if@def@and@nonempty{hyxmp@alt@#2}{%
1317 \hyxmp@multi@langstrue
1318 }{%
1319 \hyxmp@multi@langfalse
1320 }%

```

There are now four cases to consider: the cross product of `{pdfmetalang = “x-default”, pdfmetalang ≠ “x-default”}` and `{\XMPLangAlt was specified, \XMPLangAlt was not specified}`. We handle each of these in turn.

```

1321 \ifx\@pdfmetalang\hyxmp@x@default
1322 \ifhyxmp@multi@langs

```

**Case 1:** No `pdfmetalang` but `\XMPLangAlt`. We consider this an error because the x-default language will not have a matching non-default language, in violation of the XMP specification’s guidance [5, p. 23]:

An **xml:lang** value of “x-default” may be used to explicitly denote a default item. If used, the “x-default” item shall be first in the array and its simple text value should be repeated in another item in which **xml:lang** specifies its actual language. However, an “x-default” item may be the only item, in which case there is only a default value in no defined language.

```

1323 \PackageError{hyperxmp}%
1324 {\string\XMPLangAlt\space was used without specifying
1325 pdfmetalang\MessageBreak
1326 or pdflang}%
1327 {Be sure to assign a language code to the pdfmetalang key and/or a
1328 document\MessageBreak
1329 language to the pdflang key (e.g., \string\hypersetup{pdfmetalang={en}}).}%
1330 \else

```

**Case 2:** No `pdfmetalang` and no `\XMPLangAlt`. Here we specify only x-default as the language, as per the guidance quoted above.

```

1331 \hyxmp@add@to@xml{%
1332 -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@value</rdf:li>^^J%
1333 }%
1334 \fi
1335 \else
1336 \ifhyxmp@multi@langs

```

**Case 3:** Both `pdfmetalang` and `\XMPLangAlt`. In this case, we include an x-default followed by the `pdfmetalang` language, followed by all of the language alternatives.

```

1337     \hyxmp@xmlify{\@pdfmetalang}%
1338     \hyxmp@add@to@xml{%
1339 -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@value</rdf:li>^^J%
1340 -----<rdf:li xml:lang="\hyxmp@xmlified">\hyxmp@value</rdf:li>^^J%
1341     }%
1342     \def\do##1##2{
1343         \hyxmp@xmlify{##2}%
1344         \hyxmp@add@to@xml{%
1345 -----<rdf:li xml:lang="##1">\hyxmp@xmlified</rdf:li>^^J%
1346         }%
1347     }%
1348     \csname hyxmp@alt@#2\endcsname
1349     \else

```

Case 4: pdfmetalang but no \XMPLangAlt. To reduce redundancy we omit the x-default and include the single language in which the text appears.

```

1350     \hyxmp@xmlify{\@pdfmetalang}%
1351     \hyxmp@add@to@xml{%
1352 -----<rdf:li xml:lang="\hyxmp@xmlified">\hyxmp@value</rdf:li>^^J%
1353     }%
1354     \fi
1355     \fi

```

Complete this XMP element.

```

1356     \hyxmp@add@to@xml{%
1357 -----</rdf:Alt>^^J%
1358 -----</dc:#2>^^J%
1359     }%
1360     \fi
1361 }%

```

\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.

```

1362 \newcommand*{\hyxmp@list@to@xml}[4][\iffalse]{%

```

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

```

1363 \ifmtargexp{#4}{\@tempswafalse}{\@tempswatruue}%
1364 #1
1365 \@tempswatruue
1366 \fi

```

Append the corresponding XML only if \@tempswatruue.

```

1367 \if@tempswa
1368     \hyxmp@add@to@xml{%
1369 -----<dc:#2>^^J%
1370 -----<rdf:#3>^^J%
1371     }%
1372     \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`.

```

1373     \hyxmp@xmlify{#4}%
1374     \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
1375     \def\@elt##1{%
1376         \hyxmp@add@to@xml{%
1377     -----<rdf:li>##1</rdf:li>^^J%
1378         }%
1379     }%
1380     \hyxmp@list
1381     \egroup
1382     \hyxmp@add@to@xml{%
1383     -----</rdf:#3>^^J%
1384     -----</dc:#2>^^J%
1385     }%
1386     \fi
1387 }
```

`\hyxmp@singleton@dc` Given an optional list type (Seq or Bag), a Dublin Core property, and a string, append a block of XML representing a one-element list consisting of the given string.

```

1388 \newcommand{\hyxmp@singleton@dc}[3][Bag]{%
1389     \@ifnotmtargexp{#3}{%
1390         \hyxmp@xmlify{#3}%
1391         \hyxmp@add@to@xml{%
1392     -----<dc:#2>^^J%
1393     -----<rdf:#1>^^J%
1394     -----<rdf:li>\hyxmp@xmlified</rdf:li>^^J%
1395     -----</rdf:#1>^^J%
1396     -----</dc:#2>^^J%
1397         }%
1398     }
1399 }
```

`\hyxmp@cond@dc@identifier` Conditionally add a `dc:identifier` tag. Given a prefix string (#1) and a main string (#2), wrap these in a `dc:identifier` if the main string is nonempty and `\hyxmp@xmlified` is empty (implying the `dc:identifier` has not yet been written).

```

1400 \newcommand*{\hyxmp@cond@dc@identifier}[2]{%
1401     \ifx\hyxmp@xmlified\@empty
1402         \@ifnotmtargexp{#2}{%
1403             \hyxmp@add@simple@pfx{dc:identifier}{#1}{#2}%
1404         }%
1405     \fi
1406 }
```

`\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`, the `dc:language` property if the author specified `pdflang`, the `dc:type` property if the author specified `pdftype`, and the `dc:identifier` if the author specified `pdfidentifier` or if we can derive it from other options. We also specify the `dc:source` property using the base name of the source file with `.tex` appended and the `dc:date` property using the date the document was run through L<sup>A</sup>T<sub>E</sub>X—unless the author specified `pdfdate`, in which case we use that.

```

1407 \newcommand*{\hyxmp@dc@schema}{%
1408   \hyxmp@add@simple{dc:format}{application/pdf}%
1409   \hyxmp@rdf@dc[\ifHy@pdfa]{title}{\@pdftitle}%
1410   \hyxmp@rdf@dc[\ifHy@pdfa]{description}{\@pdfsubject}%
1411   \hyxmp@rdf@dc{rights}{\@pdfcopyright}%
1412   \hyxmp@singleton@dc{publisher}{\@pdfpublisher}%
1413   \ifmtargexp{\@pdfdatetime}{%
1414     \hyxmp@singleton@dc[Seq]{date}{\hyxmp@today@xmp}%
1415   }{%
1416     \hyxmp@singleton@dc[Seq]{date}{\@pdfdatetime}%
1417   }%
1418   \hyxmp@singleton@dc{type}{\@pdftype}%
1419   \hyxmp@list@to@xml[\ifHy@pdfa]{creator}{Seq}{\hyxmp@pdfauthor}%
1420   \hyxmp@list@to@xml{subject}{Bag}{\hyxmp@pdfkeywords}%
1421   \ifx\@pdfsource\@empty
1422   \else
1423     \hyxmp@add@simple{dc:source}{\@pdfsource}%
1424   \fi
1425   \hyxmp@list@to@xml{language}{Bag}{\hyxmp@dc@lang}%
1426 % If |\@pdfidentifier| is empty, try setting it to each of |\@pdfdoi|,
1427 % |\@pdfeissn|, |\@pdfissn|, and |\@pdfisbn|, in turn, with proper
1428 % syntactic adjustments.
1429 %   \begin{macrocode}
1430   \ifmtargexp{\@pdfidentifier}{%
1431     \let\hyxmp@xmlified=\@empty
1432     \hyxmp@cond@dc@identifier{info:doi/}{\@pdfdoi}%
1433     \hyxmp@cond@dc@identifier{urn:ISSN:}{\@pdfeissn}%
1434     \hyxmp@cond@dc@identifier{urn:ISSN:}{\@pdfissn}%
1435     \hyxmp@cond@dc@identifier{urn:ISBN:}{\@pdfisbn}%
1436   }{%
1437     \hyxmp@add@simple{dc:identifier}{\@pdfidentifier}%
1438   }%
1439 }
```

### 3.6.4 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.

```

1440 \newcommand*{\hyxmp@xmpRights@schema}{%
\hyxmp@legal Set \hyxmp@rights to YES if either pdfcopyright or pdflicenseurl was specified.
1441 \let\hyxmp@rights=\@empty
1442 \ifx\@pdflicenseurl\@empty
1443 \else
1444 \def\hyxmp@rights{YES}%
1445 \fi
1446 \ifx\@pdfcopyright\@empty
1447 \else
1448 \def\hyxmp@rights{YES}%
1449 \fi

Include the license-statement URL and/or the copyright indication. The copyright
statement itself is included by \hyxmp@dc@schema in Section 3.6.3.

1450 \ifx\hyxmp@rights\@empty
1451 \else
1452 \ifx\@pdfcopyright\@empty
1453 \else
1454 \hyxmp@add@simple{xmpRights:Marked}{True}%
1455 \fi
1456 \hyxmp@add@simple{xmpRights:WebStatement}{\@pdflicenseurl}%
1457 \fi
1458 }

```

### 3.6.5 The XMP Media Management schema

`\hyxmp@aep@toks` Once we reach the end of the preamble and know that `\@pdftitle` and `\@pdfauthor` are no longer expected to change we use those macros (and others) to define one UUID for the document (`\hyxmp@DocumentID`) and one for the document instance (`\hyxmp@InstanceID`). As explained in Section 3.1, we defer the invocation of `\AtEndPreamble` to the end of the file.

```

1459 \expandafter\hyxmp@aep@toks\expandafter=\expandafter{%
1460 \the\hyxmp@aep@toks
1461 \AtEndPreamble{%
1462 \ifmtargexp{\hyxmp@DocumentID}{\hyxmp@def@DocumentID}{}%
1463 \ifmtargexp{\hyxmp@InstanceID}{\hyxmp@def@InstanceID}{}%
1464 }%
1465 }

```

`\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 [5]. As seen in Section 3.5, we do what we can to honor this intention from within a T<sub>E</sub>X-based workflow. We additionally support the `xmpMM:VersionID` property, whose value is supplied by the author using `pdfversionid`.

```

1466 \gdef\hyxmp@mm@schema{%

```



```

1467 \hyxmp@add@simple{xmpMM:DocumentID}{\hyxmp@DocumentID}%
1468 \hyxmp@add@simple{xmpMM:InstanceID}{\hyxmp@InstanceID}%
1469 \hyxmp@add@simple{xmpMM:VersionID}{\@pdfversionid}%
1470 \hyxmp@add@simple{xmpMM:RenditionClass}{\@pdfrendition}%
1471 }

```

### 3.6.6 The XMP Basic schema

`\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`.

```

1472 \newcommand*{\hyxmp@xmp@basic@schema}{%
    For the document's creation date, use the user-specified \@pdfcreationdate if
    defined and non-empty. Otherwise use our fabricated \hyxmp@today@xmp.
1473 \@ifmtargexp{\@pdfcreationdate}{%
1474     \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@today@xmp}%
1475 }{%
1476     \hyxmp@add@simple{xmp:CreateDate}{%
1477         \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfcreationdate}}%
1478 }%
    For the document's modification date, use the user-specified \@pdfmoddate if
    defined and non-empty. Otherwise use our fabricated \hyxmp@today@xmp.
1479 \@ifmtargexp{\@pdfmoddate}{%
1480     \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@today@xmp}%
1481 }{%
1482     \hyxmp@add@simple{xmp:ModifyDate}{%
1483         \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfmoddate}}%
1484 }%
    For the document's metadata date, use the user-specified \@pdfmetadatetime if
    defined and non-empty. Otherwise use our fabricated \hyxmp@today@xmp.
1485 \@ifmtargexp{\@pdfmetadatetime}{%
1486     \hyxmp@add@simple{xmp:MetadataDate}{\hyxmp@today@xmp}%
1487 }{%
1488     \hyxmp@add@simple{xmp:MetadataDate}{\@pdfmetadatetime}%
1489 }%
    Define the creation tool and the base URL.
1490 \hyxmp@add@simple{xmp:CreatorTool}{\@pdfcreator}%
1491 \hyxmp@add@simple{xmp:BaseURL}{\@baseurl}%
1492 }

```

### 3.6.7 The Photoshop schema

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

```

1493 \gdef\hyxmp@photoshop@schema{%
1494   \edef\hyxmp@photoshop@data{\@pdfauthortitle\@pdfcaptionwriter}%
1495   \hyxmp@add@simple{photoshop:AuthorsPosition}{\@pdfauthortitle}%
1496   \hyxmp@add@simple{photoshop:CaptionWriter}{\@pdfcaptionwriter}%
1497 }

```

### 3.6.8 PDF/\* Identification schemata

**\hyxmp@pdfa@id@schema** Add properties defined by the PDF/A Identification schema [13] 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**.

```

1498 \newcommand*{\hyxmp@pdfa@id@schema}{%
1499   \ifHy@pdfa
1500     \hyxmp@add@simple{pdfaid:part}{\pdfapart}%
1501     \hyxmp@add@simple{pdfaid:conformance}{\pdfaconformance}%
1502   \fi
1503 }

```

**\hyxmp@pdfua@id@schema** If the document conforms to a PDF/UA standard, the author can indicate the standard version with **pdfuapart**.

```

1504 \newcommand*{\hyxmp@pdfua@id@schema}{%
1505   \hyxmp@add@simple{pdfuaid:part}{\pdfuapart}%
1506 }

```

**\hyxmp@pdfx@id@schema** If the document conforms to a PDF/X standard, the author can indicate the standard version with **pdfxstandard**. We separately handle PDF/X-1, PDF/X-2 and PDF/X-3, and PDF/X-4 onwards.

```

1507 \newcommand*{\hyxmp@pdfx@id@schema}{%
1508   \@tempcnta=0\hyxmp@pdfx@major\relax
1509   \ifnum\@tempcnta=0
1510     \else
1511       \ifnum\@tempcnta=1
1512         \hyxmp@add@simple{pdfx:GTS_PDFXVersion}{PDF/X-1:2001}%
1513         \hyxmp@add@simple{pdfx:GTS_PDFXConformance}{\pdfxstandard}%
1514       \else
1515         \ifnum\@tempcnta<4
1516           \hyxmp@add@simple{pdfx:GTS_PDFXVersion}{\pdfxstandard}%
1517         \else
1518           \hyxmp@add@simple{pdfxid:GTS_PDFXVersion}{\pdfxstandard}%
1519         \fi
1520       \fi
1521     \fi
1522 }

```

### 3.6.9 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.

```
1523 \begingroup
1524   \catcode'\&=12
1525   \catcode'\#=12
1526   \gdef\xmplinesep{&#xA;}
1527 \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.

```
1528 \newcommand*{\hyxmp@list@to@lines}[2]{%
1529   \ifnotmtargexp{#2}{%
1530     \bgroup
1531     \hyxmp@add@to@xml{%
1532       \hyxmp@extra@indent_____<#1>%
1533     }%
```

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

```
1534   \def\@elt@first##1{%
1535     \hyxmp@add@to@xml{##1}%
1536     \let\@elt=\@elt@rest
1537   }%
```

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

```
1538   \def\@elt@rest##1{%
1539     \hyxmp@add@to@xml{\xmplinesep##1}%
1540   }%
```

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

```
1541   \let\@elt=\@elt@first
1542   \hyxmp@xmlify{#2}%
1543   \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
1544   \hyxmp@list
1545   \hyxmp@add@to@xml{</#1>^J}%
1546 \egroup
1547 }%
1548 }
```

`\hyxmp@iptc@schema` Add properties defined by the IPTC Photo Metadata schema [10] to the `\hyxmp@xml` macro. We currently support only the `lptc4xmpCore:CreatorContactInfo` property, although this is a structure containing multiple fields.

```
1549 \gdef\hyxmp@iptc@schema{%
```

Because we currently support only `Iptc4xmpCore:CreatorContactInfo` it suffices to check if we have any relevant data. If so, we instantiate a `Iptc4xmpCore:ContactInfo` structure with all available fields.

```

1550 \ifx\hyxmp@iptc@data\@empty
1551 \else
1552 \hyxmp@add@to@xml{%
1553 -----<Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">^^J%
1554 }%

```

We locally redefine `\hyxmp@extra@indent` to increase the indentation of the assignments to `Iptc4xmpCore:CreatorContactInfo`'s fields.

```

1555 \bgroup
1556 \edef\hyxmp@extra@indent{\hyxmp@extra@indent\space\space}%
1557 \hyxmp@list@to@lines{Iptc4xmpCore:CiAdrExtadr}{\@pdfcontactaddress}%
1558 \hyxmp@add@simple{Iptc4xmpCore:CiAdrCity}{\@pdfcontactcity}%
1559 \hyxmp@add@simple{Iptc4xmpCore:CiAdrRegion}{\@pdfcontactregion}%
1560 \hyxmp@add@simple{Iptc4xmpCore:CiAdrPcode}{\@pdfcontactpostcode}%
1561 \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” [10]. 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.

```

1562 \def\xmplinesep{,}%
1563 \hyxmp@list@to@lines{Iptc4xmpCore:CiTelWork}{\@pdfcontactphone}%
1564 \hyxmp@list@to@lines{Iptc4xmpCore:CiEmailWork}{\@pdfcontactemail}%
1565 \hyxmp@list@to@lines{Iptc4xmpCore:CiUrlWork}{\@pdfcontacturl}%
1566 \egroup
1567 \hyxmp@add@to@xml{%
1568 -----</Iptc4xmpCore:CreatorContactInfo>^^J%
1569 }%
1570 \fi
1571 }

```

### 3.6.10 The PRISM Basic Metadata schema

`\hyxmp@prism@schema` Add properties defined by the PRISM Basic Metadata schema [8].

```

1572 \newcommand*{\hyxmp@prism@schema}{%
1573 \ifx\hyxmp@prism@data\@empty
1574 \else
1575 \hyxmp@add@simple{prism:complianceProfile}{three}%
1576 \fi
1577 \hyxmp@add@simple@lang{prism:subtitle}{\@pdfsubtitle}%
1578 \hyxmp@add@simple@lang{prism:publicationName}{\@pdfpublication}%
1579 \hyxmp@add@simple{prism:aggregationType}{\@pdfpubtype}%

```

```

1580 \hyxmp@add@simple@lang{prism:bookEdition}{\@pdfbookedition}%
1581 \hyxmp@add@simple{prism:volume}{\@pdfvolumenum}%
1582 \hyxmp@add@simple{prism:number}{\@pdfissuenum}%
1583 \hyxmp@add@simple{prism:pageRange}{\@pdfpagerange}%
1584 \hyxmp@add@simple{prism:isbn}{\@pdfisbn}%
1585 \hyxmp@add@simple{prism:issn}{\@pdfissn}%
1586 \hyxmp@add@simple{prism:eIssn}{\@pdfeissn}%
1587 \hyxmp@add@simple{prism:doi}{\@pdfdoi}%
1588 \hyxmp@add@simple{prism:url}{\@pdfurl}%
1589 \hyxmp@add@simple{prism:byteCount}{\@pdfbytes}%
1590 \hyxmp@add@simple{prism:pageCount}{\@pdfnumpages}%
1591 }

```

### 3.6.11 The Journal Article Versions (JAV) schema

`\hyxmp@jav@schema` Add properties defined by the NISO/ALPSP Journal Article Versions schema [1].

```

1592 \newcommand*{\hyxmp@jav@schema}{%
1593   \hyxmp@add@simple{jav:journal_article_version}{\@pdfpubstatus}%
1594 }

```

### 3.6.12 The XMP Paged-Text schema

`\hyxmp@xmptpg@schema` The XMP Paged-Text schema [5] includes properties related to the construction of the PDF file itself. We acquire most of this information through LuaTeX mechanisms and therefore include much less information when run from other TeX engines.

```

1595 \newcommand*{\hyxmp@xmptpg@schema}{%
1596   \ifLuaTeX
1597     \luadirect{write_xmp_font_list(\the\hyxmp@cct)}%
1598   \fi
1599   \hyxmp@add@simple{xmpTPg:NPages}{\@pdfnumpages}%
1600 }

```

`\hyxmp@cct` We store the current category-code table to ensure that `write_xmp_font_list`'s output uses our redefined category codes.

```

1601 \ifLuaTeX
1602   \newcatcodetable\hyxmp@cct
1603   \savecatcodetable\hyxmp@cct
1604 \fi

```

`\hyxmp@prot@us` Define an underscore character that's protected from being converted into a space when passed to `\hyxmp@add@to@xml`. `\hyxmp@prot@us` is used within `write_xmp_font_list` (below) in particular to typeset filenames that may contain underscores.

```

1605 \bgroup
1606   \catcode'\_ =11
1607   \gdef\hyxmp@prot@us{_%}
1608 \egroup

```

Here we define a Lua function, `write_xmp_font_list`, that writes font information to the XMP packet.

```

1609 \ifLuaTeX
1610   \begin{luacode*}
1611   function write_xmp_font_list (cct)
1612     local function show_field(name, ...)
1613       for i = 1, select("#", ...) do
1614         local val = select(i, ...)
1615         if val then
1616           local xml = string.gsub(val, "&", "&")
1617           xml = string.gsub(xml, "<", "<")
1618           xml = string.gsub(xml, ">", ">")
1619           xml = string.gsub(xml, "_", "\\hyxmp@prot@us ")
1620           tex.print(cct, "-----<stFnt:" .. name .. ">" ..
1621             xml .. "</stFnt:" .. name .. ">^^J%")
1622         end
1623       end
1624     end
1625   end
1626   tex.print(cct, "\\hyxmp@add@to@xml{")
1627   tex.print(cct, "-----<xmpTPg:Fonts>^^J%")
1628   tex.print(cct, "-----<rdf:Bag>^^J%")
1629   for i, f in font.each() do
1630     tex.print(cct, "-----<rdf:li rdf:parseType=\"Resource\">^^J%")
1631     if f.filename then
1632       local fname = string.gsub(f.filename, "~harfloaded:(.*)", "%1")
1633       local info = fontloader.info(fname)
1634       if info then
1635         show_field("fontFace", info.fullname)
1636         show_field("fontFamily", info.familyname)
1637         show_field("fontName", info.fontname)
1638         show_field("versionString", info.version)
1639       end
1640       local baseName = string.gsub(f.filename, ".*[/\\](.*)", "%1")
1641       show_field("fontFileName", baseName)
1642     else
1643       show_field("fontName", f.psname, f.fullname, f.name)
1644     end
1645     if f.format and f.format ~= "unknown" then
1646       show_field("fontType", f.format)
1647     end
1648     tex.print(cct, "-----</rdf:li>^^J%")
1649   end
1650   tex.print(cct, "-----</rdf:Bag>^^J%")
1651   tex.print(cct, "-----</xmpTPg:Fonts>^^J%")
1652   tex.print(cct, "}")
1653 end
1654 \end{luacode*}
1655 \fi

```

### 3.6.13 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 [14]. 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
\hyxmp@iptc@data by the document.
1656 \newcommand*{\hyxmp@check@iptc@data}{%
1657   \edef\hyxmp@iptc@data{%
1658     \@pdfcontactaddress
1659     \@pdfcontactcity
1660     \@pdfcontactregion
1661     \@pdfcontactpostcode
1662     \@pdfcontactcountry
1663     \@pdfcontactphone
1664     \@pdfcontactemail
1665     \@pdfcontacturl
1666   }%
1667 }%

\hyxmp@check@prism@data Define \hyxmp@prism@data as the concatenation of all PRISM metadata supplied
\hyxmp@prism@data by the document.
1668 \newcommand*{\hyxmp@check@prism@data}{%
1669   \edef\hyxmp@prism@data{%
1670     \@pdfbookedition
1671     \@pdfbytes
1672     \@pdfdoi
1673     \@pdfeiissn
1674     \@pdfisbn
1675     \@pdfissn
1676     \@pdfissuenum
1677     \@pdfnumpages
1678     \@pdfpagerange
1679     \@pdfpublication
1680     \@pdfpubtype
1681     \@pdfsubtitle
1682     \@pdfurl
1683     \@pdfvolumenum
1684   }%
1685 }%

\hyxmp@check@jav@data Define \hyxmp@jav@data as the concatenation of all JAV metadata supplied by the
\hyxmp@jav@data document.
1686 \newcommand*{\hyxmp@check@jav@data}{%
1687   \edef\hyxmp@jav@data{%
1688     \@pdfpubstatus
```

```

1689 }%
1690 }

\hyxmp@begin@extension@decls  Begin a block of XML tags that indicates we're declaring one or more extension
                                schemata.
1691 \newcommand*{\hyxmp@begin@extension@decls}{%
1692   \hyxmp@add@to@xml{%
1693     <pdfaExtension:schemas>^^J%
1694     <rdf:Bag>^^J%
1695   }%
1696 }

\hyxmp@end@extension@decls  End the block of XML tags begun by \hyxmp@begin@extension@decls.
1697 \newcommand*{\hyxmp@end@extension@decls}{%
1698   \hyxmp@add@to@xml{%
1699     </rdf:Bag>^^J%
1700     </pdfaExtension:schemas>^^J%
1701   }%
1702 }

\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.
1703 \newcommand*{\hyxmp@begin@ext@decl}[3]{%
1704   \hyxmp@add@to@xml{%
1705     <rdf:li rdf:parseType="Resource">^^J%
1706     <pdfaSchema:schema>#1</pdfaSchema:schema>^^J%
1707     <pdfaSchema:prefix>#2</pdfaSchema:prefix>^^J%
1708     <pdfaSchema:namespaceURI>#3</pdfaSchema:namespaceURI>^^J%
1709     <pdfaSchema:property>^^J%
1710     <rdf:Seq>^^J%
1711   }%
1712 }%

\hyxmp@end@ext@decl  End the declaration of a single extension schema.
1713 \newcommand*{\hyxmp@end@ext@decl}{%
1714   \hyxmp@add@to@xml{%
1715     </rdf:Seq>^^J%
1716     </pdfaSchema:property>^^J%
1717     </rdf:li>^^J%
1718   }%
1719 }%

\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.
1720 \newcommand{\hyxmp@declare@property}[4][Text]{%
1721   \hyxmp@add@to@xml{%
1722     <rdf:li rdf:parseType="Resource">^^J%
1723     <pdfaProperty:name>%

```



```

1724 \xdef\hyxmp@xml{\hyxmp@xml#2}%
1725 \hyxmp@add@to@xml{</pdfaProperty:name>^^J%
1726 -----<pdfaProperty:valueType>#1</pdfaProperty:valueType>^^J%
1727 -----<pdfaProperty:category>#3</pdfaProperty:category>^^J%
1728 -----<pdfaProperty:description>#4</pdfaProperty:description>^^J%
1729 -----</rdf:li>^^J%
1730 }%
1731 }%

```

**\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.

```

1732 \newcommand{\hyxmp@declare@field}[3][Text]{%
1733 \hyxmp@add@to@xml{%
1734 -----<rdf:li rdf:parseType="Resource">^^J%
1735 -----<pdfaField:name>#2</pdfaField:name>^^J%
1736 -----<pdfaField:valueType>#1</pdfaField:valueType>^^J%
1737 -----<pdfaField:description>#3</pdfaField:description>^^J%
1738 -----</rdf:li>^^J%
1739 }%
1740 }

```

**\hyxmp@pdf@extensions** Declare the Adobe PDF schema.

```

1741 \newcommand*{\hyxmp@pdf@extensions}{%
1742 \hyxmp@begin@ext@decl
1743 {Adobe PDF Schema}%
1744 {pdf}%
1745 {http://ns.adobe.com/pdf/1.3/}%
1746 \hyxmp@declare@property
1747 {Trapped}%
1748 {internal}%
1749 {Indication if the document has been modified to include trapping information}%
1750 \hyxmp@end@ext@decl
1751 }%

```

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

```

1752 \newcommand*{\hyxmp@mm@extensions}{%
1753 \hyxmp@begin@ext@decl
1754 {XMP Media Management Schema}%
1755 {xmpMM}%
1756 {http://ns.adobe.com/xap/1.0/mm/}%
1757 \hyxmp@declare@property
1758 [URI]
1759 {DocumentID}%
1760 {internal}%
1761 {UUID based identifier for all versions and renditions of a document}%
1762 \hyxmp@declare@property
1763 [URI]
1764 {InstanceID}%

```

```

1765         {internal}}%
1766         {UUID based identifier for specific incarnation of a document}}%
1767 \hyxmp@declare@property
1768     {VersionID}}%
1769     {internal}}%
1770     {Document version identifier}}%
1771 \hyxmp@declare@property
1772     [RenditionClass}}%
1773     {RenditionClass}}%
1774     {internal}}%
1775     {The manner in which a document is rendered}}%
1776 \hyxmp@end@ext@decl
1777 }%

```

\hyxmp@pdfa@id@extensions Declare the PDF/A Identification schema [13].

```

1778 \newcommand*{\hyxmp@pdfa@id@extensions}{%
1779 \hyxmp@begin@ext@decl
1780     {PDF/A Identification Schema}}%
1781     {pdfaid}}%
1782     {http://www.aiim.org/pdfa/ns/id/}}%
1783 \hyxmp@declare@property
1784     [Integer}}%
1785     {part}}%
1786     {internal}}%
1787     {Part of PDF/A standard}}%
1788 \hyxmp@declare@property
1789     {conformance}}%
1790     {internal}}%
1791     {Conformance level of PDF/A standard}}%
1792 \hyxmp@end@ext@decl
1793 }%

```

\hyxmp@pdfua@id@extensions Declare the PDF/UA Universal Accessibility schema.

```

1794 \newcommand*{\hyxmp@pdfua@id@extensions}{%
1795 \hyxmp@begin@ext@decl
1796     {PDF/UA Universal Accessibility Schema}}%
1797     {pdfuaid}}%
1798     {http://www.aiim.org/pdfua/ns/id/}}%
1799 \hyxmp@declare@property
1800     [Integer}}%
1801     {part}}%
1802     {internal}}%
1803     {Part of ISO 14289 standard}}%
1804 \hyxmp@end@ext@decl
1805 }%

```

\hyxmp@pdfx@id@extensions Declare the schema used pre-PDF/X-4. Because Adobe Acrobat DC (at least) defines this even for PDF/X-4 and later, we follow suit.

```

1806 \newcommand*{\hyxmp@pdfx@id@extensions}{%

```

```

1807 \ifx\hyxmp@pdfx@major\empty
1808 \else
1809   \hyxmp@begin@ext@decl
1810     {Adobe Document Info PDF/X eXtension Schema}%
1811     {pdfx}%
1812     {http://ns.adobe.com/pdfx/1.3/}%
1813   \hyxmp@declare@property
1814     {GTS_PDFXVersion}%
1815     {internal}%
1816     {ID of PDF/X standard}%
1817   \hyxmp@declare@property
1818     {GTS_PDFXConformance}%
1819     {internal}%
1820     {Conformance level of PDF/X standard}%
1821   \hyxmp@end@ext@decl
1822 \fi

```

Declare the schema used in PDF/X-4 and later versions.

```

1823 \@tempcnta=0\hyxmp@pdfx@major\relax
1824 \ifnum\@tempcnta>3
1825   \hyxmp@begin@ext@decl
1826     {PDF/X ID Schema}%
1827     {pdfxid}%
1828     {http://www.npes.org/pdfx/ns/id/}%
1829   \hyxmp@declare@property
1830     {GTS_PDFXVersion}%
1831     {internal}%
1832     {ID of PDF/X standard}%
1833   \hyxmp@end@ext@decl
1834 \fi
1835 }%

```

`\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.

```

1836 \newcommand*{\hyxmp@iptc@extensions}{%
1837   \hyxmp@begin@ext@decl
1838     {IPTC Core Schema}%
1839     {Iptc4xmpCore}%
1840     {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}%
1841   \hyxmp@declare@property
1842     [ContactInfo]
1843     {CreatorContactInfo}
1844     {external}
1845     {Document creator's contact information}

```

We can't call `\hyxmp@end@ext@decl` because we need first need to define the `iptc4xmpCore:ContactInfo` structure.

```

1846   \hyxmp@add@to+xml{%

```

```

1847 -----</rdf:Seq>^^J%
1848 -----</pdfaSchema:property>^^J%
1849 -----<pdfaSchema:valueType>^^J%
1850 -----<rdf:Seq>^^J%
1851 -----<rdf:li rdf:parseType="Resource">^^J%
1852 -----<pdfaType:type>ContactInfo</pdfaType:type>^^J%
1853 -----<pdfaType:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaType:
1854 -----<pdfaType:prefix>Iptc4xmpCore</pdfaType:prefix>^^J%
1855 -----<pdfaType:description>%
1856         Basic set of information to get in contact with a person%
1857 -----</pdfaType:description>^^J%
1858 -----<pdfaType:field>^^J%
1859 -----<rdf:Seq>^^J%
1860 }%
1861 \hyxmp@declare@field
1862     {CiAdrCity}%
1863     {Contact information city}%
1864 \hyxmp@declare@field
1865     {CiAdrCtry}%
1866     {Contact information country}%
1867 \hyxmp@declare@field
1868     {CiAdrExtadr}%
1869     {Contact information address}%
1870 \hyxmp@declare@field
1871     {CiAdrPcode}%
1872     {Contact information local postal code}%
1873 \hyxmp@declare@field
1874     {CiAdrRegion}%
1875     {Contact information regional information such as state or province}%
1876 \hyxmp@declare@field
1877     {CiEmailWork}%
1878     {Contact information email address(es)}%
1879 \hyxmp@declare@field
1880     {CiTelWork}%
1881     {Contact information telephone number(s)}%
1882 \hyxmp@declare@field
1883     {CiUrlWork}%
1884     {Contact information Web URL(s)}%
1885 \hyxmp@add@to+xml{%
1886 -----</rdf:Seq>^^J%
1887 -----</pdfaType:field>^^J%
1888 -----</rdf:li>^^J%
1889 -----</rdf:Seq>^^J%
1890 -----</pdfaSchema:valueType>^^J%
1891 -----</rdf:li>^^J%
1892 }%
1893 }

```

\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.

```

1894 \newcommand*{\hyxmp@prism@extensions}{%
1895   \hyxmp@begin@ext@decl
1896     {PRISM Basic Metadata}%
1897     {prism}%
1898     {http://prismstandard.org/namespaces/basic/3.0/}%
1899   \hyxmp@declare@property
1900     {complianceProfile}%
1901     {internal}%
1902     {PRISM specification compliance profile to which this document adheres}%
1903   \hyxmp@declare@property
1904     {publicationName}%
1905     {external}%
1906     {Publication name}%
1907   \hyxmp@declare@property
1908     {aggregationType}%
1909     {external}%
1910     {Publication type}%
1911   \hyxmp@declare@property
1912     {bookEdition}%
1913     {external}%
1914     {Edition of the book in which the document was published}%
1915   \hyxmp@declare@property
1916     {volume}%
1917     {external}%
1918     {Publication volume number}%
1919   \hyxmp@declare@property
1920     {number}%
1921     {external}%
1922     {Publication issue number within a volume}%
1923   \hyxmp@declare@property
1924     {pageRange}%
1925     {external}%
1926     {Page range for the document within the print version of its publication}%
1927   \hyxmp@declare@property
1928     {issn}%
1929     {external}%
1930     {ISSN for the printed publication in which the document was published}%
1931   \hyxmp@declare@property
1932     {eIssn}%
1933     {external}%
1934     {ISSN for the electronic publication in which the document was published}%
1935   \hyxmp@declare@property
1936     {isbn}%
1937     {external}%
1938     {ISBN for the publication in which the document was published}%
1939   \hyxmp@declare@property
1940     {doi}%

```

```

1941         {external}%
1942         {Digital Object Identifier for the document}%
1943 \hyxmp@declare@property
1944     [URL]
1945     {url}%
1946     {external}%
1947     {URL at which the document can be found}%
1948 \hyxmp@declare@property
1949     [Integer]
1950     {byteCount}%
1951     {internal}%
1952     {Approximate file size in octets}%
1953 \hyxmp@declare@property
1954     [Integer]
1955     {pageCount}%
1956     {internal}%
1957     {Number of pages in the print version of the document}%
1958 \hyxmp@declare@property
1959     {subtitle}%
1960     {external}%
1961     {Document's subtitle}%
1962 \hyxmp@end@ext@decl
1963 }%

```

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

```

1964 \newcommand*{\hyxmp@jav@extensions}{%
1965     \hyxmp@begin@ext@decl
1966     {NISO/ALPSP Journal Article Versions}%
1967     {jav}%
1968     {http://www.niso.org/schemas/jav/1.0/}%
1969 \hyxmp@declare@property
1970     [Closed Choice of Text]%
1971     {journal_article_version}%
1972     {external}%
1973     {Article status, one of "A0", "SMUR", "AM", "P", "VoR", "CVoR", or "EVoR"}%
1974 \hyxmp@end@ext@decl
1975 }%

```

\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.

```

1976 \newcommand*{\hyxmp@declare@extensions}{%
1977     \hyxmp@begin@extension@decls
1978     Declare the Adobe PDF schema (always present).
1979     \hyxmp@pdf@extensions
1980     Declare the XMP Media Management extensions (always present).

```

```

1979 \hyxmp@mm@extensions
    Declare the PDF/A Identification extensions, but only when generating a PDF/A
    document.
1980 \ifHy@pdfa
1981 \hyxmp@pdfa@id@extensions
1982 \fi

    Conditionally declare the PDF/UA Universal Accessibility extensions.
1983 \ifx\@pdfuapart\@empty
1984 \else
1985 \hyxmp@pdfua@id@extensions
1986 \fi

\next Conditionally declare the PDF/X extensions.
1987 \ifx\@pdfxversion\@empty
1988 \else
1989 \hyxmp@pdfx@id@extensions
1990 \fi

    Conditionally declare IPTC photo metadata extensions.
1991 \ifx\hyxmp@iptc@data\@empty
1992 \else
1993 \hyxmp@iptc@extensions
1994 \fi

    Conditionally declare PRISM basic metadata extensions.
1995 \ifx\hyxmp@prism@data\@empty
1996 \else
1997 \hyxmp@prism@extensions
1998 \fi

    Conditionally declare JAV metadata.
1999 \ifx\hyxmp@jav@data\@empty
2000 \else
2001 \hyxmp@jav@extensions
2002 \fi

2003 \hyxmp@end@extension@decls
2004 }

```

### 3.6.14 Combining schemata into an XMP packet

```

\hyxmp@bom Define a macro for the Unicode byte-order marker (BOM).
2005 \begingroup
2006 \ifhyxmp@unicodetex
2007 \lccode'\!="FEFF %
2008 \lowercase{%
2009 \gdef\hyxmp@bom{!}%
2010 }%
2011 \else

```

```

2012 \catcode'\^^ef=12
2013 \catcode'\^^bb=12
2014 \catcode'\^^bf=12
2015 \gdef\hyxmp@bom{^^ef^^bb^^bf}%
2016 \fi
2017 \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.

```

2018 \def\hyxmp@construct@packet{%
2019 \gdef\hyxmp+xml{%
2020 \hyxmp@add@to+xml{<?xpacket begin="\hyxmp@bom" %
2021 id="W5M0MpCehiHzreSzNTczkc9d"?>^^J%
2022 <x:xmpmeta xmlns:x="adobe:ns:meta/">^^J%
2023 __<rdf:RDF %
2024 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\hyxmp@hash">^^J%
2025 ____<rdf:Description rdf:about="^^J%

```

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

```

2026 -----xmlns:pdf="http://ns.adobe.com/pdf/1.3/"^^J%
2027 -----xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/"^^J%
2028 -----xmlns:dc="http://purl.org/dc/elements/1.1/"^^J%
2029 -----xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/"^^J%
2030 -----xmlns:xmp="http://ns.adobe.com/xap/1.0/"^^J%
2031 -----xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"^^J%
2032 -----xmlns:stEvt="http://ns.adobe.com/xap/1.0/sType/ResourceEvent\hyxmp@hash"^^J%
2033 -----xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/"^^J%
2034 -----xmlns:pdfuaid="http://www.aiim.org/pdfua/ns/id/"^^J%
2035 -----xmlns:pdfx="http://ns.adobe.com/pdfx/1.3/"^^J%
2036 -----xmlns:pdfxid="http://www.npes.org/pdfx/ns/id/"^^J%
2037 -----xmlns:prism="http://prismstandard.org/namespaces/basic/3.0/"^^J%
2038 -----xmlns:jav="http://www.niso.org/schemas/jav/1.0/"^^J%
2039 -----xmlns:xmpTPg="http://ns.adobe.com/xap/1.0/t/pg/"^^J%
2040 -----xmlns:stFnt="http://ns.adobe.com/xap/1.0/sType/Font\hyxmp@hash"^^J%
2041 -----xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"^^J%
2042 -----xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"^^J%
2043 -----xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema\hyxmp@hash"^^J%
2044 -----xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property\hyxmp@hash"^^J%
2045 -----xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type\hyxmp@hash"^^J%
2046 -----xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field\hyxmp@hash">^^J%
2047 }%

```

Declare non-standard schemata.

```

2048 \hyxmp@check@iptc@data
2049 \hyxmp@check@prism@data
2050 \hyxmp@check@jav@data
2051 \hyxmp@declare@extensions

```

Insert all the metadata we know how to insert.

```

2052 \hyxmp@pdf@schema

```



```

2053 \hyxmp@xmpRights@schema
2054 \hyxmp@dc@schema
2055 \hyxmp@photoshop@schema
2056 \hyxmp@xmp@basic@schema
2057 \hyxmp@pdfa@id@schema
2058 \hyxmp@pdfua@id@schema
2059 \hyxmp@pdfx@id@schema
2060 \hyxmp@mm@schema
2061 \hyxmp@iptc@schema
2062 \hyxmp@prism@schema
2063 \hyxmp@jav@schema
2064 \hyxmp@xmptpg@schema
2065 \hyxmp@addto@xml{%
2066 ____</rdf:Description>^^J%
2067 __</rdf:RDF>^^J%
2068 </x:xmpmeta>^^J%
2069 \hyxmp@padding
2070 <?xpacket end="w"?>^^J%
2071 }%
2072 }

```

### 3.7 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” [4] so that’s what we do here.

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

```

2073 \newcommand*{\hyxmp@embed@packet}{%
2074 \hyxmp@construct@packet
2075 \def\hyxmp@driver{hpdftex}%
2076 \ifx\hyxmp@driver\Hy@driver
2077 \hyxmp@embed@packet@pdftex
2078 \else
2079 \def\hyxmp@driver{hluatex}%
2080 \ifx\hyxmp@driver\Hy@driver
2081 \hyxmp@embed@packet@luatex
2082 \else
2083 \def\hyxmp@driver{hdvipdfm}%
2084 \ifx\hyxmp@driver\Hy@driver
2085 \hyxmp@embed@packet@dvipdfm
2086 \else
2087 \def\hyxmp@driver{hxetex}%
2088 \ifx\hyxmp@driver\Hy@driver
2089 \hyxmp@embed@packet@xetex
2090 \else
2091 \@ifundefined{pdfmark}{%
2092 \PackageWarningNoLine{hyperxmp}{%

```

```

2093             Unrecognized hyperref driver ‘\Hy@driver’.\MessageBreak
2094             \hyxmp@jobname.tex’s XMP metadata will *not* be\MessageBreak
2095             embedded in the resulting file}%
2096         }{%
2097             \hyxmp@embed@packet@pdfmark
2098         }%
2099     \fi
2100 \fi
2101 \fi
2102 \fi
2103 }

```

### 3.7.1 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.

```

2104 \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.

```

2105 \newcommand*{\hyxmp@embed@packet@pdftex}{%
2106     \bgroup
2107     \ifluatex
2108     \else
2109         \pdfcompresslevel=0
2110     \fi
2111     \immediate\pdfobj \ifluatex uncompressed\fi stream attr {%
2112         /Type /Metadata
2113         /Subtype /XML
2114     }{\hyxmp@xml}%
2115     \pdfcatalog {/Metadata \the\pdflastobj\space 0 R}%
2116 \egroup
2117 }

```

### 3.7.2 Embedding using LuaTeX 0.85+

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

```
2118 \newcommand*{\hyxmp@embed@packet@luatex}{%
2119 \immediate\pdfextension obj uncompressed stream attr {%
2120     /Type /Metadata
2121     /Subtype /XML
2122 }{\hyxmp@xml}%
2123 \pdfextension catalog {/Metadata \the\numexpr\pdffeedback lastobj\relax\space 0 R}%
2124 }
```

### 3.7.3 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.

```
2125 \newcommand*{\hyxmp@embed@packet@pdfmark}{%
2126 \pdfmark{%
2127     pdfmark=/NamespacePush
2128 }%
2129 \pdfmark{%
2130     pdfmark=/OBJ,
2131     Raw={/_objdef \string{hyxmp@Metadata\string} /type /stream}%
2132 }%
2133 \pdfmark{%
2134     pdfmark=/PUT,
2135     Raw={\string{hyxmp@Metadata\string}
2136         2 dict begin
2137             /Type /Metadata def
2138             /Subtype /XML def
2139             currentdict
2140         end
2141     }%
2142 }%
2143 \pdfmark{%
2144     pdfmark=/PUT,
2145     Raw={\string{hyxmp@Metadata\string} (\hyxmp@xml)}%
2146 }%
2147 \pdfmark{%
2148     pdfmark=/Metadata,
2149     Raw={\string{Catalog\string} \string{hyxmp@Metadata\string}}%
2150 }%
2151 \pdfmark{%
2152     pdfmark=/NamespacePop
2153 }%
2154 }
```

### 3.7.4 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.

```

2155 \newcommand*{\hyxmp@embed@packet@dvipdfm}{%
2156   \hyxmp@string@len{\hyxmp@xml}%
2157   \special{pdf: object @hyxmp@Metadata
2158     <<
2159       /Type /Metadata
2160       /Subtype /XML
2161       /Length \the\@tempcnta
2162     >>
2163     stream^^J\hyxmp@xml endstream%
2164   }%
2165   \special{pdf: docview
2166     <<
2167       /Metadata @hyxmp@Metadata
2168     >>
2169   }%
2170 }
```

`\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.

```

2171 \newcommand*{\hyxmp@string@len}[1]{%
2172   \@tempcnta=0
2173   \expandafter\hyxmp@count@spaces#1 {} %
2174   \expandafter\hyxmp@count@non@spaces#1{}%
2175 }
```

`\hyxmp@count@spaces` Count the number of spaces in a given string. We rely on the built-in pattern matching of T<sub>E</sub>X's `\def` primitive to pry one word at a time off the head of the input string.

```

2176 \def\hyxmp@count@spaces#1 {%
2177   \def\hyxmp@one@token{#1}%
2178   \ifx\hyxmp@one@token\@empty
2179     \advance\@tempcnta by -1
2180   \else
2181     \advance\@tempcnta by 1
2182     \expandafter\hyxmp@count@spaces
2183   \fi
2184 }
```

`\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<sub>E</sub>X 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.

```

2185 \newcommand*{\hyxmp@count@non@spaces}[1]{%
2186   \def\hyxmp@one@token{#1}%
2187   \ifx\hyxmp@one@token\@empty
2188     \else
2189       \advance\@tempcnta by 1
2190       \expandafter\hyxmp@count@non@spaces
2191     \fi
2192 }

```

### 3.7.5 Embedding using X<sub>Y</sub>TeX

`\hyxmp@embed@packet@xetex` Embed the XMP packet using xdvipdfmx-specific `\special` commands. I don't know how to tell xdvipdfmx always to leave the Metadata stream uncompressed, so the XMP metadata is likely to be missed by non-PDF-aware XMP viewers.

```

2193 \newcommand*{\hyxmp@embed@packet@xetex}{%
2194   \special{pdf:stream @hyxmp@Metadata (\hyxmp+xml)
2195     <<
2196       /Type /Metadata
2197       /Subtype /XML
2198     >>
2199   }%
2200   \special{pdf:put @catalog
2201     <<
2202       /Metadata @hyxmp@Metadata
2203     >>
2204   }%
2205 }

```

## 3.8 Final clean-up

As explained in Section 3.1, all invocations of `\AtEndPreamble` have been stored in `\hyxmp@aep@toks` rather than executed. Now that `hyperxmp` has been initialized completely, it is finally safe to execute the accumulated `\AtEndPreamble` stanzas.

```

2206 \the\hyxmp@aep@toks

```

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.

```

2207 \catcode'\="=\hyxmp@dq@code

```

## 4 Help Wanted

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

## A Sample XMP Packet

The following is an example of a complete XMP packet as may be produced by hyperxmp. This packet corresponds to the metadata included in the sample L<sup>A</sup>T<sub>E</sub>X document presented on pages 10–11. 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:pdfuaid="http://www.aiim.org/pdfua/ns/id/"
      xmlns:pdfx="http://ns.adobe.com/pdfx/1.3/"
      xmlns:pdfxid="http://www.npes.org/pdfx/ns/id/"
      xmlns:prism="http://prismstandard.org/namespaces/basic/3.0/"
      xmlns:jav="http://www.niso.org/schemas/jav/1.0/"
      xmlns:xmpTPg="http://ns.adobe.com/xap/1.0/t/pg/"
      xmlns:stFnt="http://ns.adobe.com/xap/1.0/sType/Font#"
      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 200 lines of boilerplate definitions not shown]
        ⋮
      </rdf:Bag>
    </pdfaExtension:schemas>
    <pdf:Keywords>
      energy quanta, Hertz effect, quantum physics
    </pdf:Keywords>
    <pdf:Producer>
      LuaHBTeX, Version 1.12.0 (TeX Live 2020)
    </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="x-default">
      On a heuristic viewpoint concerning the production
      and transformation of light
    </rdf:li>
    <rdf:li xml:lang="en">
      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:Alt>
</dc:description>
<dc:rights>
  <rdf:Alt>
    <rdf:li xml:lang="en">Copyright (C) 1905, Albert Einstein</rdf:li>
  </rdf:Alt>
</dc:rights>
<dc:publisher>
  <rdf:Bag>
    <rdf:li>Wiley-VCH</rdf:li>
  </rdf:Bag>
</dc:publisher>
<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>
<dc:identifier>info:lccn/50013519</dc:identifier>
<photoshop:AuthorsPosition>
    Technical Assistant, Level III
</photoshop:AuthorsPosition>
<photoshop:CaptionWriter>Scott Pakin</photoshop:CaptionWriter>
<xmp:CreateDate>2020-07-25T21:37:02-06:00</xmp:CreateDate>
<xmp:ModifyDate>2020-07-25T21:37:02-06:00</xmp:ModifyDate>
<xmp:MetadataDate>2020-07-25T21:37:02-06:00</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>
<xmpMM:RenditionClass>default</xmpMM:RenditionClass>
<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/190517132-148.pdf
</prism:url>
<prism:byteCount>41197</prism:byteCount>
<prism:pageCount>1</prism:pageCount>
<jav:journal_article_version>VoR</jav:journal_article_version>
<xmpTPg:Fonts>
  <rdf:Bag>
    <rdf:li rdf:parseType="Resource">
      <stFnt:fontFace>LMRoman10-Regular</stFnt:fontFace>
      <stFnt:fontFamily>LM Roman 10</stFnt:fontFamily>
      <stFnt:fontName>LMRoman10-Regular</stFnt:fontName>
      <stFnt:versionString>
        2.004;PS 2.004;hotconv 1.0.49;makeotf.lib2.0.14853
      </stFnt:versionString>
      <stFnt:fontFileName>lmroman10-regular.otf</stFnt:fontFileName>
      <stFnt:fontType>opentype</stFnt:fontType>
    </rdf:li>
    <rdf:li rdf:parseType="Resource">
      <stFnt:fontFace>LMRoman17-Regular</stFnt:fontFace>
      <stFnt:fontFamily>LM Roman 17</stFnt:fontFamily>
      <stFnt:fontName>LMRoman17-Regular</stFnt:fontName>
      <stFnt:versionString>
        2.004;PS 2.004;hotconv 1.0.49;makeotf.lib2.0.14853
      </stFnt:versionString>
      <stFnt:fontFileName>lmroman17-regular.otf</stFnt:fontFileName>
      <stFnt:fontType>opentype</stFnt:fontType>
    </rdf:li>
    <rdf:li rdf:parseType="Resource">
      <stFnt:fontFace>LMRoman12-Regular</stFnt:fontFace>
      <stFnt:fontFamily>LM Roman 12</stFnt:fontFamily>
      <stFnt:fontName>LMRoman12-Regular</stFnt:fontName>
      <stFnt:versionString>
        2.004;PS 2.004;hotconv 1.0.49;makeotf.lib2.0.14853
      </stFnt:versionString>
      <stFnt:fontFileName>lmroman12-regular.otf</stFnt:fontFileName>
      <stFnt:fontType>opentype</stFnt:fontType>
    </rdf:li>
    <rdf:li rdf:parseType="Resource">
      <stFnt:fontName>cmr12</stFnt:fontName>
    </rdf:li>
    <rdf:li rdf:parseType="Resource">
      <stFnt:fontName>cmr8</stFnt:fontName>
    </rdf:li>
  </rdf:Bag>

```

```

</rdf:li>
<rdf:li rdf:parseType="Resource">
  <stFnt:fontName>cmr6</stFnt:fontName>
</rdf:li>
<rdf:li rdf:parseType="Resource">
  <stFnt:fontName>cmmi12</stFnt:fontName>
</rdf:li>
<rdf:li rdf:parseType="Resource">
  <stFnt:fontName>cmmi8</stFnt:fontName>
</rdf:li>
<rdf:li rdf:parseType="Resource">
  <stFnt:fontName>cmmi6</stFnt:fontName>
</rdf:li>
<rdf:li rdf:parseType="Resource">
  <stFnt:fontName>cmsy10</stFnt:fontName>
</rdf:li>
<rdf:li rdf:parseType="Resource">
  <stFnt:fontName>cmsy8</stFnt:fontName>
</rdf:li>
<rdf:li rdf:parseType="Resource">
  <stFnt:fontName>cmsy6</stFnt:fontName>
</rdf:li>
<rdf:li rdf:parseType="Resource">
  <stFnt:fontName>cmex10</stFnt:fontName>
</rdf:li>
</rdf:Bag>
</xmpTPg:Fonts>
<xmpTPg:NPages>1</xmpTPg:NPages>
</rdf:Description>
</rdf:RDF>
</x:xmpmeta>
<?xpacket end="w"?>

```

## References

- [1] Beverley Acreman, Claire Bird, Catherine Jones, Peter McCracken, Cliff Morgan, John Ober, Evan Owens, T. Scott Plutchak, Bernie Rous, and Andrew Wray. Journal Article Versions (JAV): Recommendations of the NISO/ALPSP JAV Technical Working Group. Recommended practice, National Information Standards Organization, Baltimore, Maryland, USA, April 2008. ISBN 978-1-880124-79-6. Available from <https://www.niso.org/sites/default/files/2017-08/RP-8-2008.pdf>.
- [2] Adobe Systems, Inc., San Jose, California. *Adobe Acrobat X SDK Help, pdfmark Reference*. Available from <http://www.adobe.com/devnet/acrobat/documentation.html>.
- [3] Adobe Systems, Inc. *PostScript Language Reference Manual*. Addison-Wesley, 2nd edition, January 1996, ISBN: 0-201-18127-4.

- [4] Adobe Systems, Inc., San Jose, California. *Document Management—Portable Document Format—Part 1: PDF 1.7*, July 2008. ISO 32000-1 standard document. Available from [http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000\\_2008.pdf](http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000_2008.pdf).
- [5] 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>.
- [6] DCMI Usage Board *DCMI Metadata Terms*, June 14, 2012. Available from <http://dublincore.org/documents/dcmi-terms/>.
- [7] 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>.
- [8] 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](http://www.prismstandard.org/specifications/3.0/PRISM_Basic_Metadata_3.0.htm).
- [9] 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](http://www.prismstandard.org/specifications/3.0/PRISM_CV_Spec_3.0.pdf).
- [10] International Press Telecommunications Council. *IPTC Photo Metadata: Core 1.1/Extension 1.1*, July 2010. Revision 1. Available from [http://www.iptc.org/std/photometadata/specification/IPTC-PhotoMetadata-201007\\_1.pdf](http://www.iptc.org/std/photometadata/specification/IPTC-PhotoMetadata-201007_1.pdf).
- [11] Internet Assigned Numbers Authority. Language subtag registry, January 11, 2011. Available from <http://www.iana.org/assignments/language-subtag-registry>.
- [12] 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>.
- [13] PDF/A Competence Center, Berlin, Germany. *TechNote 0008: Predefined XMP Properties in PDF/A-1*, March 20, 2008. Available from [http://www.pdfa.org/wp-content/uploads/2011/08/tn0008\\_predefined\\_xmp\\_properties\\_in\\_pdfa-1\\_2008-03-20.pdf](http://www.pdfa.org/wp-content/uploads/2011/08/tn0008_predefined_xmp_properties_in_pdfa-1_2008-03-20.pdf).
- [14] PDF/A Competence Center, Berlin, Germany. *TechNote 0009: XMP Extension Schemas in PDF/A-1*, March 20, 2008. Available from [http://www.pdfa.org/wp-content/uploads/2011/08/tn0009\\_xmp\\_extension\\_schemas\\_in\\_pdfa-1\\_2008-03-20.pdf](http://www.pdfa.org/wp-content/uploads/2011/08/tn0009_xmp_extension_schemas_in_pdfa-1_2008-03-20.pdf).

- [15] 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		v2.0	
General: Initial version	1	\ProcessKeyvalOptions: Added	
v1.1		this macro	31
\hyxmp@construct@packet:		\XMPTruncateList: Added this	
Explicitly set the category		macro	46
codes of characters $\langle EF \rangle$ , $\langle BB \rangle$ ,		\hyxmp@ProcessKeyvalOptions:	
and $\langle BF \rangle$ to “letter”. Thanks to		Added this macro	31
Daniel Schömer for the bug		\hyxmp@SpaceOther: Added by	
report	88	Heiko Oberdiek	56
v1.2		\hyxmp@add@simple: Added this	
General: Added support for the		macro	58
X <sub>Y</sub> TEX backend (xdvipdfmx)	1	\hyxmp@add@to@xml: Updated also	
Added support for the		to replace commas	64
Photoshop schema	1	\hyxmp@bom: Added by Heiko	
Made the package compatible		Oberdiek	87
with ngerman. Thanks to		\hyxmp@comma: Added this macro	45
Tobias Mueller for the bug		\hyxmp@construct@packet:	
report.	20	Modified by Heiko Oberdiek to	
v1.3		use an appropriate BOM	
General: Introduced the		representation via \hyxmp@bom	88
pdfmetalang package option,		\hyxmp@crap@convert: Added by	
which enables an author to		Heiko Oberdiek	57
specify the language in which he		\hyxmp@crap@test: Added by	
wrote the document’s metadata	35	Heiko Oberdiek	56
v1.4		\hyxmp@dc@schema: Added support	
\hyxmp@mm@schema: Renamed the		for dc:language and dc:source	71
xapMM namespace prefix to		\hyxmp@is@unicode: Added by	
xmpMM	72	Heiko Oberdiek	54
\hyxmp@rdf@dc: Included metadata		\hyxmp@list@to@xml: Modified by	
in the x-default language		Heiko Oberdiek to use the new	
regardless of the specified		Unicode-processing macros	69
metadata language	67	\hyxmp@photoshop@schema:	
\hyxmp@xmpRights@schema:		Simplified using	
Renamed the xapRights		\hyxmp@add@simple	73
namespace prefix to xmpRights	71	\hyxmp@skiptorelax: Added by	
v1.5		Heiko Oberdiek	57
General: Made the XMP inclusion		\hyxmp@skipzeros: Added by	
more robust. Thanks to Heiko		Heiko Oberdiek	55
Oberdiek for the bug report		\hyxmp@toxml: Added by Heiko	
and suggested modifications.	20	Oberdiek	54

Escaped parentheses written with <code>pdfmarks</code> to prevent dvips from line-wrapping the XMP packet . . . . .	55	v2.2	<code>\hyxmp@iptc@extensions</code> : Added this macro to support PDF/A generation . . . . .	83
<code>\hyxmp@toxml@unicodetex</code> : Added by Heiko Oberdiek . . . . .	55		<code>\hyxmp@iptc@schema</code> : Added this macro . . . . .	75
<code>\hyxmp@xetex@crap</code> : Added by Heiko Oberdiek . . . . .	56		<code>\hyxmp@list@to@lines</code> : Added this macro . . . . .	75
<code>\hyxmp@xmlify</code> : Completely rewritten by Heiko Oberdiek to better support Unicode-enabled TeX programs . . . . .	53		<code>\xmpcomma</code> : Changed the default from <code>\relax</code> to an ordinary comma . . . . .	45
<code>\hyxmp@xmp@basic@schema</code> : Added this macro . . . . .	73		<code>\xmplinesep</code> : Added this macro . .	75
<code>\hyxmp@xmpRights@schema</code> : Modified to include <code>xmpRights:Marked</code> only when <code>pdfcopyright</code> is specified and <code>xmpRights:WebStatement</code> only when <code>pdflicenseurl</code> is specified .	71		General: Added support for the IPTC Photo Metadata schema .	1
<code>\hyxmp@zero</code> : Added by Heiko Oberdiek . . . . .	57	v2.3	<code>\hyxmp@iptc@extensions</code> : Gave the <code>lptc4xmpCore:CreatorContactInfo</code> fields a unique <code>pdfaType:prefix</code> to better support conversion of the document to PDF/A . . . . .	83
<code>\ifhyxmp@unicodetex</code> : Added by Heiko Oberdiek . . . . .	52	v2.3a	<code>\hyxmp@detect@langs</code> : Bug fix: Redefine <code>\@pdflang</code> as <code>\@empty</code> when <code>hyperref</code> has set it to <code>\relax</code> . . . . .	43
<code>\xmpcomma</code> : Added this macro . .	45			
<code>\xmpquote</code> : Added this macro . .	45	v2.3b	<code>\XMPTruncateList</code> : Made all definitions local to avoid spurious <code>Too many unprocessed floats</code> errors when running with <code>memoir</code> . .	46
General: Added support for the XMP Basic schema and miscellaneous other bits of metadata . . . . .	1	v2.4	<code>\hyxmp@add@simple@var</code> : Added this macro . . . . .	58
Heiko Oberdiek's major rewrite of the code to better support native-Unicode TeX implementations (XeTeX and LuaTeX) . . . . .	1		<code>\hyxmp@create@uuid</code> : Modified this macro to produce a proper version 4 (random or pseudorandom) UUID . . . . .	62
New <code>\AtBeginDocument</code> code from Heiko Oberdiek to properly encode <code>\@pdfmetalang</code> . . . . .	35		<code>\hyxmp@dc@schema</code> : Made <code>dc:language</code> a Bag instead of an individual item so as to conform to the latest XMP specifications, a detail identified by Florian Breitwieser . . . . .	71
v2.1			<code>\hyxmp@parse@time</code> : Added this macro . . . . .	47
<code>\hypersetup</code> : Added this macro . .	31		<code>\hyxmp@parse@tz</code> : Added this macro . . . . .	47
<code>\hyxmp@hypersetup</code> : Added this macro . . . . .	31			
<code>\hyxmp@redefine@Hyp</code> : Added this macro . . . . .	30			
General: Enabled <code>hyperxmp</code> and <code>hyperref</code> to be loaded in either order. This addresses a bug report by Yury Donskoy . . . .	29			

\hyxmp@parse@tz@char: Added this macro . . . . .	47	Thanks to Maciej Radziejewski for the suggestion . . . . .	37
\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 . . . . .	67	v2.8 \hyxmp@add@to+xml: Corrected inadvertent lowercasing of non-Latin characters when run under X <sub>Y</sub> LaTeX or LuaLaTeX (bug reported by Leonid Sinev) . . . . .	64
\hyxmp@pdf@to@xmp@date: Added this macro . . . . .	47	v2.9 \hyxmp@iptc@schema: Use lptc4xmpCore instead of lptc4ContInfo as the contact-information metadata prefix. Leonid Sinev reports that Acrobat's PDF/A validator seems to prefer lptc4xmpCore . . . . .	75
\hyxmp@pdfa@id@schema: Added this macro . . . . .	74	\hyxmp@pdfa@id@schema: Let the author specify the PDF/A part and conformance IDs, as requested by Leonid Sinev . . . . .	74
\hyxmp@today@xmp: Modified the code to parse the time and timezone from \pdfcreationdate, as proposed by Florian Breitwieser . . . . .	50	General: Force inclusion of dc:creator, dc:title, and dc:description—even if empty—when hyperref is loaded with the pdfa option (suggested by Leonid Sinev) . . . . .	1
\hyxmp@today@xmp@define: Added this macro . . . . .	49	Introduced the pdftype package option, which enables an author to specify the type of document being produced . . . . .	1
\hyxmp@xmp@to@pdf@date: Added this macro . . . . .	48		
\xmp tilde: Added this macro . . . . .	46		
General: Added support for the PDF/A Identification schema, as requested by Florian Breitwieser . . . . .	1		
v2.5 \hyxmp@add@to+xml: Updated also to replace underscores and to modify only the text being added, not the already-modified text . . . . .	64	v3.0 \hyxmp@embed@packet@luatex: Added this macro . . . . .	91
\hyxmp@textunderscore: Added this macro . . . . .	22	\hyxmp@today@xmp@define: Modified to accept the name of a macro to define . . . . .	49
\hyxmp@uscore: Added this macro . . . . .	45	\hyxmp@xmp@basic@schema: Made the XMP xmp:CreateDate, xmp:ModifyDate, and xmp:MetadataDate match the PDF CreationDate . . . . .	73
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 hyperxmp code . . . . .	1
v2.6 General: Added support for a new pdfdate key to explicitly specify the document date (and optionally time) . . . . .	1		
v2.7 \hyxmp@auto@assign@data: Automatically use \title and \author if pdftitle and pdfauthor are left unspecified.		v3.1 \hyxmp@embed@packet@luatex:	

Updated to use <code>\pdfextension</code> obj uncompressed as suggested by Hans Hagen . . . . .	91	non-empty but all spaces . . . . .	1
<code>\hyxmp@embed@packet@pdftex:</code> Leave the XMP packet—and only the XMP packet—uncompressed in both pdfTeX and pre-0.85 LuaTeX . . . . .	90	v3.5 <code>\hyxmp@DocumentID:</code> Added the pdfdocumentid option, at Michael Osipov's request . . . . .	25
v3.2 <code>\hyxmp@as@pdf@date:</code> Added this macro . . . . .	47	<code>\hyxmp@InstanceID:</code> Added the pdfinstanceid option, at Michael Osipov's request . . . . .	25
<code>\hyxmp@as@xmp@date:</code> Added this macro . . . . .	46	<code>\hyxmp@mm@schema:</code> Generate <code>\hyxmp@DocumentID</code> and <code>\hyxmp@InstanceID</code> only if the document does not already define these using the pdfdocumentid and pdfinstanceid options . . . . .	72
<code>\hyxmp@today@xmp@define:</code> Modified to include hours and minutes . . . . .	49	<code>\hyxmp@seed@string:</code> Seed with the TeX timestamp in addition to the document-specified timestamp . . . . .	63
<code>\hyxmp@xmp@basic@schema:</code> Honor hyperref's pdfcreationdate and pdfmoddate options plus a new pdfmetadate option. Leonid Sinev requested this additional control and helped test the resulting hyperxmp code . . . . .	73	v4.0 <code>\XMPTruncateList:</code> Deprecated this macro . . . . .	46
v3.3 <code>@pdfsource:</code> Added this macro and the corresponding pdfsource option, at Niklas Beisert's request . . . . .	25	<code>\hyxmp@add@simple@lang:</code> Added this macro . . . . .	58
<code>\XMPLangAlt:</code> Added this macro based on a request—and some code—by Niklas Beisert to support metadata expressed in multiple languages . . . . .	60	<code>\hyxmp@begin@ext@decl:</code> Added this macro . . . . .	80
<code>\hyxmp@rdf@dc:</code> Bug fix: Output the metadata language as correct XML even when hyperref is loaded with the unicode option . . . . .	67	<code>\hyxmp@declare@field:</code> Replaced <code>\hyxmp@declare@resource</code> with this macro . . . . .	81
General: Don't overwrite an existing pdfmetalang with pdflang or x-default. This addresses a bug report by Niklas Beisert . . . . .	35	<code>\hyxmp@declare@property:</code> Added this macro . . . . .	80
v3.4 <code>\hyxmp@seed@string:</code> Correctly handle an author field of all spaces. Bug reported by Gaëtan Leurent . . . . .	63	<code>\hyxmp@end@ext@decl:</code> Added this macro . . . . .	80
General: Use ifmtarg to test for empty arguments, including		<code>\hyxmp@iptc@extensions:</code> Moved the header code from here into <code>\hyxmp@begin@extension@decls</code> and the trailer code from here into <code>\hyxmp@end@extension@decls</code> . . . . .	83
		Rewrote to more closely honor the XMP specification . . . . .	83
		<code>\hyxmp@iptc@schema:</code> Moved the definition of <code>\hyxmp@iptc@data</code> from here into <code>\hyxmp@check@iptc@data</code> . . . . .	75
		Renamed this macro to <code>\hyxmp@iptc@schema</code> from <code>\hyxmp@photometa@schema</code> . . . . .	75
		Rewrote this macro entirely to correct the use of fields within a	



structure	75	possible, only Author and	
\hyxmp@mm@extensions: Added		Keywords	29
this macro	81	\hyxmp@pdf@extensions: Added	
\hyxmp@mm@schema: Include		this macro	81
xmpMM:VersionID in the XMP		\hyxmp@pdf@schema: Honor	
packet	72	pdftrapped	67
\hyxmp@no@info@lists: Added		\hyxmp@pdfua@id@extensions:	
this macro	29	Added this macro	82
\hyxmp@pdfa@id@extensions:		\hyxmp@pdfua@id@schema: Added	
Added this macro	82	this macro	74
\hyxmp@prism@extensions: Added		\hyxmp@pdfx@id@extensions:	
this macro	84	Added this macro	82
\hyxmp@prism@schema: Added this		\hyxmp@pdfx@id@schema: Added	
macro	76	this macro	74
General: Include all metadata		\hyxmp@today@pdf: Added this	
within a single pdf:Description		macro	51
block	1	\hyxmp@today@xmp: Support	
v4.1		X <sub>Y</sub> TeX's \filemoddate	50
\hyxmp@aep@toks: Invoke		\hyxmp@today@xmp@define:	
\hyxmp@no@info@lists at the		Modified to specify UTC	49
beginning of the document, for		General: Added support for	
compatibility with both newer		PDF/UA standards, as requested	
and older versions of hyperref	34	by Robin Schwab	1
\hyxmp@singleton@dc: Added this		Added support for PDF/x	
macro	70	standards, as requested by	
General: Updated the		Robin Schwab	1
documentation to refer to		Define a default producer	66
\pdfnumpages by its correct		Don't set any document dates	
name. Thanks to Volker RW		(creation, modification, or	
Schaa for catching the		metadata) from pdfdate	1
discrepancy	1	v5.1	
v5.0		\hyxmp@banner@to@producer:	
\@pdfrendition: Added the		Prevent the category code of	
pdfrendition option	26	"@ " from propagating past the	
\@pdfxstandard: Added this macro	24	\begin{document}. Thanks to	
\hyxmp@add@simple: Insert the tag		Robert Schlicht for noticing this	
name (#1) verbatim	58	catcode "leak" and providing a	
\hyxmp@check@standards: Added		correction	66
this macro	33	\hyxmp@define@pdfproducer:	
\hyxmp@check@std: Added this		Check for LuaTeX before	
macro	24	checking for pdfTeX to work	
\hyxmp@declare@property: Insert		around luatex85's confusing	
the property name (#2)		iftex by defining	
verbatim	80	\pdftexversion. Thanks to	
\hyxmp@define@pdfproducer:		Robin Schwab for the bug	
Added this macro	65	report	65
\hyxmp@no@info@lists: Renamed		\hyxmp@timestamp: Don't rely on	
this macros from		\jobname.aux existing to query	
\hyxmp@suppress@pdf@metadata		the current time under X <sub>Y</sub> TeX.	
and rewrote it to replace, if		Instead, use \jobname.log.	



Thanks to Ulrike Fischer for the bug report and for her suggestion to use the log file. . . . .	51	<code>\hyxmp@parse@acmart</code> : Added this macro . . . . .	40
v5.2		<code>\hyxmp@set@koma@phones</code> : Added this macro . . . . .	35
<code>\hyxmp@add@simple@pfx</code> : Added this macro . . . . .	59	<code>\hyxmp@use@first@valid</code> : Added this macro . . . . .	36
<code>\hyxmp@assign@major@minor</code> : Added this macro. <code>hyperxmp</code> now correctly specifies <code>pdf:PDFVersion</code> when generating PDF 2.0+. Thanks to Ulrike Fischer for alerting me to PDF 2.0's availability in the T <sub>E</sub> X ecosystem and informing me how to activate it . . . . .	66	v5.4	
<code>\hyxmp@cond@dc@identifier</code> : Added this macro . . . . .	70	<code>\hyxmp@dc@schema</code> : Bug fix: Use <code>\hyxmp@today@xmp</code> as the date only if <code>\@pdfdatetime</code> is undefined . . . . .	71
<code>\next</code> : Define <code>\ifdraft</code> only locally, at Niklas Beisert's request . . . . .	25	<code>\hyxmp@detect@langs</code> : Added support for <code>babel</code> . . . . .	43
General: Introduced the <code>pdfidentifier</code> package option, which enables an author to specify a unique identifier for the document . . . . .	1	Refactored language detection into a separate command . . . . .	43
v5.3		<code>\hyxmp@parse@acmart</code> : Bug fix: Correct a missing "else" argument in two invocations of <code>\@if@def@and@nonempty</code> . . . . .	42
<code>\@if@def@and@nonempty</code> : Added this macro . . . . .	21	General: Moved the automatic assignment of <code>\@pdflang</code> and <code>\@pdfmetalang</code> from <code>\hyxmp@auto@assign@data</code> to within a call to <code>\hyxmp@at@end</code> . . . . .	35
<code>\hyxmp@at@end</code> : Use <code>\AtEndDocument</code> in all T <sub>E</sub> X back ends that provide it. Thanks to Nelson Posse Lago for pointing out why <code>atenddvi</code> is best avoided if possible . . . . .	20	v5.5	
<code>\hyxmp@auto@assign@data</code> : Consider other author-provided sources of metadata. Thanks to Robin Schwab for proposing that <code>hyperxmp</code> use the Koma letter classes's metadata . . . . .	37	<code>\hyxmp@aep@toks</code> : Copy <code>\title</code> to <code>pdftitle</code> and <code>\author</code> to <code>pdfauthor</code> at the start of the document to improve consistency between XMP and PDF metadata . . . . .	34
<code>\hyxmp@dc@schema</code> : Include all languages used in the document in <code>dc:language</code> . . . . .	71	Load <code>hyperref</code> automatically if the document does not do so explicitly, as requested by Robin Schwab . . . . .	34
<code>\hyxmp@detect@langs</code> : Acquire the default language from the <code>polyglossia</code> package, if loaded. Thanks to Robin Schwab for bringing that package to my attention . . . . .	43	<code>\hyxmp@auto@assign@data</code> : Moved the language-detection and X <sub>Y</sub> T <sub>E</sub> X date-detection code here from the <code>\hyxmp@at@end</code> block . . . . .	37
		Moved title and author autodetection to the <code>\AtEndPreamble</code> . . . . .	37
		Use LuaT <sub>E</sub> X mechanisms, when available, to automatically compute the page count . . . . .	37
		<code>\hyxmp@detect@langs</code> : Set the language(s) immediately instead of deferring them to <code>\hyxmp@set@dc@lang</code> . . . . .	43

Store the main language in <code>\@pdflang</code> . Thanks to Javier Bezos for his help with the <code>hyperxmp</code> code and for modifying <code>babel</code> for <code>hyperxmp</code> 's benefit . . . . .	43	Thanks to Robin Schwab for pointing me to the Journal Article Versions recommendation [1] . . . . .	1
<code>\hyxmp@jav@extensions</code> : Added this macro . . . . .	86	Move most of the <code>\AtEndPreamble</code> code to <code>\hyxmp@at@end</code> . . . . .	35
<code>\hyxmp@jav@schema</code> : Added this macro . . . . .	77	General: Don't inadvertently replace underscores in filenames when writing font-related metadata . . . . .	78
<code>\hyxmp@mm@extensions</code> : Corrected the type of <code>xmpMM:RenditionClass</code> . Thanks to Thorsten Wißmann for the bug report and patch . .	81	Make <code>write_xmp_font_list</code> robust to fonts loaded using <code>HarfBuzz</code> . Thanks to John Lienhard for the bug report . .	78
<code>\hyxmp@query@self</code> : Added this macro . . . . .	40	Make conditional the loading of the <code>ifdraft</code> package. Thanks to Tobias Pape for reporting the incompatibility between <code>hyperxmp</code> and <code>ifdraft</code> . . . . .	1
<code>\hyxmp@rdf@dc</code> : List x-default alternatives before language-specific alternatives, as dictated by the XMP specification [5] . . . . .	67	<code>\hyxmp@aep@toks</code> : As requested by Moritz Heckscher, define <code>\hyxmp@DocumentID</code> and <code>\hyxmp@InstanceID</code> at the end of the preamble instead of at the end of the document . . . .	72
Rewrite the core part of this macro to divide it into four, cleanly defined cases . . . . .	67	General: Do not automatically compute the PDF file size under <code>pdfL<sup>A</sup>T<sub>E</sub>X</code> because this confuses <code>latexmk</code> . Thanks to John Collins, Nelson Posse Lago, Derek Dreyer, and the other contributors to <code>acmart</code> issue #413, "Latexmk goes into an infinite loop even on sample files from ACM" . . . . .	38
<code>\hyxmp@set@koma@phones</code> : Support hyperlinks and other markup in <code>frommobilephone</code> and <code>fromphone</code> , as requested by Robin Schwab . . . . .	35	General: Distribute an <code>add_byteCount</code> script and document some sample <code>latexmk</code> configuration code that invokes it. Thanks to John Collins for providing both of those . . . .	19
<code>\hyxmp@xmptpg@schema</code> : Added this macro . . . . .	77	Take <code>--output-directory</code> into consideration when querying the output file size. Thanks to John Collins for pointing out that the user can change the output directory . . . . .	38
General: Automatically assign <code>pdfnumpages</code> and <code>pdfbytes</code> under <code>pdfL<sup>A</sup>T<sub>E</sub>X</code> and <code>LuaL<sup>A</sup>T<sub>E</sub>X</code> .	1		
Correctly handle source files with spaces in their name. Thanks to Peter Dyballa for the bug report . . . . .	20		
Defer <code>\AtEndPreamble</code> execution until the end of the document. This enables <code>hyperxmp</code> itself to be loaded from <code>\AtEndPreamble</code> , as is done by <code>doclicense</code> v2.2.0. Thanks to Tommaso Pecorella for the bug report and help testing . . . . .	1		
Introduced the <code>pdfpubstatus</code> package option, which enables an author to specify the document's publication status.			

# 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.

<b>Symbols</b>	<b>\@pdfauthortitle</b> ..	<u>161</u> , 320, 1427,
<b>\#</b> ..... 1238, 1525	<u>68</u> , 304, 1494, 1495	1433, 1586, 1673
<b>\&amp;</b> ..... 914, 946, 1524	<b>\@pdfbookedition</b> ..	<b>\@pdfidentifier</b> ...
<b>\@acmBooktitle</b> .... 603	<u>165</u> , 305, 1580, 1670	..... <u>179</u> , 321,
<b>\@acmConference</b> ... 604	<b>\@pdfbytes</b> ..... <u>155</u> ,	1426, 1430, 1437
<b>\@acmDOI</b> .. 574, 575, 577	306, 524, 1589, 1671	<b>\@pdfisbn</b> ..... <u>163</u> ,
<b>\@acmISBN</b> . 584, 585, 588	<b>\@pdfcaptionwriter</b> .	322, 590, 1427,
<b>\@acmNumber</b> ..... 618	<u>70</u> , 307, 1494, 1496	1435, 1584, 1674
<b>\@acmVolume</b> ..... 615	<b>\@pdfcontactaddress</b>	<b>\@pdfissn</b> .....
<b>\@author</b> ..... 391, <u>530</u>	..... <u>185</u> ,	<u>159</u> , 323, 1427,
<b>\@baseurl</b> ... 302, 1491	308, 539, 1557, 1658	1434, 1585, 1675
<b>\@elt</b> <u>676</u> , <u>1373</u> , 1536, <u>1541</u>	<b>\@pdfcontactcity</b> <u>193</u> ,	<b>\@pdfissuenum</b> .. <u>171</u> ,
<b>\@elt@first</b> ..... <u>1534</u>	309, 545, 1558, 1659	324, 617, 1582, 1676
<b>\@elt@rest</b> .. 1536, <u>1538</u>	<b>\@pdfcontactcountry</b>	<b>\@pdfkeywords</b> .....
<b>\@if@def@and@nonempty</b>	..... <u>199</u> ,	.... 224, <u>263</u> , 325
..... <u>29</u> , 405,	310, 557, 1561, 1662	<b>\@pdflang</b> .....
406, 412, 441,	<b>\@pdfcontactemail</b> ..	. 326, 442, 447,
516, 573, 583, 1316	..... <u>203</u> , 311,	450, 626, <u>628</u> , 643
<b>\@ifclassloaded</b> ... 621	453, 533, 1564, 1664	<b>\@pdflicenseurl</b> ...
<b>\@ifmtarg</b> .... 27, 1079	<b>\@pdfcontactphone</b> ..	<u>66</u> , 327, 1442, 1456
<b>\@ifmtargexp</b> .....	..... <u>201</u> ,	<b>\@pdfmetadatettime</b> ..
<u>27</u> , 31, 352, 369,	312, 457, 1563, 1663	<u>51</u> , 328, 1485, 1488
420, 471, 815,	<b>\@pdfcontactpostcode</b>	<b>\@pdfmetalang</b> <u>72</u> , 446,
1305, 1363, 1413,	..... <u>197</u> ,	448, 450, 1074,
1430, 1462, 1463,	313, 563, 1560, 1661	1321, 1337, 1350
1473, 1479, 1485	<b>\@pdfcontactregion</b> .	<b>\@pdfmoddate</b> .....
<b>\@ifnextchar</b> .. 10, 1074	..... <u>195</u> ,	.. 329, 1479, 1483
<b>\@ifnotmtarg</b> .....	314, 551, 1559, 1660	<b>\@pdfnumpages</b> .. <u>157</u> ,
. 28, 75, 1068, 1077	<b>\@pdfcontacturl</b> <u>205</u> ,	330, 519, 522,
<b>\@ifnotmtargexp</b> ...	315, 460, 1565, 1665	1590, 1599, 1677
..... <u>27</u> , 368,	<b>\@pdfcopyright</b> ....	<b>\@pdfpagerange</b> ....
432, 1050, 1091,	..... <u>62</u> , 316,	<u>173</u> , 331, 1583, 1678
1389, 1402, 1529	1411, 1446, 1452	<b>\@pdfproducer</b> . <u>1254</u> ,
<b>\@journalName</b> ..... 602	<b>\@pdfcreationdate</b> ..	<u>1268</u> , <u>1275</u> , 1277
<b>\@latex@warning@no@line</b>	..... 317,	<b>\@pdfpublication</b> <u>151</u> ,
..... 384	471, 472, 1473, 1477	332, 601, 1578, 1679
<b>\@pdfaformance</b> ..	<b>\@pdfcreator</b> ..... 1490	<b>\@pdfpublisher</b> ....
.... <u>84</u> , 359, 1501	<b>\@pdfdatetime</b> .....	<u>167</u> , 466, 598, 1412
<b>\@pdfapart</b> ..... <u>79</u> ,	<u>40</u> , 318, 1413, 1416	<b>\@pdfpubstatus</b> ....
352, 358, 364, 1500	<b>\@pdfdoi</b> ..... <u>175</u> ,	.. <u>183</u> , 1593, 1688
<b>\@pdfauthor</b> 218, <u>242</u> ,	319, 576, 1426,	<b>\@pdfpubtype</b> ... <u>153</u> ,
303, 389, 1203, 1211	1432, 1587, 1672	333, 611, 1579, 1680
	<b>\@pdffeissn</b> .....	<b>\@pdfrendition</b> <u>143</u> , 1470

<code>\@pdfsource</code> . . . . .	<code>atenddvi</code> (package) 20, 105	<code>dc:subject</code> . . . . . 3, 71
. . . <a href="#">125</a> , 1421, 1423	<code>\AtEndPreamble</code> . . . .	<code>dc:title</code> . 3, 13, 60, 70, 102
<code>\@pdfsubject</code> . 334, 1410	. . . 288, 380, 1461	<code>dc:type</code> . . . . . 3, 71
<code>\@pdfsubtitle</code> . . <a href="#">181</a> ,	<code>Author</code> . . . 12, 13, 29, 104	<code>DCMI</code> . . . . . 9
335, 463, 1577, 1681		<code>\define@key</code> 41, 52, 63,
<code>\@pdftitle</code> . . . . .	<b>B</b>	65, 67, 69, 71, 73,
. 336, 369, 385,	<code>babel</code> (package) . . . .	80, 85, 89, 108,
1203, 1211, 1409	. . . . 16, 19, 31,	126, 128, 130,
<code>\@pdftrapped</code> . . . . 1299	36, 43, 44, 105, 106	132, 150, 152,
<code>\@pdftype</code> . . . . <a href="#">64</a> , 1418	<code>\BabelEnsureInfo</code> . . 628	154, 156, 158,
<code>\@pdfuapart</code> . . . . <a href="#">88</a> ,	<code>Bag</code> . . . . . 101	160, 162, 164,
337, 366, 1505, 1983	<code>baseurl</code> (option) . . . .	166, 168, 170,
<code>\@pdfurl</code> . . . . .	5, 7, 16, 21, 27, 73	172, 174, 176,
<a href="#">177</a> , 338, 1588, 1682	<code>\bbl@main@language</code> . 637	178, 180, 182,
<code>\@pdfversionid</code> <a href="#">131</a> , 1469	<code>BOM</code> . . . . . 87, 100	184, 186, 194,
<code>\@pdfvolumenum</code> . <a href="#">169</a> ,		196, 198, 200,
339, 614, 1581, 1683	<b>C</b>	202, 204, 206,
<code>\@pdfxstandard</code> . . . .	<code>CiAdrCity</code> . . . . . 2	230, 242, 263, 1106
. <a href="#">106</a> , 340, 365,	<code>CiAdrCtry</code> . . . . . 2	<code>\do</code> . . . 1108, 1115, 1342
1513, 1516, 1518	<code>CiAdrExtadr</code> . . . . . 2	<code>doclicense</code> (package) . 106
<code>\@pdfxversion</code> . . . . 1987	<code>CiAdrPcode</code> . . . . . 2	<code>DOI</code> . . . . . 2, 7, 27, 42
<code>\@publishers</code> . . . . . 467	<code>CiAdrRegion</code> . . . . . 2	<code>DOS</code> . . . . . 38, 39
<code>\@subtitle</code> . . . . . 464	<code>CiEmailWork</code> . . . . . 2	<code>draft</code> (option) . . . . . 9
<code>\@tempswafalse</code> 1305, 1363	<code>CiTelWork</code> . . . . . 3	Dublin Core schema .
<code>\@tempswattrue</code> . 1305,	<code>\city</code> . . . . . <a href="#">543</a>	. . . . . 2, 64, 67–71
1307, 1363, 1365	<code>CiUrlWork</code> . . . . . 3	<code>DVI</code> . . . . . 38
<code>\@title</code> . . . . . 387	classes	<code>dvipdf</code> (option) . . . . . 91
<code>\^</code> 664, 668, 851, 1231,	<code>ACM</code> . . . . . 16, 35	<code>dvipdfm</code> . . . . . 92
1232, 2012–2014	<code>acmart</code> . . . . .	<code>dvips</code> (option) . . . . . 91
<code>\_</code> . . . . 1230, 1232, 1606	. 16, 37, 40, 42, 43	<code>dvips</code> . 11, 12, 54, 55, 101
<code>\~</code> . . . . . 673, 984, 985	<code>Koma</code> . . . 16, 35, 105	<code>dvipsone</code> (option) . . . . 91
	<code>scrlltr2</code> . . . . . 16	<code>dviwindo</code> (option) . . . . 91
	<code>\clearpage</code> . . . . . 518	
<code>\_</code> . . . . . 950, 985, 1230	<code>\country</code> . . . . . <a href="#">555</a>	<b>E</b>
	<code>CreationDate</code> . . . . 37, 102	<code>ε-TeX</code> . . . . . 66
<b>A</b>		<code>\EdefEscapeHex</code> 873, 886
<code>ACM</code> . . . . . 16, 35, 42	<b>D</b>	<code>\EdefUnescapeHex</code> . . 890
<code>acmart</code> (class) . . . . .	<code>Date</code> . . . . . 50	<code>\EdefUnescapeString</code> 860
. 16, 37, 40, 42, 43	<code>\day</code> . . . . . 793, 794, 796	<code>\email</code> . . . . . <a href="#">531</a>
<code>acmart</code> (package) . . . . 106	<code>dc:creator</code> . 2, 13, 70, 102	<code>\empty</code> . . . . . 1807
<code>add_byteCount</code> 18, 19, 106	<code>dc:date</code> . . . . . 2, 71	<code>\equal</code> . . . . . 102
<code>\addresses</code> . . . . . 571	<code>dc:description</code> . . . . .	<code>etoolbox</code> (package) . . . 21
Adobe PDF schema 64–67	. 3, 13, 60, 70, 102	<code>ETX</code> . . . . . 45, 52
<code>\affiliation</code> . . . . . <a href="#">567</a>	<code>dc:format</code> . . . . . 2	
<code>ALPSP</code> . . . . . 8, 77	<code>dc:identifier</code> . . . . 3, 70, 71	<b>F</b>
<code>\and</code> . . . . . <a href="#">242</a>	<code>dc:language</code> . . . . 3, 16,	<code>\filemoddate</code> . . . . . 828
<code>ASCII</code> . . . . . 22, 53	31, 71, 100, 101, 105	
<code>\AtBeginDocument</code> . 1276	<code>dc:publisher</code> . . . . . 3	<b>G</b>
<code>\AtEndDocument</code> . . . . 7	<code>dc:rights</code> . . . 2, 17, 60, 70	<code>\getlocaleproperty</code> . 630
<code>\AtEndDvi</code> . . . . . 5	<code>dc:source</code> . . . . 2, 71, 100	<code>Ghostscript</code> . . . . . 12

gitver (package) . . . . .	7	\hyxmp@add@simple@var . . . . .	1058, 1297, 1298, 1301	\hyxmp@begin@ext@decl . . . . .	1703, 1742, 1753, 1779, 1795, 1809, 1825, 1837, 1895, 1965
<b>H</b>					
\hbox . . . . .	571	\hyxmp@add@to+xml . . . . .	1052, 1054, 1062, 1080, 1084, 1092, 1096, 1098, 1217, 1243, 1312, 1331, 1338, 1344, 1351, 1356, 1368, 1376, 1382, 1391, 1531, 1535, 1539, 1545, 1552, 1567, 1692, 1698, 1704, 1714, 1721, 1725, 1733, 1846, 1885, 2020, 2065	\hyxmp@begin@extension@decls . . . . .	1691, 1977
\Hy@driver 2076, 2080, 2084, 2088, 2093				\hyxmp@big@prime . . . . .	1125, 1128, 1138, 1148
\Hy@unicodefalse . . . . .	43, 54, 404			\hyxmp@big@prime@ii . . . . .	1125, 1147
hyperref (package) . . . . .	1, 4–6, 9, 10, 13, 16, 19, 21–23, 29–31, 33, 34, 43, 44, 65–67, 89–91, 101–105			\hyxmp@bom . . . . .	2005, 2020
\hypersetup 280, 433, 1329				\hyxmp@ccct . . . . .	1597, 1601
hyperxmp (package) . . . . .	1, 2, 4–10, 13–24, 28, 29, 31, 34–38, 43–46, 52, 61, 65, 67, 79, 90, 93, 94, 101–103, 105, 106	\hyxmp@address@val . . . . .	531, 537, 543, 549, 555, 561	\hyxmp@check@iptc@data . . . . .	1656, 2048
\hyxmp@@is@unicode . . . . .	894	\hyxmp@aep@toks . . . . .	15, 285, 378, 1459, 2206	\hyxmp@check@jav@data . . . . .	1686, 2050
\hyxmp@acm@isbn . . . . .	583	\hyxmp@alt@description . . . . .	1102, 1112	\hyxmp@check@prism@data . . . . .	1668, 2049
\hyxmp@acm@publisher . . . . .	597	\hyxmp@alt@rights . . . . .	1102, 1113	\hyxmp@check@standards . . . . .	350, 398
\hyxmp@acm@pubtype . . . . .	606	\hyxmp@alt@title . . . . .	1102, 1111	\hyxmp@check@std . . . . .	101, 113–121
\hyxmp@add@simple . . . . .	1049, 1299, 1408, 1423, 1437, 1454, 1456, 1467–1470, 1474, 1476, 1480, 1482, 1486, 1488, 1490, 1491, 1495, 1496, 1500, 1501, 1505, 1512, 1513, 1516, 1518, 1558–1561, 1575, 1579, 1581–1590, 1593, 1599	\hyxmp@and . . . . .	242	\hyxmp@comma . . . . .	187, 243, 264, 663
\hyxmp@add@simple@lang . . . . .	1067, 1577, 1578, 1580	\hyxmp@append@hex . . . . .	1151, 1170–1172, 1176	\hyxmp@commas@to@list . . . . .	647, 683, 1374, 1543
\hyxmp@add@simple@lang@i . . . . .	1070, 1073	\hyxmp@append@hex@iii . . . . .	1169, 1175, 1185, 1196	\hyxmp@commas@to@list@i . . . . .	649, 651
\hyxmp@add@simple@lang@ii . . . . .	1074, 1076	\hyxmp@append@hex@iv . . . . .	1174, 1180, 1181, 1183, 1198–1200	\hyxmp@concat@metadata . . . . .	285, 299
\hyxmp@add@simple@pfx . . . . .	1090, 1403	\hyxmp@as@pdf@date . . . . .	720	\hyxmp@cond@dc@identifier . . . . .	1400, 1432–1435
		\hyxmp@as@xmp@date . . . . .	46, 57, 692, 830, 1477, 1483	\hyxmp@construct@packet . . . . .	2018, 2074
		\hyxmp@assign@major@minor . . . . .	1281, 1300	\hyxmp@count@non@spaces . . . . .	2174, 2185
		\hyxmp@at@end . . . . .	3, 396	\hyxmp@count@spaces . . . . .	2173, 2176
		\hyxmp@auto@assign@data . . . . .	397, 440	\hyxmp@crap@convert . . . . .	976, 1010
		\hyxmp@banner@to@producer . . . . .	1257, 1260, 1268	\hyxmp@crap@result . . . . .	966, 1002
				\hyxmp@crap@test . . . . .	973, 998
				\hyxmp@create@uuid . . . . .	1178, 1206, 1215
				\hyxmp@cur@lang . . . . .	1108, 1116

\hyxmp@dc@lang . 442,	1804, 1821,	\hyxmp@list ... 1374,
622, 628, 644, 1425	1833, 1962, 1974	1380, 1543, 1544
\hyxmp@dc@schema ..	\hyxmp@end@extension@decls	\hyxmp@list@to@lines
..... 1407, 2054	..... 1697, 2003	..... 1528,
\hyxmp@declare@extensions	\hyxmp@extra@indent	1557, 1563–1565
..... 1976, 2051	..... 1048,	\hyxmp@list@to@xml .
\hyxmp@declare@field	1052, 1063,	..... 1362,
..... 1732,	1092, 1532, 1556	1419, 1420, 1425
1861, 1864, 1867,	\hyxmp@first@char .. 690	\hyxmp@major@minor 1281
1870, 1873,	\hyxmp@first@char@i	\hyxmp@mm@extensions
1876, 1879, 1882	.... 690, 693, 721	..... 1752, 1979
\hyxmp@declare@property	\hyxmp@gobbletwo 760, 773	\hyxmp@mm@schema ..
... 1720, 1746,	\hyxmp@hash ... 1237,	..... 1466, 2060
1757, 1762, 1767,	2024, 2032,	\hyxmp@modulo@a ...
1771, 1783, 1788,	2040, 2043–2046	... 1119, 1138,
1799, 1813, 1817,	\hyxmp@Hyp@pdfauthor	1148, 1154, 1189
1829, 1841, 1899,	..... 236	\hyxmp@multi@langsfalse
1903, 1907, 1911,	\hyxmp@Hyp@pdfkeywords	..... 1303, 1319
1915, 1919, 1923,	..... 257	\hyxmp@multi@langstrue
1927, 1931, 1935,	\hyxmp@hypersetup .. 280	..... 1303, 1317
1939, 1943, 1948,	\hyxmp@InstanceID ..	\hyxmp@new@xml 1233, 1234
1953, 1958, 1969	..... 129,	\hyxmp@no@bad@parts
\hyxmp@def@DocumentID	1208, 1463, 1468	..... 74, 81, 90
..... 1202, 1462	\hyxmp@iprefix 1094, 1095	\hyxmp@no@info@lists
\hyxmp@def@InstanceID	\hyxmp@iptc@data ...	.... 207, 231, 382
..... 1208, 1463	.. 1550, 1656, 1991	\hyxmp@num ..... 1010
\hyxmp@define@pdfproducer	\hyxmp@iptc@extensions	\hyxmp@one@token ..
..... 1254, 1278	..... 1836, 1993	..... 1127,
\hyxmp@detect@langs	\hyxmp@iptc@schema .	1131, 2177,
..... 444, 623	..... 1549, 2061	2178, 2186, 2187
\hyxmp@DocumentID ..	\hyxmp@is@unicode ..	\hyxmp@padding 1241, 2069
..... 127,	.... 862, 879, 894	\hyxmp@parse@acmart
1202, 1462, 1467	\hyxmp@jav@data ...	.... 469, 528, 621
\hyxmp@dq@code . 1, 2207	..... 1686, 1999	\hyxmp@parse@time ..
\hyxmp@driver .... 2073	\hyxmp@jav@extensions	..... 701, 703
\hyxmp@embed@packet	..... 1964, 2001	\hyxmp@parse@tz ...
..... 400, 2073	\hyxmp@jav@schema ..	.... 710, 713, 717
\hyxmp@embed@packet@dvipdfm	..... 1592, 2063	\hyxmp@parse@tz@char
..... 2085, 2155	\hyxmp@jobname ....	..... 705, 707
\hyxmp@embed@packet@luatex	... 12, 13, 125,	\hyxmp@pdf@extensions
..... 2081, 2118	344, 507, 828,	..... 1741, 1978
\hyxmp@embed@packet@pdfmark	1203, 1211, 2094	\hyxmp@pdf@schema ..
..... 2097, 2125	\hyxmp@koma@phones .	..... 1296, 2052
\hyxmp@embed@packet@pdftex	..... 402, 458	\hyxmp@pdf@to@xmp@date
..... 2077, 2105	\hyxmp@LA@accept ..	. 694, 699, 822, 825
\hyxmp@embed@packet@xetex	... 1105, 1111–1113	\hyxmp@pdfa@id@extensions
..... 2089, 2193	\hyxmp@lang@name .. 628	..... 1778, 1981
\hyxmp@end@ext@decl	\hyxmp@lang@tag ... 628	\hyxmp@pdfa@id@schema
..... 1713,	\hyxmp@legal ..... 1441	..... 1498, 2057
1750, 1776, 1792,		



\hyxmp@pdfauthor ..	1188, 1205, 1214	\hyxmp@text .....	858, 936, 966, 1010
... 233, 242, 1419	\hyxmp@rdfsdc .....	\hyxmp@textunderscore	..... 34
\hyxmp@pdfkeywords .	.. 1304, 1409–1411	\hyxmp@timestamp ..	827
... 233, 263, 1420	\hyxmp@redefine@Hyp	\hyxmp@today@pdf	472, 837
\hyxmp@pdfstringdef	.... 235, 277, 282	\hyxmp@today@xmp ..	..... 815, 820,
..... 34,	\hyxmp@remove@this .	838, 1211, 1414,	1474, 1480, 1486
45, 56, 63, 65, 67,	..... 1272, 1275	\hyxmp@today@xmp@define	... 786, 835, 1209
69, 71, 73, 82,	\hyxmp@rights . 1441,	\hyxmp@toxml ..	888, 911
86, 91, 109, 126,	1444, 1448, 1450	\hyxmp@toxml@unicodetex	..... 876, 936
128, 130, 132,	\hyxmp@seed@rng ...	\hyxmp@trimb ..	844, 847
150, 152, 154,	.. 1127, 1204, 1213	\hyxmp@trimc ..	847, 848
156, 158, 160,	\hyxmp@seed@rng@i ..	\hyxmp@trimspaces ..	..... 656, 840
162, 164, 166,	..... 1129, 1131	\hyxmp@try .....	966
168, 170, 172,	\hyxmp@seed@string .	\hyxmp@try@today	814,
174, 176, 178,	..... 1202, 1208	821, 824, 827, 834	\hyxmp@unicodetexfalse
180, 182, 184,	\hyxmp@set@jobname 9, 14	..... 850	\hyxmp@unicodetextrue
189, 194, 196,	\hyxmp@set@jobname@dbl	..... 850	\hyxmp@uscore .. 36, 667
198, 200, 202,	..... 10, 12	\hyxmp@use@first@valid	. 385, 389, 419,
204, 206, 407,	\hyxmp@set@jobname@plain	453, 457, 460,	463, 466, 524,
409, 413, 1094, 1107	..... 10, 13	533, 539, 545,	551, 557, 563,
\hyxmp@pdfua@id@extensions	\hyxmp@set@koma@phones	576, 590, 598,	601, 611, 614, 617
..... 1794, 1985	..... 402, 456	\hyxmp@use@first@valid@i	..... 421, 425
\hyxmp@pdfua@id@schema	\hyxmp@set@pdfx@major	\hyxmp@value	1107, 1311
..... 1504, 2058	..... 93, 123	\hyxmp@warn@if@no@metadata	..... 299, 399
\hyxmp@pdfx@id@extensions	\hyxmp@set@pdfx@major@i	\hyxmp@x@default ..	.... 448, 1253,
..... 1806, 1989	..... 93, 94	1321, 1332, 1339	\hyxmp@xetex@crap ..
\hyxmp@pdfx@id@schema	\hyxmp@set@pdfx@major@ii	..... 867, 966	\hyxmp+xml 1053, 1055,
..... 1507, 2059	..... 95, 98	1093, 1099, 1234,	1241, 1724, 2018,
\hyxmp@pdfx@major ..	\hyxmp@set@rand@enum	2114, 2122, 2145,	2156, 2163, 2194
.. 98, 107, 123,	.. 1144, 1152, 1187		
1508, 1807, 1823	\hyxmp@singleton@dc		
\hyxmp@photoshop@data	... 1388, 1412,		
..... 1493	1414, 1416, 1418		
\hyxmp@photoshop@schema	\hyxmp@skiptorelax .		
..... 1493, 2055	..... 1003, 1009		
\hyxmp@prev@pdf@size	\hyxmp@skipzeros ..		
..... 505, 525	..... 961		
\hyxmp@prism@data ..	\hyxmp@SpaceOther ..		
.. 1573, 1668, 1995	..... 970, 983		
\hyxmp@prism@extensions	\hyxmp@standards ..		
..... 1894, 1997	..... 363		
\hyxmp@prism@schema	\hyxmp@string@len ..		
..... 1572, 2062	..... 2156, 2171		
\hyxmp@ProcessKeyvalOptions	\hyxmp@strip@isbn@date		
..... 275	..... 583		
\hyxmp@prot@us ...	\hyxmp@sublist ....		
1605	.. 652, 653, 656, 657		
\hyxmp@query@self ..	\hyxmp@suppress@pdf@info		
..... 476, 514	..... 208		
\hyxmp@rand@num ...	\hyxmp@temp@list ..		
... 1144, 1153,	..... 676		
	\hyxmp@temp@str ...		
	..... 676		

<code>\hyxmp@xmlified</code> . . .	<code>\ifHy@pdfa</code> . . . . .	Keywords 12, 13, 29, 67, 104
. . . . 858, 1054,	351, 1409, 1410,	Koma (class) 16, 35, 105
1063, 1070, 1081,	1419, 1499, 1980	<code>\KV@Hyp@pdfauthor</code> . . 242
1085, 1096, 1098,	<code>\ifhyxmp@multi@langs</code>	<code>\KV@Hyp@pdfkeywords</code> 263
1311, 1340, 1345,	. . 1303, 1322, 1336	<code>kvoptions</code> (package) 21, 31
1352, 1374, 1394,	<code>\ifhyxmp@unicodetex</code>	
1401, 1431, 1543	850, 861, 1220, 2006	<b>L</b>
<code>\hyxmp@xmlify</code> . . . . .	<code>\ifLuaTeX</code> . . 24, 478,	<code>latexmk</code> . . . . 19, 38, 106
. . . . 858, 1051,	482, 515, 1256,	<code>LF</code> . . . . . 75
1061, 1069, 1078,	1596, 1601, 1609	<code>\LocaleForEach</code> . . . . 629
1095, 1097, 1310,	<code>ifluatex</code> (package) . . . 90	<code>Lua</code> . . . . . 78
1337, 1343, 1350,	<code>\ifluatex</code> . . . 2107, 2111	<code>luacode</code> (package) . . . . 21
1373, 1390, 1542	<code>ifmtarg</code> (package) 21, 103	<code>\luadirect</code> 506, 519, 1597
<code>\hyxmp@xmp@basic@schema</code>	<code>\ifPDFTeX</code> . . . . . 1259	<code>Lua<sup>A</sup>TeX</code> . . 10, 11, 14,
. . . . . 1472, 2056	<code>\IfSubStr</code> . . . . .	15, 18, 19, 38, 50,
<code>\hyxmp@xmp@to@pdf@date</code>	. 574, 575, 584, 585	51, 66, 67, 102, 106
. . . . 724, 727, 838	<code>iftex</code> (package) . . 21, 104	<code>LuaTeX</code> . . . . . 21,
<code>\hyxmp@xmp@to@pdf@date@i</code>	<code>ifthen</code> (package) . . . . . 21	38, 40, 52, 55, 77,
. . . . . 728, 730	<code>\ifthenelse</code> . . . . . 102	90, 91, 93, 101–105
<code>\hyxmp@xmp@to@pdf@date@ii</code>	<code>\ifXeTeX</code> . . . . 866, 1262	<code>luatex85</code> (package) . . . 104
. . . . . 733, 736	<code>Info</code> 12–14, 29, 34, 37, 65	<code>\luatexbanner</code> . . . . 1257
<code>\hyxmp@xmp@to@pdf@date@iii</code>	<code>intcalc</code> (package) . . . . 21	
. . . . . 739, 742	<code>\intcalcDiv</code> . . . . .	<b>M</b>
<code>\hyxmp@xmp@to@pdf@date@iv</code>	. . 1015, 1022, 1029	<code>\makeatletter</code> . . . . 1272
. . . . . 745, 748	<code>\intcalcMod</code> . . . . .	<code>memoir</code> (package) . . . 101
<code>\hyxmp@xmp@to@pdf@date@v</code>	. . 1017, 1024, 1031	<code>Metadata</code> . . . . 12, 89, 93
. . . . . 751, 754	<code>IPTC</code> . . . . . 14,	<code>\month</code> . . . . 788, 789, 791
<code>\hyxmp@xmp@to@pdf@date@vi</code>	28, 64, 75, 76, 79,	
. . . . . 757, 761	83, 87, 101, 112, 115	<b>N</b>
<code>\hyxmp@xmp@to@pdf@date@vii</code>	<code>IPTC Photo Metadata</code>	<code>NAK</code> . . . . . 22, 45, 52
. . . . 764, 767, 777	schema . 64, 75–76	<code>nativepdf</code> (option) . . . . 91
<code>\hyxmp@xmp@to@pdf@date@viii</code>	<code>lptc4xmpCore:Contact-</code>	<code>\newcatcodetable</code> . 1602
. . . . . 780, 783	Info . . . . . 76, 83	<code>\newif</code> . . . . . 850, 1303
<code>\hyxmp@xmpRights@schema</code>	<code>lptc4xmpCore:Creator-</code>	<code>\newtoks</code> . . . . . 15
. . . . . 1440, 2053	ContactInfo . 2,	<code>\next</code> . . . . 44, 55, 103,
<code>\hyxmp@xmptpg@schema</code>	3, 75, 76, 101	110, 133, 148,
. . . . . 1595, 2064	<code>ISBN</code> . . . . . 2, 7, 26, 42	208, 426, 428,
<code>\hyxmp@zero</code> . . . . .	<code>ISO</code> . . . . . 6, 7, 16, 23, 60	434, 438, 651,
. . . 1019, 1026,	<code>ISSN</code> . . . . . 2, 7, 26	829, 832, 1131, 1987
1033, 1039, 1044		<code>ngerman</code> (package) 20, 100
<b>I</b>	<b>J</b>	<code>NISO</code> . . . . . 8, 77
<code>IETF</code> . . . . . 6	<code>JAV</code> . . . . . 79, 86, 87	<code>\number</code> . . . . . 1013,
<code>\if@ACM@journal</code> . . . 606	<code>jav:journal_article_ver-</code>	1015, 1017, 1022,
<code>\if@tempswa</code> . 1309, 1367	sion . . . . . 3	1024, 1029, 1031
<code>ifdraft</code> (package) 19, 25, 106	<code>\jobname</code> . . . . . 14	<code>\numexpr</code> . . . . . 2123
<code>\ifdraft</code> . . . . 133, 143	<code>Journal Article Versions</code>	
<code>\iffalse</code> . . . 1304, 1362	schema . . . 64, 77	<b>O</b>
	<b>K</b>	<code>options</code>
	<code>keeppdfinfo</code> (option) 14, 29	<code>baseurl</code> . . . . .
		5, 7, 16, 21, 27, 73



draft . . . . . 9  
 dvipdf . . . . . 91  
 dvips . . . . . 91  
 dvipsone . . . . . 91  
 dviwindo . . . . . 91  
 keeppdfinfo . . . 14, 29  
 nativepdf . . . . . 91  
 pdfa . . 9, 23, 33, 102  
 pdfaconformance .  
     . . . . . 5, 9, 74  
 pdfapart . . 5, 9, 23, 74  
 pdfauthor . . 5, 6,  
     13, 15, 16, 21, 29,  
     30, 34, 71, 102, 105  
 pdfauthoritle . 5, 6, 16  
 pdfbookedition . . 5, 8  
 pdfbytes . 5, 10, 40, 106  
 pdfcaptionwriter . 5, 6  
 pdfcontactaddress .  
     . . . . . 5, 6, 14  
 pdfcontactcity . . 5, 6  
 pdfcontactcountry . 5, 6  
 pdfcontactemail . . 5, 6  
 pdfcontactphone . . 5, 6  
 pdfcontactpostcode .  
     . . . . . 5, 6  
 pdfcontactregion . . 5, 6  
 pdfcontacturl . . 5, 6, 16  
 pdfcopyright . . . .  
     . . 5, 6, 70, 72, 101  
 pdfcreationdate . .  
     . . . . . 5, 9, 37, 103  
 pdfdate . 5, 8, 9, 16,  
     22, 63, 71, 102, 104  
 pdfdocumentid . 5, 7, 103  
 pdfdoi . . . . . 5, 7  
 pdfeissn . . . . . 5, 7  
 pdfidentifier . 5, 7, 71, 105  
 pdfinstanceid . 5, 7, 103  
 pdfisbn . . . . . 5, 7  
 pdfissn . . . . . 5, 7  
 pdfissuenumber . . . 5, 8  
 pdfkeywords . . . 5,  
     13, 15, 21, 29, 30, 71  
 pdflang . . 5–7, 16,  
     21, 43, 60, 71, 103  
 pdflicenseurl . . . .  
     . . 5, 6, 16, 72, 101  
 pdfmark . . . . . 91  
 pdfmetadate . . . .  
     . . . . . 5, 9, 22, 103  
 pdfmetalang . 5, 6, 16,  
     60, 68, 69, 100, 103  
 pdfmoddate . . 5, 9, 103  
 pdfnumpages . 5, 10, 106  
 pdfpagerange . . . .  
     . . . . . 5, 8, 17, 18  
 pdfproducer . . 5, 21, 66  
 pdfpublication . . . 5–8  
 pdfpublisher . . . . 5, 8  
 pdfpubstatus . 5, 8, 106  
 pdfpubtype . . . . 5, 8  
 pdfrendition . . 5, 9, 104  
 pdfsource . . 5, 10, 103  
 pdfsubject . 5, 13, 21, 70  
 pdfsubtitle . . . . 5, 6  
 pdftitle . 5, 6, 13, 16,  
     21, 34, 70, 102, 105  
 pdftrapped . 5, 9, 21, 104  
 pdftype . . 6, 9, 71, 102  
 pdfuapart . . 6, 9, 23, 74  
 pdfurl . . . . . 6, 7  
 pdfversionid . . 6, 7, 72  
 pdfvolumenum . . 6, 8  
 pdfxstandard . . . .  
     . . . 6, 9, 24, 25, 74  
 ps2pdf . . . . . 91  
 textures . . . . . 91  
 unicode . . . . . 16, 103  
 vtexpdfmark . . . . . 91

## P

\PackageError . . . . 1323  
 packages  
   acmart . . . . . 106  
   atenddvi . . . . . 20, 105  
   babel . . 16, 19, 31,  
     36, 43, 44, 105, 106  
   doclicense . . . . . 106  
   etoolbox . . . . . 21  
   gitver . . . . . 7  
   hyperref . . . . . 1,  
     4–6, 9, 10, 13,  
     16, 19, 21–23,  
     29–31, 33, 34,  
     43, 44, 65–67,  
     89–91, 101–105  
   hyperxmp . . . . . 1,  
     2, 4–10, 13–24,  
     28, 29, 31, 34–38,  
     43–46, 52, 61, 65,  
     67, 79, 90, 93, 94,  
     101–103, 105, 106  
   ifdraft . . . 19, 25, 106  
   ifluatex . . . . . 90  
   ifmtarg . . . . . 21, 103  
   iftex . . . . . 21, 104  
   ifthen . . . . . 21  
   intcalc . . . . . 21  
   kvoptions . . . . . 21, 31  
   luacode . . . . . 21  
   luatex85 . . . . . 104  
   memoir . . . . . 101  
   ngerman . . . . . 20, 100  
   pdfescape . . . . . 21  
   pdfx . . . . . 4, 5  
   polyglossia . . . . .  
     . . . . 16, 19, 31,  
     35, 36, 43, 44, 105  
   stringenc . . . . . 21  
   texdate . . . . . 17  
   totpages . 17, 18, 38, 40  
   xmpincl . . . . . 4  
 \PackageWarning . . .  
     . . . . . 76, 111, 677  
 \PackageWarningNoLine  
     . . . . . 210,  
     343, 353, 370, 2092  
 \patchcmd . . . . . 216, 222  
 PDF . 1–5, 8, 9, 11–15,  
     17–19, 22, 29,  
     33–35, 37–39, 44,  
     46–48, 51, 53, 54,  
     63–67, 77, 81, 86,  
     88–90, 93, 102,  
     105, 106, 108, 115  
   Author . 12, 13, 29, 104  
   CreationDate . 37, 102  
   Info . . . . . 12–14,  
     29, 34, 37, 65  
   Keywords . . . . .  
     12, 13, 29, 67, 104  
   Metadata . . 12, 89, 93  
   Producer . . . . . 67  
   Subject . . . . . 12, 13  
   Title . . . . . 12, 13  
 PDF/A . . . . . 3, 9, 13,  
     14, 23, 29, 33, 65,

67, 74, 79, 82–87,  
 101, 102, 114, 115  
 PDF/A Identification  
   schema . . . 64, 74  
 PDF/UA 3, 9, 24, 33, 74,  
   82, 87, 104, 114, 115  
 PDF/UA Identification  
   schema . . . 64, 74  
 PDF/X . . . . . 3, 9,  
   24, 25, 33, 74, 82,  
   83, 87, 104, 114, 115  
 PDF/X Identification  
   schema . . . 64, 74  
 pdf:Keywords . . . 3, 13, 67  
 pdf:PDFVersion 3, 67, 105  
 pdf:Producer . . . 3, 65, 67  
 pdf:trapped . . . . . 3  
 \PDF@FinishDoc . . . .  
   . . . . 209, 217, 223  
 pdfa (option) 9, 23, 33, 102  
 pdfa:conformance (op-  
   tion) . . . 5, 9, 74  
 pdfaid:conformance . . . 3  
 pdfaid:part . . . . . 3  
 pdfapart (option) . . . .  
   . . . . 5, 9, 23, 74  
 pdfaType:prefix . . . . 101  
 pdfauthor (option) 5, 6,  
   13, 15, 16, 21, 29,  
   30, 34, 71, 102, 105  
 pdfauthor:part (option)  
   . . . . . 5, 6, 16  
 pdfbookedition (option)  
   . . . . . 5, 8  
 pdfbytes (option) . . . .  
   . . . . 5, 10, 40, 106  
 pdfcaptionwriter (op-  
   tion) . . . . . 5, 6  
 \pdfcatalog . . . . . 2115  
 \pdfcompresslevel . 2109  
 pdfcontactaddress (op-  
   tion) . . . 5, 6, 14  
 pdfcontactcity (option)  
   . . . . . 5, 6  
 pdfcontactcountry (op-  
   tion) . . . . . 5, 6  
 pdfcontactemail (op-  
   tion) . . . . . 5, 6  
 pdfcontactphone (op-  
   tion) . . . . . 5, 6  
 pdfcontactpostcode (op-  
   tion) . . . . . 5, 6  
 pdfcontactregion (op-  
   tion) . . . . . 5, 6  
 pdfcontacturl (option)  
   . . . . . 5, 6, 16  
 pdfcopyright (option) .  
   . . . 5, 6, 70, 72, 101  
 pdfcreationdate (option)  
   . . . . . 5, 9, 37, 103  
 \pdfcreationdate . . 822  
 pdfdate (option) . . . .  
   . . . . 5, 8, 9, 16,  
   22, 63, 71, 102, 104  
 PDFDocEncoding . . . .  
   . . . . . 29, 52, 53  
 pdfdocumentid (option)  
   . . . . . 5, 7, 103  
 pdfdoi (option) . . . . 5, 7  
 pdfeissn (option) . . . 5, 7  
 pdfescape (package) . . 21  
 \pdfextension 2119, 2123  
 \pdffeedback . 825, 2123  
 pdfidentifier (option) .  
   . . . . . 5, 7, 71, 105  
 pdfinstanceid (option)  
   . . . . . 5, 7, 103  
 pdfisbn (option) . . . . 5, 7  
 pdfissn (option) . . . . 5, 7  
 pdfissuenumber (option) 5, 8  
 pdfkeywords (option) 5,  
   13, 15, 21, 29, 30, 71  
 pdflang (option) 5–7, 16,  
   21, 43, 60, 71, 103  
 \pdflastobj . . . . . 2115  
 pdfL<sup>A</sup>T<sub>E</sub>X . . 4, 10, 11,  
   14, 38, 50, 66, 106  
 pdflicenseurl (option) .  
   . . . 5, 6, 16, 72, 101  
 \pdfmajorversion . 1289  
 pdfmark (option) . . . . 91  
 \pdfmark . . . . . 2126,  
   2129, 2133,  
   2143, 2147, 2151  
 pdfmetadate (option) .  
   . . . . 5, 9, 22, 103  
 pdfmetalang (option) .  
   . . . . 5, 6, 16,  
   60, 68, 69, 100, 103  
 \pdfminorversion . 1285  
 pdfmoddate (option) .  
   . . . . . 5, 9, 103  
 pdfnumpages (option)  
   . . . . . 5, 10, 106  
 \pdfobj . . . . . 2111  
 pdfpagerange (option)  
   . . . . . 5, 8, 17, 18  
 pdfproducer (option) .  
   . . . . . 5, 21, 66  
 pdfpublication (option) 5–8  
 pdfpublisher (option) 5, 8  
 pdfpubstatus (option)  
   . . . . . 5, 8, 106  
 pdfpubtype (option) . 5, 8  
 pdfrendition (option) .  
   . . . . . 5, 9, 104  
 pdfsource (option) 5, 10, 103  
 \pdfstringdef . . 37, 522  
 pdfsubject (option) . .  
   . . . . . 5, 13, 21, 70  
 pdfsubtitle (option) . 5, 6  
 pdfT<sub>E</sub>X 54, 90, 93, 103, 104  
 \pdftexbanner . . . . 1260  
 pdftitle (option) . . . .  
   . . . 5, 6, 13, 16,  
   21, 34, 70, 102, 105  
 pdftrapped (option) . .  
   . . . . 5, 9, 21, 104  
 pdftype (option) . . . .  
   . . . . 6, 9, 71, 102  
 pdfuaid:part . . . . . 3  
 pdfuapart (option) . . .  
   . . . . . 6, 9, 23, 74  
 pdfurl (option) . . . . 6, 7  
 \pdfvariable . . . . . 1293  
 pdfversionid (option) .  
   . . . . . 6, 7, 72  
 pdfvolumenum (option)  
   . . . . . 6, 8  
 pdfx (package) . . . . 4, 5  
 pdfxid:GTS\_PDFXVer-  
   sion . . . . . 3  
 pdfxstandard (option)  
   . . . 6, 9, 24, 25, 74  
 Perl . . . . . 18  
 Photoshop schema . .  
   . . . . . 64, 73–74  
 photoshop:AuthorsPosi-  
   tion . . . . . 3, 73

photoshop:Caption-Writer . . . . .	3, 73		xmpRights:WebState-ment . . .	3, 71, 101
PI . . . . .	64		ps2pdf (option) . . . . .	91
polyglossia (package) . . . . .	16, 19, 31, 35, 36, 43, 44, 105			
\postcode . . . . .	<u>561</u>		<b>Q</b>	
PRISM . . . . .	8, 18, 64, 76, 79, 84, 85, 87, 115		\Q . . . . .	840, 849
PRISM Basic Metadata schema . . . . .	18, 64, 76–77		<b>R</b>	
prism:aggregationType . . . . .	3		RDF . . . . .	69
prism:bookEdition . . . . .	2		rdf:Description . . . . .	104
prism:byteCount . . . . .	2, 18		rdf:li . . . . .	2
prism:doi . . . . .	2		rdf:Seq . . . . .	2
prism:elssn . . . . .	2		\ref . . . . .	522
prism:isbn . . . . .	2		\renewcommand . . . . .	276
prism:issn . . . . .	2		\RequirePackage . . . . .	4, 16–23, 25, 138, 381, 480, 2104
prism:number . . . . .	2		<b>S</b>	
prism:pageCount . . . . .	3		\savecatcodetable . . . . .	1603
prism:pageRange . . . . .	3		\scantokens . . . . .	1269, 1272
prism:publicationName . . . . .	3		schemata	
prism:subtitle . . . . .	3		Adobe PDF . . . . .	64–67
prism:url . . . . .	3		Dublin Core . . . . .	2, 64, 67–71
prism:volume . . . . .	4		IPTC Photo Meta-data . . . . .	64, 75–76
\ProcessKeyvalOptions . . . . .	<u>275</u>		Journal Article Versions . . . . .	64, 77
Producer . . . . .	67		PDF/A Identification . . . . .	64, 74
properties, XMP			PDF/UA Identification . . . . .	64, 74
dc:creator . . . . .	2, 13, 70, 102		PDF/X Identification . . . . .	64, 74
dc:date . . . . .	2, 71		Photoshop . . . . .	64, 73–74
dc:description . . . . .	3, 13, 60, 70, 102		PRISM Basic Meta-data . . . . .	18, 64, 76–77
dc:format . . . . .	2		XMP Basic . . . . .	64, 73
dc:identifier . . . . .	3, 70, 71		XMP Media Manage-ment . . . . .	64, 72–73
dc:language . . . . .	3, 16, 31, 71, 100, 101, 105		XMP Paged-Text . . . . .	64, 77–78
dc:publisher . . . . .	3		XMP Rights Manage-ment . . . . .	64, 71–72
dc:rights . . . . .	2, 17, 60, 70		\scr@fromemail@var . . . . .	454
dc:source . . . . .	2, 71, 100		\scr@frommobilephone@var . . . . .	407, 409
dc:subject . . . . .	3, 71		\scr@fromname@var . . . . .	390
dc:title . . . . .	3, 13, 60, 70, 102			
dc:type . . . . .	3, 71			
lptc4xmpCore:Con-tactInfo . . . . .	76, 83			
lptc4xmpCore:Cre-atorContact-Info . . . . .	2, 3, 75, 76, 101			
jav:journal_arti-cle_version . . . . .	3			
pdf:Keywords . . . . .	3, 13, 67			
pdf:PDFVersion . . . . .	3, 67, 105			
pdf:Producer . . . . .	3, 65, 67			
pdf:trapped . . . . .	3			
pdfaid:conformance . . . . .	3			
pdfaid:part . . . . .	3			
pdfaType:prefix . . . . .	101			
pdfuaid:part . . . . .	3			
pdfxid:GTS_PDFXVer-sion . . . . .	3			
photoshop:Author-sPosition . . . . .	3, 73			
photoshop:Caption-Writer . . . . .	3, 73			
prism:aggregation-Type . . . . .	3			
prism:bookEdition . . . . .	2			
prism:byteCount . . . . .	2, 18			
prism:doi . . . . .	2			
prism:elssn . . . . .	2			
prism:isbn . . . . .	2			
prism:issn . . . . .	2			
prism:number . . . . .	2			
prism:pageCount . . . . .	3			
prism:pageRange . . . . .	3			
prism:publication-Name . . . . .	3			
prism:subtitle . . . . .	3			
prism:url . . . . .	3			
prism:volume . . . . .	4			
xmp:BaseURL . . . . .	2			
xmp:CreateDate . . . . .	2, 37, 102			
xmp:CreatorTool . . . . .	3			
xmp:MetadataDate . . . . .	2, 102			
xmp:ModifyDate . . . . .	2, 102			
xmpMM:Documen-tID . . . . .	3, 61, 72, 86			
xmpMM:InstanceID . . . . .	4, 61, 72, 86			
xmpMM:Rendition-Class . . . . .	3, 106			
xmpMM:VersionID . . . . .	4, 72, 104			
xmpRights:Marked . . . . .	2, 71, 101			

<code>\scr@fromphone@var</code> .	URI . . . . . 7	XMP Basic schema . 64, 73
. . . . . 407, 413	URL . . . 2, 3, 6, 7, 16,	XMP Media Management
<code>\scr@fromurl@var</code> . . 461	23, 27, 28, 71–73, 76	schema . 64, 72–73
<code>\scr@subject@var</code> . . 386	UTF-16BE . . . . . 54	XMP Paged-Text
<code>scr\trr2</code> (class) . . . . . 16	UTF-32BE . . . . . 53	schema . 64, 77–78
<code>\SE-&gt;pdfdoc@03</code> . . . . . <u>856</u>	UTF-8 . . . . . 53, 54	XMP Rights Manage-
<code>\SE-&gt;pdfdoc@15</code> . . . . . <u>857</u>	UUID . . . . . 3, 4, 6,	ment schema .
<code>\setbox</code> . . . . . 571	7, 25, 61–63, 72, 101	. . . . . 64, 71–72
<code>\setkeys</code> . . . . . 1117		<code>xmp:BaseURL</code> . . . . . 2
<code>\special</code> . . . . . 2157,	<b>V</b>	<code>xmp:CreateDate</code> 2, 37, 102
2165, 2194, 2200	<code>\vfuzz</code> . . . . . 848	<code>xmp:CreatorTool</code> . . . . . 3
<code>\state</code> . . . . . <u>549</u>	<code>vtexpdfmark</code> (option) . . 91	<code>xmp:MetadataDate</code> . 2, 102
<code>\streetaddress</code> . . . . . <u>537</u>		<code>xmp:ModifyDate</code> . . 2, 102
<code>stringenc</code> (package) . . . 21	<b>X</b>	<code>\xmcomma</code> . . . . . 187,
<code>\StringEncodingConvert</code>	<code>\x</code> . . . . . <u>966</u>	190, <u>242</u> , <u>263</u> , <u>662</u>
. . . . . 863,	<code>x-default</code> . . . . .	<code>xmpincl</code> (package) . . . . 4
869, 880, 883, 978	6, 16, 37, 60, 65,	<code>\XMPLangAlt</code> . <u>1114</u> , 1324
<code>Subject</code> . . . . . 12, 13	68, 69, 100, 103, 106	<code>\xmplinesep</code> . . . . .
	<code>xdvipdfmx</code> . . . 15, 37, 93	. . . <u>1523</u> , <u>1539</u> , <u>1562</u>
<b>T</b>	<code>X<sub>La</sub>TeX</code> . . . . . 11, 15,	<code>xmpMM:DocumentID</code> .
<code>TeX</code> 17, 20, 21, 49–52,	37, 51, 66, 102, 104	. . . . . 3, 61, 72, 86
54, 55, 61, 63, 65,	<code>X<sub>Y</sub>TeX</code> 52, 55, 56, 93,	<code>xmpMM:InstanceID</code> . .
66, 72, 92, 93, 105	100, 101, 104, 105	. . . . . 4, 61, 72, 86
<code>texdate</code> (package) . . . 17	<code>\XeTeXrevision</code> . . . 1263	<code>xmpMM:RenditionClass</code>
<code>Text</code> . . . . . 80, 81	<code>\XeTeXversion</code> . . . . 1263	. . . . . 3, 106
<code>\textunderscore</code> . . .	XML 1, 2, 14, 44, 52–55,	<code>xmpMM:VersionID</code> . . .
. . . . . 35, 36, 38	58, 59, 64, 67–70,	. . . . . 4, 72, 104
<code>textures</code> (option) . . . . 91	75, 76, 80, 88, 103	<code>\xmquote</code> . . . . . 188,
<code>\time</code> . . . . . 798, 806	XMP . . . . . 1, 2, 4–6,	191, <u>242</u> , <u>263</u> , <u>671</u>
<code>Title</code> . . . . . 12, 13	8, 9, 12–19, 21,	<code>xmpRights:Marked</code> 2, 71, 101
<code>totpages</code> (package) . .	22, 24, 29, 34,	<code>xmpRights:WebState-</code>
. . . . . 17, 18, 38, 40	35, 37, 40–48, 50,	ment . . 3, 71, 101
	51, 54, 55, 58–61,	<code>\xmptilde</code> . . . . . <u>672</u>
<b>U</b>	64, 65, 67–69,	<code>\XMPTruncateList</code> . . <u>676</u>
<code>\undefined</code> . . . . . 430	71–73, 77–79,	<code>\xpg@bcp@loaded</code> . . . 644
Unicode 16, 21, 52–56,	81, 86, 90–94,	
70, 75, 87, 93, 100	100–106, 115, 116	<b>Y</b>
<code>unicode</code> (option) . 16, 103	properties . . . . .	<code>\year</code> . . . . . 787
	<i>see</i> properties, XMP	