

The hyperxmp package*

Scott Pakin
scott+hyxmp@pakin.org

February 19, 2023

Abstract

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

1 Introduction

Adobe Systems, Inc. has been promoting XMP [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 \LaTeX document with three authors—Jack Napier, Edward Nigma, and Harvey Dent—named in the \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>
    <rdf:li>Harvey Dent</rdf:li>
  </rdf:Seq>
</dc:creator>
```

*This document corresponds to `hyperxmp` v5.11, dated 2022/10/16.

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 \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
(lptc4xmpCore: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 (lptc4xmpCore:CreatorContactInfo.CiUrlWork)
- UUID for the document (xmpMM:DocumentID)
- UUID for the document instance (xmpMM:InstanceID)
- version identifier for the document (xmpMM:VersionID)
- volume number of parent publication (prism:volume)

More types of metadata may be added in a future release.

<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

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

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

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

2.1 Option descriptions

`pdftitle` The document title is specified as normal for hyperref with `pdftitle`, but see Note 7 on page 15 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 14 for a discussion of the correct syntax. If `pdfauthor` is not specified it will inherit its value from the document's `\author`. `pdfauthor` indicates the primary author's position or title. `pdfcaptionwriter` specifies the name of the person who added the metadata to the document.

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

`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.

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 alternatively specify a particular document identifier using `pdfdocumentid` and (not normally recommended) a particular instance identifier using `pdfinstanceid`. These should be of the form `uuid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`, where “x” is a lowercase hexadecimal number. For example, `uuid:53ab7f19-a48c-5177-8bb2-403ad907f632` is a valid argument to `pdfdocumentid` (or `pdfinstanceid`). See Leach, Mealling, and Salz's UUID specification document for details on how to produce the various forms of UUIDs [12]. A more freeform mechanism than `pdfinstanceid` for versioning documents is available via `pdfversionid`. The version specified by `pdfversionid` can be incremented as 1, 2, 3, ...; identified with a hierarchical numbering scheme (e.g., this document is versioned 5.11 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 `\gitVer` macro from the `gitver` package is an expandable (see Note 8 on page 16) version of the current `Git` hash that can suitably be passed to `pdfversionid`. If not specified, `pdfversionid` defaults to 1.

`pdfisbn` Already-published documents can be identified in a number of ways. `pdfisbn`

<code>pdfissn</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.
<code>baseurl</code>	
<code>pdfidentifier</code>	<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. ¹ <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.
<code>pdfpublication</code>	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 named using <code>pdfpublisher</code> .
<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 16 for advice on how to assign <code>pdfpagerange</code> semi-automatically.
<code>pdfpagerange</code>	
<code>pdfpubstatus</code>	A 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>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 L ^A T _E X <code>\date</code> command, and, like <code>\date</code> , defaults to the date

¹See, 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

the document was built. It must be specified in either XMP format [5] or PDF format [4]. XMP dates are written in the form `YYYY-MM-DDThh:mm:ss+TT:tt`.² 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 `2014-09-23T14:15:09-06:00`. This can be truncated (with loss of information) to `2014-09-23T14:15:09`, `2014-09-23T14:15`, `2014-09-23`, `2014-09`, or `2014` but no other subsets. PDF dates are written in the form `D:YYYYMMDDhhmmss+TT’tt’`. The same date in the preceding example would be written as `D:20140923141509-06’00’` in PDF format.

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

`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^AT_EX is commonly used to typeset.

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`.

`hyperxmp` honors `hyperref`’s `pdftrapped` option. A document can indicate whether

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

it employs **color trapping** by specifying `pdftrapped=True` or `pdftrapped=False`. (`pdftrapped=Unknown` is also allowed.)

`pdfapart` and `pdfaconformance`, are used in conjunction with `hyperref`'s `pdfa` option to claim a particular PDF/A standard by which the document abides. They default to `pdfapart=1` and `pdfaconformance=B`, indicating the PDF/A-1b standard. These can be changed (with caution) to assert that the document abides by a different standard (e.g., PDF/A-2u). A document that conforms to the PDF/UA standard can use `pdfuapart` to indicate the PDF/UA conformance level. For example, `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-1a:2003
- PDF/X-3:2002
- PDF/X-3:2003
- PDF/X-4
- PDF/X-4p
- PDF/X-5g
- PDF/X-5n
- 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^AT_EX source file. It defaults to `\jobname.tex` but can be replaced by any other string. If `pdfsource` is given an empty argument, no document source will be specified at all.

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 `pdfLATEX` or `LuaLATEX`.

`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 `pdfLATEX` and `LuaLATEX`, 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^AT_EX 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},
  pdfidentfier={info:lccn/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:

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

The L^AT_EX + Dvips + Ghostscript path stores the XMP packet in a compressed stream, which implies that a PDF reader is needed to access it. Ideally, XMP metadata should be stored uncompressed so it can be extracted as ordinary text. Unfortunately, as of 2022-10-07, Ghostscript has no plans to support uncompressed metadata streams ([Ghostscript bug report #705962](#)). It is possible to leave *all* streams uncompressed by passing `-dCompressStreams=false` to Ghostscript (e.g., via the `ps2pdf` wrapper script), but this leads to larger file sizes.

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.

2.3 Usage notes

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

```
<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>
```

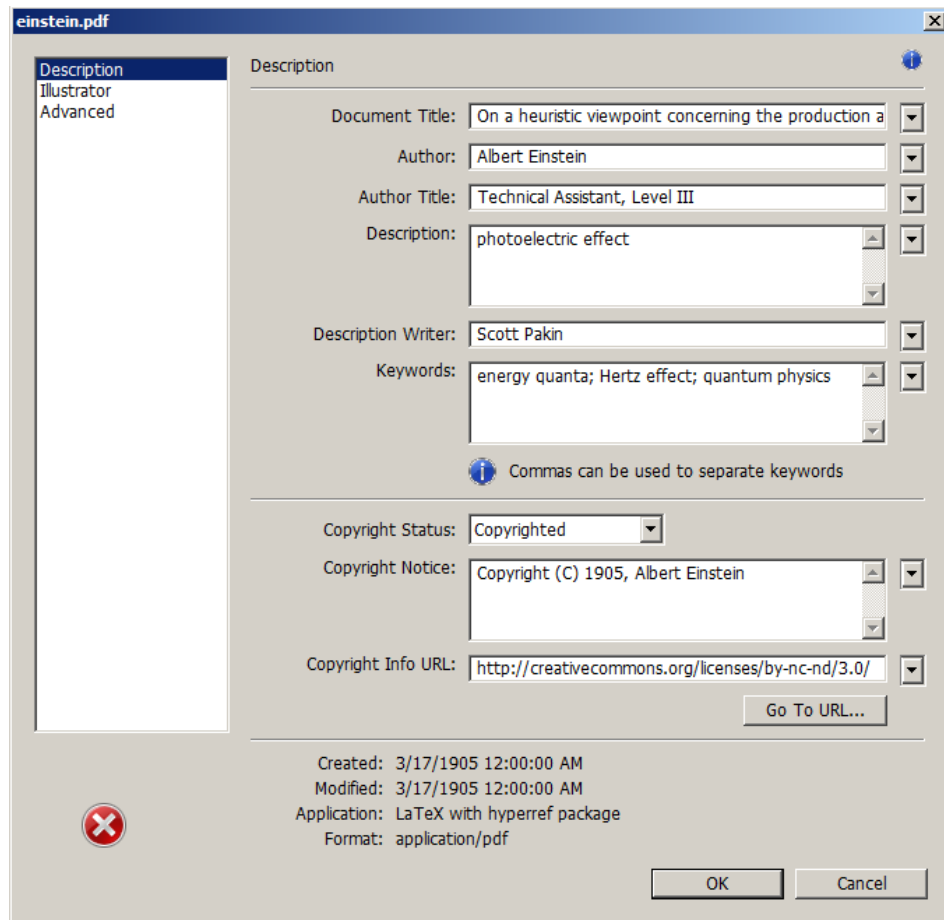


Figure 2: XMP metadata as it appears in Adobe Acrobat

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 `\hypersetup`.

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

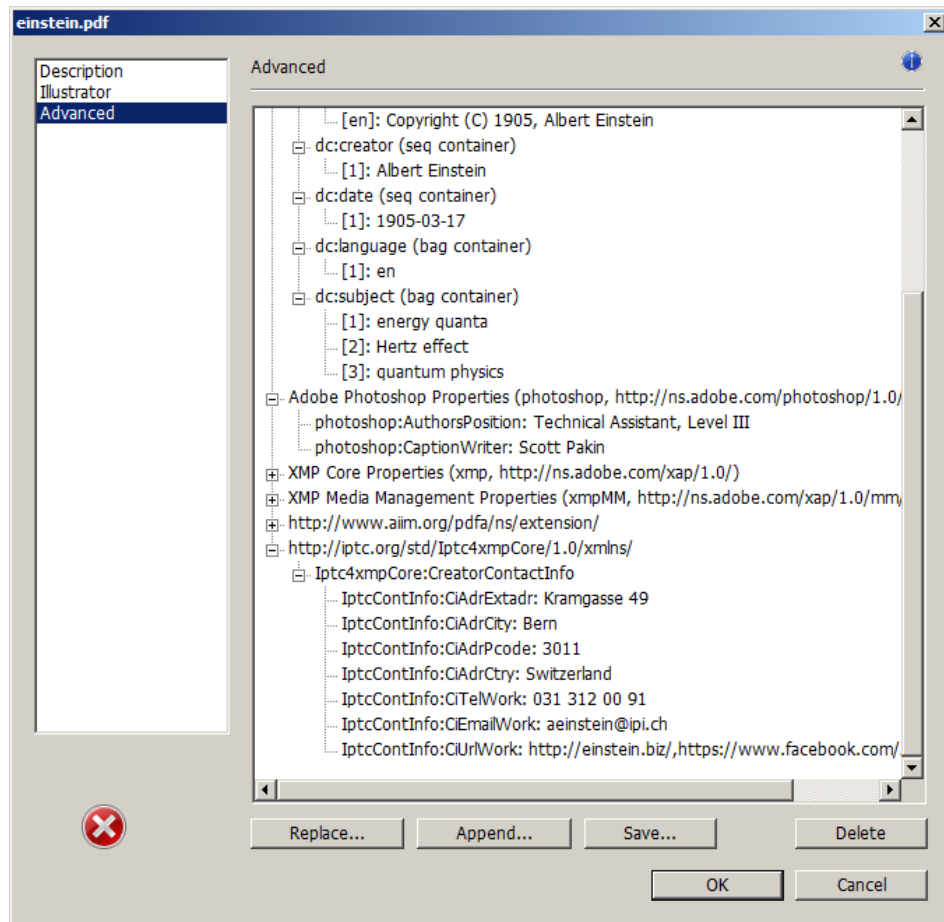


Figure 3: Additional XMP metadata as it appears in Adobe Acrobat

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 `LuaATEX` earlier than 0.85, `hyperxmp` leaves all PDF objects uncompressed. This is due to `LuaATEX` treating object compression

as a global parameter, unlike pdfL^AT_EX, which treats it as a local parameter. Hence, when hyperxmp requests that the XMP packet be left uncompressed, LuaL^AT_EX 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 LuaL^AT_EX v0.85 onwards.

2. X_YL^AT_EX (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., LuaL^AT_EX), (2) pass the `--output-driver="xdvipdfmx -z0"` option to X_YL^AT_EX to instruct xdvipdfmx to turn off all compression (which will of course make the PDF file substantially larger), or (3) postprocess the generated PDF file by loading it into the commercial version of Adobe Acrobat and re-saving it with the Save As... menu option.

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

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

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

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

`\xmpcomma`
`\xmpquote` 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, pdfauthor 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 \TeX terminology. This implies that any macros that are used in arguments are limited to a relatively small set of operations (such as conditionals and macro expansion) and must produce a string of text. Code (such as macro definitions and arithmetic operations) will be written to XMP as code, not as the result of executing the code.

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

```

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

```

Rather, it generates a `dc:rights` tag of the form “Copyright © =2=0=by-1by=02023, Scott Pakin”. The garbage in that line corresponds to the remnants of the `\printfdate` code after expanding all of the \TeX primitives and certain other control sequences it uses to the empty string. For example, “`\global\advance\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^AT_EX 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^AT_EX but not when using any other T_EX engine. Note that `hyperxmp` uses the file size from the *previous* run of LuaL^AT_EX 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` v5.9, the `hyperxmp` distribution includes a Perl script called `hyperxmp-add-bytecount` that edits a PDF file in place, adding or replacing the `prism:byteCount` property with one that specifies the final file size.³ Run the script as “`hyperxmp-add-bytecount <filename.pdf>`”.

The `latexmk` build tool can be configured to run `hyperxmp-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 LuaL^AT_EX, 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.

³The script was in fact introduced with `hyperxmp` v5.8 and was then called `add_byteCount`.

```

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

sub mycmd {
    my $retval = system @_;
    if ( $$Pdest =~ /\.pdf$/ ) {
        system 'hyperxmp-add-bytecount', $$Pdest;
    }
    return $retval;
}

```

Figure 4: latexmk configuration-file code for automatically invoking hyperxmp-add-bytecount every time a PDF file is generated

3 Implementation

This section presents the commented L^AT_EX source code for hyperxmp. Read this section only if you want to learn how hyperxmp is implemented.

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

```

1 \IfDocumentMetadataTF{%
2   \PackageWarning
3     {hyperxmp}
4     {Disabling hyperxmp because it is incompatible with PDF management}
5 }{}
6 \IfDocumentMetadataTF{\endinput}{-}

```

`\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).

```

7 \edef\hyxmp@dq@code{\the\catcode'\}
8 \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.

```

9 \@ifundefined{AddToHook}{%
10 \ifundefined{AtEndDocument}{%
11 \RequirePackage{atenddvi}
12 \let\hyxmp@at@end=\AtEndDvi
13 }{%
14 \let\hyxmp@at@end=\AtEndDocument
15 }
16 }{%
17 \def\hyxmp@at@end{\AddToHook{shipout/lastpage}}
18 }

```

`\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.

```

19 \def\hyxmp@set@jobname#1\relax{%
20 \ifnextchar"{\hyxmp@set@jobname@dbl}{\hyxmp@set@jobname@plain}#1\relax
21 }

```

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

```

\hyxmp@jobname 22 \def\hyxmp@set@jobname@dbl"#1"\relax{\xdef\hyxmp@jobname{#1}}

```

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

```

\hyxmp@jobname 23 \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 47) will fail because the filename does not in fact contain literal double quotes.

```

24 \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).

```

25 \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 page 5. For consistency with `hyperref`'s document-metadata naming conventions (which are in turn based on L^AT_EX's document-metadata naming conventions), we do not prefix metadata-related macro names with our package-specific `\hyxmp@` prefix. That is, we use names like `\pdfcopyright` instead of `\hyxmp@pdfcopyright`.

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

```

26 \RequirePackage{kvoptions}
27 \RequirePackage{pdfescape}
28 \RequirePackage{stringenc}
29 \RequirePackage{intcalc}
30 \RequirePackage{iftex}
31 \RequirePackage{ifmtarg}
32 \RequirePackage{etoolbox}
33 \RequirePackage{ifthen}

```

There are a few places where `hyperxmp` can take advantage of LuaT_EX features. To simplify the use of LuaT_EX we load the `luacode` package.

```

34 \ifLuaTeX
35   \RequirePackage{luacode}
36 \fi

```

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

```

37 \def\ifmtargexp#1{\expandafter\ifmtarg\expandafter{#1}}
38 \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.

```

39 \newcommand*\if@def@and@nonempty}[3]{%
40   \ifundefined{#1}{#3}{%
41     \expandafter\ifmtargexp\expandafter{\csname#1\endcsname}{#3}{#2}%
42   }%
43 }

```

`\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.

```

44 \newcommand{\hyxmp@pdfstringdef}[2]{%
45   \let\hyxmp@textunderscore=\textunderscore
46   \let\textunderscore=\hyxmp@uscore
47   \pdfstringdef{#1}{#2}%

```

```

48 \let\textunderscore=\hymp@textunderscore
49 }

```

`\@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.

```

50 \def\@pdfdatetime{}
51 \define@key{Hyp}{pdfdate}{%
52   \beginingroup
53     \Hy@unicodedefalse

```

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

```

54   \edef\next{%
55     \noexpand\hymp@pdfstringdef\noexpand\@pdfdatetime{%
56       \noexpand\hymp@as@xmp@date{#1}}%
57   }%
58   \next
59 \endgroup
60 }

```

`\@pdfmetadatetime` 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 `\@pdfmetadatetime` as an XMP-format string.

```

61 \def\@pdfmetadatetime{}
62 \define@key{Hyp}{pdfmetadate}{%
63   \beginingroup
64     \Hy@unicodedefalse

```

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

```

65   \edef\next{%
66     \noexpand\hymp@pdfstringdef\noexpand\@pdfmetadatetime{%
67       \noexpand\hymp@as@xmp@date{#1}}%
68   }%
69   \next
70 \endgroup
71 }

```

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

```

72 \def\@pdfcopyright{}
73 \define@key{Hyp}{pdfcopyright}{\hymp@pdfstringdef\@pdfcopyright{#1}}

```

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

```

74 \def\@pdftype{Text}
75 \define@key{Hyp}{pdftype}{\hymp@pdfstringdef\@pdftype{#1}}

```

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

```

76 \def\@pdflicenseurl{}
77 \define@key{Hyp}{pdflicenseurl}{\hymp@pdfstringdef\@pdflicenseurl{#1}}

```

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

```

78 \def\@pdfauthortitle{}
79 \define@key{Hyp}{pdfauthortitle}{\hymp@pdfstringdef\@pdfauthortitle{#1}}

```

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

```

80 \def\@pdfcaptionwriter{}
81 \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.

```

82 \def\@pdfmetalang{}
83 \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.

```

84 \def\hyxmp@no@bad@parts#1\relax{%
85   \ifnotmtarg{#1}{%
86     \PackageWarning{hyperxmp}{pdfapart and pdfuapart must be numeric}%
87   }%
88 }

```

`\@hyxmp@count` Define a temporary counter. The code previously used `\@tempcnta`, but this is no longer safe within `\pdfstringdef` as of more recent versions of hyperref.

```

89 \newcount\@hyxmp@count

```

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

```

90 \def\@pdfapart{}
91 \define@key{Hyp}{pdfapart}{%
92   \afterassignment\hyxmp@no@bad@parts\@hyxmp@count=0#1\relax
93   \hyxmp@pdfstringdef\@pdfapart{\the\@hyxmp@count}%
94 }

```

`\@pdfaconformance` Prepare to store the PDF/A conformance ID, which defaults to “b” if pdfa is passed to hyperref and `\@pdfapart` is empty.

```

95 \def\@pdfaconformance{}
96 \define@key{Hyp}{pdfaconformance}{%
97   \uppercase{\hyxmp@pdfstringdef\@pdfaconformance{#1}}%
98 }

```

`\@pdfuapart` Prepare to store the PDF/UA part ID.

```

99 \def\@pdfuapart{}
100 \define@key{Hyp}{pdfuapart}{%
101   \afterassignment\hyxmp@no@bad@parts\@hyxmp@count=0#1\relax
102   \hyxmp@pdfstringdef\@pdfuapart{\the\@hyxmp@count}%
103 }

```

`\hyxmp@set@pdfx@major` Parse pdfxstandard as “PDF/X-*<major><other>*”, setting `\hyxmp@pdfx@major` to *<major>*.

```

104 \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 `\@hyxmp@count`.

```

105 \def\hyxmp@set@pdfx@major@i PDF/X-{%
106   \afterassignment\hyxmp@set@pdfx@major@ii
107   \@hyxmp@count=%
108 }

```

```

\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 \@hyxmp@count to \@hyxmp@pdfx@major and discards the rest
of the PDF/X standard string.
109 \def\hyxmp@set@pdfx@major@ii#1!{%
110 \edef\hyxmp@pdfx@major{\the\@hyxmp@count}%
111 }

\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.
112 \newcommand*\hyxmp@check@std[2]{%
113 \ifthenelse{\equal{#1}{#2}}%
114 {\global\let\next=\relax}%
115 {}%
116 }%

\@pdfxstandard Prepare to store the PDF/X standard.
117 \def\@pdfxstandard{}
118 \def\hyxmp@pdfx@major{}
119 \define@key{Hyp}{pdfxstandard}{%
120 \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 68 above the
definition of \hyxmp@pdfx@id@schema, for example.)
121 \gdef\next{%
122 \PackageWarning{hyperxmp}{Unrecognized PDF/X standard ‘#1’}%
123 }%
124 \hyxmp@check@std{#1}{PDF/X-1a:2001}%
125 \hyxmp@check@std{#1}{PDF/X-1a:2003}%
126 \hyxmp@check@std{#1}{PDF/X-3:2002}%
127 \hyxmp@check@std{#1}{PDF/X-3:2003}%
128 \hyxmp@check@std{#1}{PDF/X-4}%
129 \hyxmp@check@std{#1}{PDF/X-4p}%
130 \hyxmp@check@std{#1}{PDF/X-5g}%
131 \hyxmp@check@std{#1}{PDF/X-5n}%
132 \hyxmp@check@std{#1}{PDF/X-5pg}%
133 \next

\hyxmp@pdfx@major Parse the PDF/X major version number from pdfxstandard and assign it to
\hyxmp@pdfx@major.
134 \hyxmp@set@pdfx@major{#1}%
135 }

\@pdfsource Prepare to store the document’s source, which defaults to the value of \jobname.
136 \edef\@pdfsource{\hyxmp@jobname.tex}
137 \define@key{Hyp}{pdfsource}{\hyxmp@pdfstringdef\@pdfsource{#1}}

\hyxmp@DocumentID Prepare to store a UUID that represents the document.
138 \def\hyxmp@DocumentID{}
139 \define@key{Hyp}{pdfdocumentid}{\hyxmp@pdfstringdef\hyxmp@DocumentID{#1}}

```

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

```

140 \def\hyxmp@InstanceID{}
141 \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”.

```

142 \def\@pdfversionid{1}
143 \define@key{Hyp}{pdfversionid}{\hyxmp@pdfstringdef\@pdfversionid{#1}}

```

`\ifdraft` Use the `ifdraft` package to determine if this is a draft or final document. The `\next` 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`.

```

144 \begingroup
145 \@ifpackageloaded{ifdraft}{%
146 \let\next=\relax
147 }{%
148 \let\ifdraft=\relax
149 \RequirePackage{ifdraft}%
150 \def\next{%
151 \expandafter\global\expandafter\let\csname ver@ifdraft.sty\endcsname=\relax
152 }%
153 }%

```

`\@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`.

```

154 \ifdraft{%
155 \gdef\@pdfrendition{draft}%
156 }{%
157 \gdef\@pdfrendition{default}%
158 }
159 \next
160 \endgroup
161 \define@key{Hyp}{pdfrendition}{\hyxmp@pdfstringdef\@pdfrendition{#1}}

```

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

```

162 \def\@pdfpublication{}
163 \define@key{Hyp}{pdfpublication}{\hyxmp@pdfstringdef\@pdfpublication{#1}}

```

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

```

164 \def\@pdfpubtype{}
165 \define@key{Hyp}{pdfpubtype}{\hyxmp@pdfstringdef\@pdfpubtype{#1}}

```

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

```

166 \def\@pdfbytes{}
167 \define@key{Hyp}{pdfbytes}{\hyxmp@pdfstringdef\@pdfbytes{#1}}

```


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

```

168 \def\@pdfnumpages{}
169 \define@key{Hyp}{pdfnumpages}{\hyxmp@pdfstringdef\@pdfnumpages{#1}}

```

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

```

170 \def\@pdfissn{}
171 \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.

```

172 \def\@pdfeissn{}
173 \define@key{Hyp}{pdfeissn}{\hyxmp@pdfstringdef\@pdfeissn{#1}}

```

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

```

174 \def\@pdfisbn{}
175 \define@key{Hyp}{pdfisbn}{\hyxmp@pdfstringdef\@pdfisbn{#1}}

```

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

```

176 \def\@pdfbookedition{}
177 \define@key{Hyp}{pdfbookedition}{\hyxmp@pdfstringdef\@pdfbookedition{#1}}

```

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

```

178 \def\@pdfpublisher{}
179 \define@key{Hyp}{pdfpublisher}{\hyxmp@pdfstringdef\@pdfpublisher{#1}}

```

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

```

180 \def\@pdfvolumenum{}
181 \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.

```

182 \def\@pdfissuenum{}
183 \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.

```

184 \def\@pdfpagerange{}
185 \define@key{Hyp}{pdfpagerange}{\hyxmp@pdfstringdef\@pdfpagerange{#1}}

```

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

```

186 \def\@pdfdoi{}
187 \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.

```

188 \def\@pdfurl{}
189 \define@key{Hyp}{pdfurl}{\hyxmp@pdfstringdef\@pdfurl{#1}}

```

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

```

190 \def\@pdfidentifier{}
191 \define@key{Hyp}{pdfidentifier}{\hyxmp@pdfstringdef\@pdfidentifier{#1}}

```

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

```
192 \def\@pdfsubtitle{}
193 \define@key{Hyp}{pdfsubtitle}{\hyxmp@pdfstringdef\@pdfsubtitle{#1}}
```

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

```
194 \def\@pdfpubstatus{}
195 \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.

```
196 \def\@pdfcontactaddress{}
197 \define@key{Hyp}{pdfcontactaddress}{%
198   \let\xmpcomma=\hyxmp@comma
199   \def\xmpquote##1{##1}%
200   \hyxmp@pdfstringdef\@pdfcontactaddress{#1}%
201   \def\xmpcomma{,}%
202   \let\xmpquote=\relax
203 }
```

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

```
204 \def\@pdfcontactcity{}
205 \define@key{Hyp}{pdfcontactcity}{\hyxmp@pdfstringdef\@pdfcontactcity{#1}}
```

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

```
206 \def\@pdfcontactregion{}
207 \define@key{Hyp}{pdfcontactregion}{\hyxmp@pdfstringdef\@pdfcontactregion{#1}}
```

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

```
208 \def\@pdfcontactpostcode{}
209 \define@key{Hyp}{pdfcontactpostcode}{\hyxmp@pdfstringdef\@pdfcontactpostcode{#1}}
```

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

```
210 \def\@pdfcontactcountry{}
211 \define@key{Hyp}{pdfcontactcountry}{\hyxmp@pdfstringdef\@pdfcontactcountry{#1}}
```

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

```
212 \def\@pdfcontactphone{}
213 \define@key{Hyp}{pdfcontactphone}{\hyxmp@pdfstringdef\@pdfcontactphone{#1}}
```

```

\@pdfcontactemail Prepare to store the email address of the document's contact person/institution.
214 \def\@pdfcontactemail{}
215 \define@key{Hyp}{pdfcontactemail}{\hyxmp@pdfstringdef\@pdfcontactemail{#1}}

\@pdfcontacturl Prepare to store the URL of the document's contact person/institution.
216 \def\@pdfcontacturl{}
217 \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.
218 \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.
219 \def\hyxmp@suppress@pdf@info{%
220   \global\let\PDF@FinishDoc=\@empty
221   \PackageWarningNoLine{hyperxmp}{%
222     Suppressing the _entire_ PDF Info dictionary.\MessageBreak
223     Please notify the hyperxmp maintainer%
224   }%
225 }%
226 \let\next=\relax
227 \patchcmd
228   {\PDF@FinishDoc}%
229   {/Author(\@pdfauthor)}%
230   {}%
231   {}%
232   {\let\next=\hyxmp@suppress@pdf@info}%
233 \patchcmd
234   {\PDF@FinishDoc}%
235   {/Keywords(\@pdfkeywords)}%
236   {}%
237   {}%
238   {\let\next=\hyxmp@suppress@pdf@info}%
239 \next
240 }

241 \define@key{Hyp}{keeppdfinfo}[true]{%
242   \gdef\hyxmp@no@info@lists{}%
243 }

```

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

For lack of a better approach, `hyperxmp` redefines `\ProcessKeyvalOptions` to alter the way `hyperref` processes `pdfauthor` and `pdfkeywords`. This is somewhat

heavy-handed as it gets executed for *every* subsequently loaded package that uses `\ProcessKeyvalOptions`, but at least it does what we need. `hyperxmp` also redefines `\hypersetup` to do the same thing. This is required in case `hyperref` is loaded before `hyperxmp`.

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

```
\hyxmp@pdfkeywords 244 \def\hyxmp@pdfauthor{}
245 \def\hyxmp@pdfkeywords{}
```

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

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

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

```
247 \@ifundefined{KV@Hyp@pdfauthor}{}{%
248 \@ifundefined{hyxmp@Hyp@pdfauthor}{%
249 \expandafter\let\expandafter\hyxmp@Hyp@pdfauthor
250 \csname KV@Hyp@pdfauthor\endcsname
251 }{}%
252 }%
```

`\KV@Hyp@pdfauthor` Redefine `\KV@Hyp@pdfauthor` to process its argument twice. The first time, `\xmpcomma` is defined as a placeholder character (`\hyxmp@comma`) and `\xmpquote` as the identity function. The result is stored in `\hyxmp@pdfauthor` for use in structured lists (those surrounding each entry with `<rdf:li>`). The second time, `\and` is defined as an ordinary comma, and `\xmpquote` is defined as a macro that puts its argument within double quotes. The result is stored in `\@pdfauthor` for use in unstructured lists (those in which the entire list appears within a single pair of tags). In case `pdfauthor` is left unspecified and we copy `\author`'s argument to `pdfauthor`, we temporarily redefine `\and` as the list separator when producing a structured list and as "and" when producing an unstructured list.

```
253 \define@key{Hyp}{pdfauthor}{%
254 \let\xmpcomma=\hyxmp@comma
255 \def\xmpquote####1{####1}%
256 \let\hyxmp@and=\and
257 \def\and{,%}
258 \hyxmp@Hyp@pdfauthor{##1}%
259 \global\let\hyxmp@pdfauthor=\@pdfauthor
260 \def\and{and\space}%
261 \def\xmpcomma{,%}
262 \def\xmpquote####1{"####1"%}
263 \hyxmp@Hyp@pdfauthor{##1}%
264 \def\xmpcomma{,%}
265 \let\xmpquote=\relax
266 \let\and=\hyxmp@and
267 }%
```

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

```

268 \@ifundefined{KV@Hyp@pdfkeywords}{}{-%
269   \@ifundefined{hyxmp@Hyp@pdfkeywords}{-%
270     \expandafter\let\expandafter\hyxmp@Hyp@pdfkeywords
271       \csname KV@Hyp@pdfkeywords\endcsname
272   }{-%
273 }%

```

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

```

274 \define@key{Hyp}{pdfkeywords}{-%
275   \let\xmpcomma=\hyxmp@comma
276   \def\xmpquote####1{####1}%
277   \hyxmp@Hyp@pdfkeywords{##1}%
278   \global\let\hyxmp@pdfkeywords=\@pdfkeywords
279   \def\xmpcomma{,}%
280   \def\xmpquote####1{"####1"%
281   \hyxmp@Hyp@pdfkeywords{##1}%
282   \def\xmpcomma{,}%
283   \let\xmpquote=\relax
284 }%
285 }

```

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

```

286 \let\hyxmp@ProcessKeyvalOptions=\ProcessKeyvalOptions
287 \renewcommand*{\ProcessKeyvalOptions}{-%
288   \global\let\ProcessKeyvalOptions=\hyxmp@ProcessKeyvalOptions
289   \hyxmp@redefine@Hyp
290   \hyxmp@ProcessKeyvalOptions
291 }

```

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

```

292 \let\hyxmp@hypersetup=\hypersetup
293 \def\hypersetup{-%
294   \hyxmp@redefine@Hyp
295   \hyxmp@hypersetup
296 }

```

`\hyxmp@concat@metadata` Assume that if the document loaded either `babel` or `polyglossia` it will eventually `\hyxmp@aep@toks` 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.

```

297 \edef\hyxmp@concat@metadata{}
298 \expandafter\hyxmp@aep@toks\expandafter=\expandafter{-%
299   \the\hyxmp@aep@toks
300   \AtEndPreamble{-%

```

```

301 \ifpackageloaded{babel}{%
302   \edef\hyxmp@concat@metadata{babel}%
303 }{%
304   \ifpackageloaded{polyglossia}{%
305     \edef\hyxmp@concat@metadata{polyglossia}%
306   }{%
307     }%
308   }%
309 }%
310 }

```

`\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.

```

311 \newcommand*{\hyxmp@warn@if@no@metadata}{%
312   \edef\hyxmp@concat@metadata{%
313     \hyxmp@concat@metadata
314     \baseurl
315     \pdfauthor
316     \pdfauthortitle
317     \pdfbookedition
318     \pdfbytes
319     \pdfcaptionwriter
320     \pdfcontactaddress
321     \pdfcontactcity
322     \pdfcontactcountry
323     \pdfcontactemail
324     \pdfcontactphone
325     \pdfcontactpostcode
326     \pdfcontactregion
327     \pdfcontacturl
328     \pdfcopyright
329     \pdfcreationdate
330     \pdfdatetime
331     \pdfdoi
332     \pdfeissn
333     \pdfidentifier
334     \pdfisbn
335     \pdfissn
336     \pdfissuenum
337     \pdfkeywords
338     \pdflang
339     \pdflicenseurl
340     \pdfmetadatetitle
341     \pdfmoddate
342     \pdfnumpages
343     \pdfpagerange
344     \pdfpublication
345     \pdfpubtype
346     \pdfsubject
347     \pdfsubtitle
348     \pdftitle

```

```

349   \@pdfupart
350   \@pdfurl
351   \@pdfvolumenum
352   \@pdfxstandard
353 }%
354 \ifx\hyxmp@concat@metadata\@empty
355   \PackageWarningNoLine{hyperxmp}{%
356     \hyxmp@jobname.tex did not specify any metadata to\MessageBreak
357     include in the XMP packet.\space\space Please see the\MessageBreak
358     hyperxmp documentation for instructions on how to\MessageBreak
359     provide metadata values to hyperxmp}%
360 \fi
361 }

```

`\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.

```
362 \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.

```

363   \ifHy@pdfa
364     \@ifmtargexp{\@pdfapart}{%
365       \PackageWarningNoLine{hyperxmp}{%
366         'pdfa' was passed to hyperref, but 'pdfapart' was\MessageBreak
367         not specified.\space\space Setting pdfapart to '1' and\MessageBreak
368         pdfaconformance to 'B'%
369       }%
370     \gdef\@pdfapart{1}%
371     \gdef\@pdfaconformance{B}%
372   }%
373   {}%
374 \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).

```

375   \edef\hyxmp@standards{%
376     \@pdfapart
377     \@pdfxstandard
378     \@pdfupart
379   }%

```

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

```

380   \@ifnotmtargexp{\hyxmp@standards}{%
381     \@ifmtargexp{\@pdftitle}{%
382       \PackageWarningNoLine{hyperxmp}{%
383         Missing pdftitle (required for PDF standards\MessageBreak
384         compliance)%
385     }%
386   }%
387   {}%
388 }%
389 }

```

`\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.

```

390 \expandafter\hyxmp@aep@toks\expandafter=\expandafter{%
391   \the\hyxmp@aep@toks
392   \AtEndPreamble{%
393     \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.

```

394   \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 L^AT_EX 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` → `pdfauthor`.

One tricky bit is that the standard L^AT_EX 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.

```

395   \begingroup
396     \let\@latex@warning@no@line=\gobble
397     \hyxmp@use@first@valid{pdftitle}{\@pdftitle}{%
398       \scr@subject@var,%
399       \@title
400     }%
401     \hyxmp@use@first@valid{pdfauthor}{\@pdfauthor}{%
402       \scr@fromname@var,%
403       \@author
404     }%
405   \endgroup
406 }%
407 }

```

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.

```

408 \hyxmp@at@end{%

```

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

```

409   \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.


```
410 \hyxmp@check@standards
```

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

```
411 \hyxmp@warn@if@no@metadata
412 \hyxmp@embed@packet
413 }
```

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 provided to a KOMA letter class (mobile and landline).

```
414 \newcommand*{\hyxmp@set@koma@phones}{%
415   \begingroup
416     \Hy@unicodedefalse
417     \@if@def@and@nonempty{scr@frommobilephone@var}{%
418       \@if@def@and@nonempty{scr@fromphone@var}{%
419         \hyxmp@pdfstringdef\hyxmp@koma@phones{\scr@frommobilephone@var,\scr@fromphone@var}%
420       }{%
421         \hyxmp@pdfstringdef\hyxmp@koma@phones{\scr@frommobilephone@var}%
422       }%
423     }{%
424       \@if@def@and@nonempty{scr@fromphone@var}{%
425         \hyxmp@pdfstringdef\hyxmp@koma@phones{\scr@fromphone@var}%
426       }{%
427         }%
428     }%
429   \endgroup
430 }
```

`\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.

```
431 \newcommand*{\hyxmp@use@first@valid}[3]{%
432   \@ifmtargexp{#2}{%
433     \hyxmp@use@first@valid{i{#1}#3,!,%
434   }%
435   }%
436 }
```

`\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`.

```

437 \def\hyxmp@use@first@valid@i#1#2,{%
438   \def\next{\hyxmp@use@first@valid@i{#1}}%
439   \ifx#2!%
440     \let\next=\relax
441   \else
442     \ifx#2\undefined
443     \else
444       \@ifnotmtargexp{#2}{%
445         \hypersetup{#1={#2}}%
446         \def\next##1!,{)%
447       }%
448     \fi
449   \fi
450   \next
451 }

```

`\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).

```

452 \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.

```

453   \@if@def@and@nonempty{@pdflang}{%
454     \let\hyxmp@dc@lang=\@pdflang
455   }{%
456     \hyxmp@detect@langs
457   }%

```

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

```

458   \ifx\@pdfmetalang\@empty
459     \ifx\@pdflang\@empty
460       \let\@pdfmetalang=\hyxmp@x@default
461     \else
462       \edef\@pdfmetalang{\@pdflang}%
463     \fi
464   \fi

```

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

```

465   \hyxmp@use@first@valid{pdfcontactemail}{\@pdfcontactemail}{%
466     \scr@fromemail@var
467   }%
468   \hyxmp@set@koma@phones
469   \hyxmp@use@first@valid{pdfcontactphone}{\@pdfcontactphone}{%
470     \hyxmp@koma@phones
471   }%
472   \hyxmp@use@first@valid{pdfcontacturl}{\@pdfcontacturl}{%
473     \scr@fromurl@var
474   }%

```

```

475 \hyxmp@use@first@valid{pdfsubtitle}{\@pdfsubtitle}{%
476   \@subtitle
477 }%
478 \hyxmp@use@first@valid{pdfpublisher}{\@pdfpublisher}{%
479   \@publishers
480 }%

```

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.

```

481 \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 \XeTeX . In this case we explicitly define `\@pdfcreationdate` as `\hyxmp@today@pdf` to prevent the `xdvi` 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.

```

482 \@ifundefined{XeTeXversion}{}{%
483   \@ifmtargexp{\@pdfcreationdate}{%
484     \let\@pdfcreationdate=\hyxmp@today@pdf
485   }%
486   {}%
487 }%

```

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

```

488 \hyxmp@query@self
489 }

```

Determine the size of the output file from the *previous* run of \LuaTeX . 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 \pdfTeX ’s `\pdffilesize` primitive to query the size of `\jobname.pdf` under \pdfTeX . 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.

```

490 \ifLuaTeX

```

Now that we know we’re running \LuaTeX 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, \LuaTeX 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.

```
491 \begin{luacode*}
492 function get_pdf_size (bname)
```

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

```
493   local odir = ""
494   for _, opt in ipairs(arg) do
495     local m = string.match(opt, "%-output%-d.-=(.*)")
496     if m then
497       odir = m
498     end
499   end
```

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

```
500   local sep = "/"
501   if os.type == "windows" or os.type == "msdos" then
502     sep = "\\\"
503   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”

```
504   local dlast = string.sub(odir, -1)
505   if odir == "" or dlast == sep or (dlast == ":" and sep == "\\\" then
506     sep = ""
507   end
508   local fname = odir .. sep .. bname .. ".pdf"
```

Query the file size and return it.

```
509   local nbytes = lfs.attributes(fname, "size")
510   return nbytes
511 end
512 \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`.

```
513 \xdef\hyxmp@prev@pdf@size{%
```

```

514 \luadirect{
515 nbytes = get_pdf_size("\hyxmp@jobname")
516 if nbytes then
517 tex.write(nbytes)
518 end
519 }%
520 }%
521 \fi

```

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

```
522 \newcommand*{\hyxmp@query@self}{%
```

L^AT_EX's `totalpages` counter tracks the number of pages written. We use this mechanism to assign `\@pdfnumpages`.

```

523 \if@def@and@nonempty{@pdfnumpages}{%
524 }{%
525 \xdef\@pdfnumpages{\thetotalpages}%
526 }%

```

If `pdfbytes` hasn't been set, set it to the output file's size from the previous run.

```

527 \hyxmp@use@first@valid{pdfbytes}{\@pdfbytes}{%
528 \hyxmp@prev@pdf@size
529 }%
530 }

```

`\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.

```

531 \newcommand*{\hyxmp@parse@acmart}{%
532 \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.

```
533 \let\@author=\@gobble
```

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

```

534 \def\email##1##2{%
535 \def\hyxmp@address@val{##2}%
536 \hyxmp@use@first@valid{pdfcontactemail}{\@pdfcontactemail}{%
537 \hyxmp@address@val
538 }%
539 }%

```

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

```

\hyxmp@address@val 540 \def\streetaddress##1{%
541 \def\hyxmp@address@val{##1}%
542 \hyxmp@use@first@valid{pdfcontactaddress}{\@pdfcontactaddress}{%
543 \hyxmp@address@val
544 }%
545 }%

```

```

\city \city wraps the author's city name.
\hyxmp@address@val 546 \def\city##1{%
547 \def\hyxmp@address@val{##1}%
548 \hyxmp@use@first@valid{pdfcontactcity}{\@pdfcontactcity}{%
549 \hyxmp@address@val
550 }%
551 }%

\state \state wraps the author's state or region name.
\hyxmp@address@val 552 \def\state##1{%
553 \def\hyxmp@address@val{##1}%
554 \hyxmp@use@first@valid{pdfcontactregion}{\@pdfcontactregion}{%
555 \hyxmp@address@val
556 }%
557 }%

\country \country wraps the author's country name.
\hyxmp@address@val 558 \def\country##1{%
559 \def\hyxmp@address@val{##1}%
560 \hyxmp@use@first@valid{pdfcontactcountry}{\@pdfcontactcountry}{%
561 \hyxmp@address@val
562 }%
563 }%

\postcode \postcode wraps the author's postal code.
\hyxmp@address@val 564 \def\postcode##1{%
565 \def\hyxmp@address@val{##1}%
566 \hyxmp@use@first@valid{pdfcontactpostcode}{\@pdfcontactpostcode}{%
567 \hyxmp@address@val
568 }%
569 }%

\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.
570 \def\affiliation##1##2{%
571 ##2%
572 \let\affiliation=\@gobbletwo
573 }%

```

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.

```

574 \setbox0=\hbox{\addresses}%
575 \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.

```

576 \if@def@and@nonempty{@acmDOI}{%
577   \IfSubStr{\@acmDOI}{10.1145/1122445.1122456}{-}{%
578     \IfSubStr{\@acmDOI}{10.1145/nnnnnnn.nnnnnnn}{-}{%
579       \hyxmp@use@first@valid{pdfdoi}{\@pdfdoi}{%
580         \@acmDOI
581       }%
582     }%
583   }%
584 }%
585 {}%

```

`\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.

```

586 \if@def@and@nonempty{@acmISBN}{%
587   \IfSubStr{\@acmISBN}{XXXX}{-}{%
588     \IfSubStr{\@acmISBN}{xxxx}{-}{%
589       \def\hyxmp@strip@isbn@date##1/##2!{##1}%
590       \edef\hyxmp@acm@isbn{%
591         \expandafter\hyxmp@strip@isbn@date\@acmISBN/!%
592       }%
593       \hyxmp@use@first@valid{pdfisbn}{\@pdfisbn}{%
594         \hyxmp@acm@isbn
595       }%
596     }%
597   }%
598 }%
599 {}%

```

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

```

600 \def\hyxmp@acm@publisher{Association for Computing Machinery}%
601 \hyxmp@use@first@valid{pdfpublisher}{\@pdfpublisher}{%
602   \hyxmp@acm@publisher
603 }%

```

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

```

604 \hyxmp@use@first@valid{pdfpublication}{\@pdfpublication}{%
605   \@journalName,%
606   \@acmBooktitle,%
607   \@acmConference
608 }%

```

`\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).

```

609 \if@ACM@journal
610   \def\hyxmp@acm@pubtype{journal}%
611 \else
612   \def\hyxmp@acm@pubtype{book}%
613 \fi

```

```

614 \hyxmp@use@first@valid{pdfpubtype}{\@pdfpubtype}{%
615   \hyxmp@acm@pubtype
616 }%

```

Journal articles have a volume and issue number.

```

617 \hyxmp@use@first@valid{pdfvolumenum}{\@pdfvolumenum}{%
618   \@acmVolume
619 }%
620 \hyxmp@use@first@valid{pdfissuenum}{\@pdfissuenum}{%
621   \@acmNumber
622 }%
623 }

```

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

```

624 \@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.

```

625 \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.

```

626 \newcommand*{\hyxmp@detect@langs}{%
627   \@ifundefined{mainbcp47id}{%
628     \@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.

```

629   \let\@pdflang=\@empty
630 }%

```

`\hyxmp@dc@lang` Use `babel`'s `\LocaleForEach` and `\getlocaleproperty` to set `\@pdflang` to the document's main language and `\hyxmp@dc@lang` to a comma-separated list of all languages used.

```

\hyxmp@lang@tag
\hyxmp@lang@name
\@pdflang
631   \BabelEnsureInfo
632   \LocaleForEach{%
633     \getlocaleproperty\hyxmp@lang@tag{##1}{identification/tag.bcp47}%
634     \ifx\hyxmp@dc@lang\@empty
635       \xdef\hyxmp@dc@lang{\hyxmp@lang@tag}%
636     \else
637       \xdef\hyxmp@dc@lang{\hyxmp@dc@lang,\hyxmp@lang@tag}%
638     \fi
639     \def\hyxmp@lang@name{##1}%
640     \ifx\hyxmp@lang@name\bb1@main@language
641       \edef\@pdflang{\hyxmp@lang@tag}%
642     \fi
643   }%
644 }%
645 }%

```


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

```
646 \xdef\@pdflang{\csname mainbcp47id\endcsname}%
647 \edef\hyxmp@dc@lang{\xpg@bcp@loaded}%
648 }%
649 }
```

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^AT_EX 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 `<scott+hyxmp@pakin.org>`) (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^AT_EX `\@elt`-separated elements.

```
\hyxmp@commas@to@list Given a macro name (#1) and a comma-separated list (#2), define the macro name
as the elements of the list, each preceded by \@elt. (Executing the macro therefore
applies \@elt to each element in turn.)
650 \newcommand*\hyxmp@commas@to@list}[2]{%
651 \gdef#1{%
652 \expandafter\hyxmp@commas@to@list@i\expandafter#1#2,,%
653 }
```

```
\hyxmp@commas@to@list@i Recursively construct macro #1 from comma-separated list #2. Stop if #2 is empty.
\next 654 \def\hyxmp@commas@to@list@i#1#2,{%
655 \gdef\hyxmp@sublist{#2}%
656 \ifx\hyxmp@sublist\@empty
657 \let\next=\relax
658 \else
659 \hyxmp@trimspaces\hyxmp@sublist
660 \@cons{#1}{\hyxmp@sublist}%
661 \def\next{\hyxmp@commas@to@list@i{#1}}%
662 \fi
663 \next
664 }
```

`\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.

```
665 \def\xmpcomma{,}%
```

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

```
666 \bgroup
667 \catcode'^^C=11
668 \gdef\hyxmp@comma{^^C}
669 \egroup
```

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

```
670 \bgroup
671 \catcode'^^U=11
672 \gdef\hyxmp@uscore{^^U}
673 \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.

```
674 \let\xmpquote=\relax
```

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

```
675 \bgroup
676 \catcode'\~=12%
677 \gdef\xmptilde{~}%
678 \egroup
```

`\XMPTruncateList` As a workaround for the inability of older Adobe Acrobat versions to display `\hyxmp@temp@str` author lists correctly we introduce a hack that replaces a list with its first element.

`\hyxmp@temp@list` One can then write “`\XMPTruncateList{pdfauthor}`” and have Adobe Acrobat `\@elt` display the author list correctly.

```
679 \newcommand{\XMPTruncateList}[1]{%
680   \PackageWarning{hyperxmp}{%
681     \noexpand\XMPTruncateList has been deprecated since\MessageBreak
682     hyperxmp 4.0 and may be removed in future\MessageBreak
683     versions of the package. \noexpand\XMPTruncateList\MessageBreak
684     was found}%
685   \edef\hyxmp@temp@str{\csname hyxmp@#1\endcsname}%
686   \hyxmp@commas@to@list{\hyxmp@temp@list}{\hyxmp@temp@str}%
687   \def\@elt##1{%
688     \expandafter\gdef\csname @#1\endcsname{##1}%
689     \let\@elt=\@gobble
690   }
691   \hyxmp@temp@list
692 }
```

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:20230219155949-07’00’) [4], while XMP timestamps are of the form “YYYY-MM-DDThh:mm:ss+TT’tt’” (e.g., 2023-02-19T15:59:49-07:00) [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 693 \def\hyxmp@first@char#1{\hyxmp@first@char@i#1\relax}
694 \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.

```
695 \def\hyxmp@as@xmp@date#1{%
696   \expandafter\ifnum\expandafter'\hyxmp@first@char@i#1\relax='D
697   \hyxmp@pdf@to@xmp@date{#1}%
698   \else
699     #1%
700   \fi
701 }
```

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

```
702 \def\hyxmp@pdf@to@xmp@date#1:#2#3#4#5#6#7#8#9{%
703   #2#3#4#5-#6#7-#8#9%
704   \hyxmp@parse@time
705 }
```

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

```
706 \def\hyxmp@parse@time#1#2#3#4#5#6{%
707   T#1#2:#3#4:#5#6%
708   \hyxmp@parse@tz@char
709 }
```

`\hyxmp@parse@tz@char` This is another helper function for `\hyxmp@pdf@to@xmp@date`. So far, the date and time have been parsed. `\hyxmp@parse@tz@char` parses the first character of the timezone descriptor. This can be one of “+” for eastern timezones (UTC+*x*, including Asia, Oceania, and most of Europe), “-” for western timezones (UTC-*x*, primarily the Americas), or “Z” for Zulu time (UTC+0). Timezones beginning with “+” or “-” are followed by an offset in hours and minutes (parsed by `\hyxmp@parse@tz`; timezones beginning with “Z” are not.

```
710 \def\hyxmp@parse@tz@char#1{%
711   #1%
712   \ifx#1-%
713     \expandafter\hyxmp@parse@tz
```

```

714 \else
715   \ifx#1+%
716     \expandafter\hyxmp@parse@tz
717   \fi
718 \fi
719 }

```

`\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.

```

720 \def\hyxmp@parse@tz#1'#2' {%
721   #1:#2%
722 }

```

`\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.

```

723 \def\hyxmp@as@pdf@date#1 {%
724   \expandafter\ifx\hyxmp@first@char@i#1\relax D%
725   #1%
726 \else
727   \hyxmp@xmp@to@pdf@date{#1}%
728 \fi
729 }

```

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

```

730 \def\hyxmp@xmp@to@pdf@date#1 {%
731   D:\hyxmp@xmp@to@pdf@date@i#1\relax\relax
732 }

```

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

```

733 \def\hyxmp@xmp@to@pdf@date@i#1#2#3#4#5#6 {%
734   #1#2#3#4%
735   \ifx#5-%
736     \expandafter\hyxmp@xmp@to@pdf@date@ii\expandafter#6%
737   \fi
738 }

```

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

```

739 \def\hyxmp@xmp@to@pdf@date@ii#1#2#3#4 {%
740   #1#2%
741   \ifx#3-%
742     \expandafter\hyxmp@xmp@to@pdf@date@iii\expandafter#4%
743   \fi
744 }

```

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

```

745 \def\hyxmp@xmp@to@pdf@date@iii#1#2#3#4 {%
746   #1#2%
747   \ifx#3T%
748     \expandafter\hyxmp@xmp@to@pdf@date@iv\expandafter#4%
749   \fi
750 }

```

```

\hyxmp@xmp@to@pdf@date@iv Parse the hour for \hyxmp@xmp@to@pdf@date.
751 \def\hyxmp@xmp@to@pdf@date@iv#1#2#3#4{%
752 #1#2%
753 \ifx#3:%
754 \expandafter\hyxmp@xmp@to@pdf@date@v\expandafter#4%
755 \fi
756 }

\hyxmp@xmp@to@pdf@date@v Parse the minute for \hyxmp@xmp@to@pdf@date.
757 \def\hyxmp@xmp@to@pdf@date@v#1#2#3#4{%
758 #1#2%
759 \ifx#3:%
760 \expandafter\hyxmp@xmp@to@pdf@date@vi\expandafter#4%
761 \fi
762 }

\hyxmp@gobbletwo This is exactly the same as LATEX 2ε's \@gobbletwo but needs to be a different
literal for \hyxmp@xmp@to@pdf@date@vii's pattern-matching to work.
763 \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).
764 \def\hyxmp@xmp@to@pdf@date@vi#1#2#3#4{%
765 #1#2%
766 \ifx#3+%
767 +\expandafter\hyxmp@xmp@to@pdf@date@vii
768 \fi
769 \ifx#3-%
770 -\expandafter\hyxmp@xmp@to@pdf@date@vii
771 \fi
772 \ifx#3Z%
773 Z%
774 \fi
775 \ifx#3\relax
776 \expandafter\hyxmp@gobbletwo
777 \fi
778 \@gobbletwo #4%
779 }

\hyxmp@xmp@to@pdf@date@vii Parse the time-zone hours for \hyxmp@xmp@to@pdf@date.
780 \def\hyxmp@xmp@to@pdf@date@vii#1\@gobbletwo#2#3#4#5{%
781 #2#3%
782 \ifx#4:%
783 \expandafter\hyxmp@xmp@to@pdf@date@viii\expandafter#5%
784 \fi
785 }

\hyxmp@xmp@to@pdf@date@viii Parse the time-zone minutes for \hyxmp@xmp@to@pdf@date.
786 \def\hyxmp@xmp@to@pdf@date@viii#1#2#3#4{%
787 '#1#2'%
788 }

```

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

```
789 \def\hyxmp@today@xmp@define#1{%
```

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

```
790 \xdef#1{\the\year}%
791 \ifnum\month<10
792   \xdef#1{#1-0\the\month}%
793 \else
794   \xdef#1{#1-\the\month}%
795 \fi
796 \ifnum\day<10
797   \xdef#1{#1-0\the\day}%
798 \else
799   \xdef#1{#1-\the\day}%
800 \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.

```
801 \@hyxmp@count=\time
802 \divide\@hyxmp@count by 60
803 \ifnum\@hyxmp@count<10
804   \xdef#1{#1T0\the\@hyxmp@count}%
805 \else
806   \xdef#1{#1T\the\@hyxmp@count}%
807 \fi
808 \multiply\@hyxmp@count by -60
809 \advance\@hyxmp@count by \time
810 \ifnum\@hyxmp@count<10
811   \xdef#1{#1:0\the\@hyxmp@count}%
812 \else
813   \xdef#1{#1:\the\@hyxmp@count}%
814 \fi
815 \xdef#1{#1Z}%
816 }
```

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

```
817 \def\hyxmp@try@today#1#2{%
818   \@ifmtargexp{\hyxmp@today@xmp}{%
819     \@ifundefined{#1}{-}{#2}%
820   }%
821   {}%
822 }
```

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

```
823 \def\hyxmp@today@xmp{}
```

Case 1: `\pdfcreationdate` is defined (pdf \LaTeX and pre-0.85 Lua \LaTeX).

```
824 \hyxmp@try@today{\pdfcreationdate}{%
```

```

825 \edef\hyxmp@today@xmp{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
826 }

```

Case 2: `\pdffeedback` is defined (Lua^AT_EX 0.85+).

```

827 \hyxmp@try@today{pdffeedback}{%
828 \edef\hyxmp@today@xmp{\expandafter\hyxmp@pdf@to@xmp@date\pdffeedback creationdate}%
829 }

```

`\hyxmp@timestamp` Case 3: `\filemoddate` is defined (X_YL^AT_EX). In this case, we treat the timestamp of the job's `.log` file as the current date/time.

```

830 \hyxmp@try@today{filemoddate}{%
831 \edef\hyxmp@today@xmp{\filemoddate{\hyxmp@jobname.log}}%
832 \edef\next{%
833 \edef\noexpand\hyxmp@today@xmp{\noexpand\hyxmp@as@xmp@date{\hyxmp@today@xmp}}%
834 }%
835 \next
836 }%

```

Case 4: None of the above. Do the best we can using the available T_EX primitives (`\year`, `\month`, `\day`, and `\time`).

```

837 \hyxmp@try@today{year}{%
838 \hyxmp@today@xmp@define\hyxmp@today@xmp
839 }

```

`\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`.

```

840 \expandafter\edef\expandafter\hyxmp@today@pdf\expandafter{%
841 \expandafter\hyxmp@xmp@to@pdf@date\expandafter{\hyxmp@today@xmp}%
842 }

```

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.

```

843 \catcode'\Q=3

```

`\hyxmp@trimspaces\x` redefines `\x` to have the same replacement text sans leading and trailing space tokens.

```

844 \newcommand{\hyxmp@trimspaces}[1]{%

```

Use grouping to emulate a multi-token `afterassignment` queue.

```

845 \begingroup

```

Put “`\toks 0 {`” into the `afterassignment` queue.

```

846 \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.

```

847 \expandafter\hyxmp@trimb\expandafter\noexpand#1Q Q}%

```

Transfer the trimmed text back into #1.

```
848 \edef#1{\the\toks0}%
849 }
```

`\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.

```
850 \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.

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

```
853 \newif\ifhyxmp@unicodetex
854 \ifnum64='\^^^0040\relax
855 \hyxmp@unicodetextrue
856 \else
857 \hyxmp@unicodetexfalse
858 \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.

```
859 \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.

```
860 \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@text` but with all occurrences of “<” replaced with `<`; all occurrences of “>” replaced with `>`; and all occurrences of “&” replaced with `&`;

```
861 \newcommand*{\hyxmp@xmlify}[1]{%
862 \gdef\hyxmp@xmlified{}
```


Escaped PDF string → PDFDocEncoding/Unicode

```
863 \EdefUnescapeString\hyxmp@text{#1}%  
864 \ifhyxmp@unicodetex
```

PDFDocEncoding/Unicode → UTF-32BE

```
865 \hyxmp@is@unicode\hyxmp@text{%  
866 \StringEncodingConvert  
867 \hyxmp@text\hyxmp@text{utf16be}{utf32be}%  
868 }{%  
869 \ifXeTeX  
870 \hyxmp@xetex@crap  
871 \else  
872 \StringEncodingConvert  
873 \hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%  
874 \fi  
875 }%
```

UTF-32BE → UTF-32BE as hex string

```
876 \EdefEscapeHex\hyxmp@text{\hyxmp@text}%
```

UTF-32BE → XML in ASCII

```
877 \edef\hyxmp@text{%  
878 \expandafter  
879 }\expandafter\hyxmp@toxml@unicodetex\hyxmp@text  
880 \relax\relax\relax\relax\relax\relax\relax\relax  
881 \else
```

PDFDocEncoding/Unicode → UTF-8

```
882 \hyxmp@is@unicode\hyxmp@text{%  
883 \StringEncodingConvert  
884 \hyxmp@text\hyxmp@text{utf16be}{utf8}%  
885 }{%  
886 \StringEncodingConvert  
887 \hyxmp@text\hyxmp@text{pdfdoc}{utf8}%  
888 }%
```

UTF-8 → UTF-8 as hex string

```
889 \EdefEscapeHex\hyxmp@text{\hyxmp@text}%
```

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

```
890 \edef\hyxmp@text{%  
891 \expandafter\hyxmp@toxml\hyxmp@text\@empty\@empty  
892 }%
```

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

```
893 \EdefUnescapeHex\hyxmp@text{\hyxmp@text}%  
894 \fi  
895 \global\let\hyxmp@xmlified\hyxmp@text  
896 }
```

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

```
897 \begingroup  
898 \lccode'\<=254 %  
899 \lccode'\>=255 %  
900 \catcode254=12 %
```

```

901 \catcode255=12 %
902 \lowercase{\endgroup
903 \def\hyxmp@is@unicode#1{%
904 \expandafter\hyxmp@@is@unicode#1<>\@nil
905 }%
906 \def\hyxmp@@is@unicode#1<>#2\@nil{%
907 \ifx\#1\%
908 \expandafter\@firstoftwo
909 \else
910 \expandafter\@secondoftwo
911 \fi
912 }%
913 }

```

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

```

914 \def\hyxmp@toxml#1#2{%
915 \ifx#1\@empty
916 \else
917 \ifnum"#1#2='\& %
918 26616D703B% &
919 \else\ifnum"#1#2='\< %
920 266C743B% &lt;
921 \else\ifnum"#1#2='\> %
922 2667743B% &gt;
923 \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.

```

924 \@ifundefined{pdfmark}{%
925 #1#2%
926 }{%
927 \ifnum"#1#2='\( %
928 5C28% \(
929 \else\ifnum"#1#2='\) %
930 5C29% \)
931 \else
932 #1#2%
933 \fi\fi
934 }%
935 \fi\fi\fi
936 \expandafter\hyxmp@toxml
937 \fi
938 }

```

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

```

939 \def\hyxmp@toxml@unicodetex#1#2#3#4#5#6#7#8{%
940   \ifx#1\relax
941   \else
942     \ifnum"#1#2#3#4#5#6#7#8>127 %
943       \uccode'\*="#1#2#3#4#5#6#7#8\relax
944       \uppercase{%
945         \edef\hyxmp@text{\hyxmp@text *}%
946       }%
947     \else\ifnum"#7#8='< %
948       \edef\hyxmp@text{\hyxmp@text &lt;}%
949     \else\ifnum"#7#8='& %
950       \edef\hyxmp@text{\hyxmp@text &amp;}%
951     \else\ifnum"#7#8='> %
952       \edef\hyxmp@text{\hyxmp@text &gt;}%
953     \else\ifnum"#7#8='\ %
954       \edef\hyxmp@text{\hyxmp@text\space}%
955     \else
956       \uccode'\*="#7#8\relax
957       \uppercase{%
958         \edef\hyxmp@text{\hyxmp@text *}%
959       }%
960     \fi\fi\fi\fi\fi
961     \expandafter\hyxmp@toxml@unicodetex
962   \fi
963 }

```

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

```

964 \def\hyxmp@skipzeros#1{%
965   \ifx#10%
966     \expandafter\hyxmp@skipzeros
967   \fi
968 }

```

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

```

\hyxmp@xetex@crap
\hyxmp@try
\hyxmp@crap@result
\hyxmp@text
969 \begingroup
970 \def\x#1{\endgroup
971   \def\hyxmp@xetex@crap{%
972     \edef\hyxmp@try{%
973       \expandafter\hyxmp@SpaceOther\hyxmp@text#1\@nil
974     }%
975     \let\hyxmp@crap@result=N%
976     \expandafter\hyxmp@crap@test\hyxmp@try\relax
977     \ifx\hyxmp@crap@result Y%
978       \let\hyxmp@text\@empty
979     \expandafter\hyxmp@crap@convert\hyxmp@try\relax
980   \else
981     \StringEncodingConvert\hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
982   \fi
983 }%
984 }
985 \x{ }

```

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

```
986 \begingroup
987 \catcode'\~ =12 %
988 \lccode'\~ ='\ %
989 \lowercase{\endgroup
990 \def\hyxmp@SpaceOther#1 #2\@nil{%
991   #1%
992   \ifx\relax#2\relax
993     \expandafter\@gobble
994   \else
995     ~%
996     \expandafter\@firstofone
997   \fi
998   {\hyxmp@SpaceOther#2\@nil}%
999 }%
1000 }
```

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

```
1001 \def\hyxmp@crap@test#1{%
1002   \ifx#1\relax
1003   \else
1004     \ifnum'#1>127 %
1005       \let\hyxmp@crap@result=Y%
1006       \expandafter\expandafter\expandafter\hyxmp@skiptorelax
1007     \else
1008       \expandafter\expandafter\expandafter\hyxmp@crap@test
1009     \fi
1010   \fi
1011 }
```

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

```
1012 \def\hyxmp@skiptorelax#1\relax{}
```

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

```
\hyxmp@num 1013 \def\hyxmp@crap@convert#1{%
\hyxmp@text 1014   \ifx#1\relax
1015   \else
1016     \edef\hyxmp@num{\number'#1}%
1017     \ifnum\hyxmp@num>"FFFFFF" %
1018       \lccode'\! =\intcalDiv{\hyxmp@num}{\number"1000000}\relax
1019       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
1020     \edef\hyxmp@num{\intcalMod{\hyxmp@num}{\number"1000000}}%
1021   \else
1022     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
1023   \fi
1024   \ifnum\hyxmp@num>"FFFF" %
1025     \lccode'\! =\intcalDiv{\hyxmp@num}{\number"10000}\relax
1026     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
1027   \edef\hyxmp@num{\intcalMod{\hyxmp@num}{\number"10000}}%
1028   \else
1029     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
1030   \fi
1031   \ifnum\hyxmp@num>"FF" %
1032     \lccode'\! =\intcalDiv{\hyxmp@num}{\number"100}\relax
```

```

1033     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
1034     \edef\hyxmp@num{\intcalMod{\hyxmp@num}{\number"100}}%
1035     \else
1036     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
1037     \fi
1038     \ifnum\hyxmp@num>0 %
1039     \lccode'\!=\hyxmp@num\relax
1040     \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
1041     \else
1042     \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
1043     \fi
1044     \expandafter\hyxmp@crap@convert
1045     \fi
1046 }

```

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

```

1047 \begingroup
1048   \catcode0=12 %
1049   \gdef\hyxmp@zero{^^00}%
1050 \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.

```

1051 \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.

```

1052 \newcommand*{\hyxmp@add@simple}[2]{%
1053   \@ifnotmtargexp{#2}{%
1054     \hyxmp@xmlify{#2}%
1055     \hyxmp@add@to+xml{\hyxmp@extra@indent_____<}%
1056     \xdef\hyxmp+xml{\hyxmp+xml#1}%
1057     \hyxmp@add@to+xml{>\hyxmp@xmlified</}%
1058     \xdef\hyxmp+xml{\hyxmp+xml#1>^^J}%
1059   }%
1060 }

```

`\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.

```

1061 \newcommand*{\hyxmp@add@simple@var}[2]{%
1062   \expandafter\ifx\csname#2\endcsname\relax

```

```

1063 \else
1064   \hyxmp@xmlify{\csname#2\endcsname}%
1065   \hyxmp@add@to@xml{%
1066     \hyxmp@extra@indent_____<#1>\hyxmp@xmlied</#1>^^J%
1067   }%
1068 \fi
1069 }

```

`\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.

```

1070 \newcommand*{\hyxmp@add@simple@lang}[2]{%
1071   \@ifnotmtarg{#2}{%
1072     \hyxmp@xmlify{#2}%
1073     \expandafter\hyxmp@add@simple@lang@i\hyxmp@xmlied\relax{#1}%
1074   }%
1075 }

```

`\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.

```

1076 \newcommand*{\hyxmp@add@simple@lang@i}{%
1077   \@ifnextchar[\hyxmp@add@simple@lang@i]{\hyxmp@add@simple@lang@i[\pdfmetalang]}%
1078 }

```

`\hyxmp@add@simple@lang@ii` 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.

```

1079 \def\hyxmp@add@simple@lang@ii[#1]#2\relax#3{%
1080   \@ifnotmtarg{#2}{%
1081     \hyxmp@xmlify{#2}%
1082     \@ifmtarg{#1}{%
1083       \hyxmp@add@to@xml{%
1084         _____<#3>\hyxmp@xmlied</#3>^^J%
1085       }%
1086     }{%
1087       \hyxmp@add@to@xml{%
1088         _____<#3 xml:lang="#1">\hyxmp@xmlied</#3>^^J%
1089       }%
1090     }%
1091   }%
1092 }

```

`\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.

```

1093 \newcommand*{\hyxmp@add@simple@pfx}[3]{%
1094   \@ifnotmtargexp{#3}{%
1095     \hyxmp@add@to@xml{\hyxmp@extra@indent_____<}%
1096     \xdef\hyxmp@xml{\hyxmp@xml#1}%

```

```

1097 \hyxmp@pdfstringdef\hyxmp@iprefix{#2}%
1098 \hyxmp@xmlify{\hyxmp@iprefix}%
1099 \hyxmp@add@to@xml{>\hyxmp@xmlified}%
1100 \hyxmp@xmlify{#3}%
1101 \hyxmp@add@to@xml{\hyxmp@xmlified</}%
1102 \xdef\hyxmp@xml{\hyxmp@xml#1>^^J}%
1103 }%
1104 }

```

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}`”.

```

1105 \def\hyxmp@alt@title{}
1106 \def\hyxmp@alt@description{}
1107 \def\hyxmp@alt@rights{}

```

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

```

1108 \newcommand{\hyxmp@LA@accept}[2]{%
1109 \define@key{hyxmp@LA}{#1}{%

```

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

```

1110 \hyxmp@pdfstringdef\hyxmp@value{##1}%
1111 \xdef#2{#2\noexpand\do {\hyxmp@cur@lang} {\hyxmp@value}}%
1112 }
1113 }

```

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

```

1114 \hyxmp@LA@accept{pdftitle}{\hyxmp@alt@title}
1115 \hyxmp@LA@accept{pdfsubject}{\hyxmp@alt@description}
1116 \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.

```

1117 \newcommand{\XMPLangAlt}[2]{%
1118 \let\do=\relax

```

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

```
1119 \edef\hyxmp@cur@lang{#1}%
1120 \setkeys{hyxmp@LA}{#2}%
1121 }
```

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

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

```
1122 \def\hyxmp@modulo@a#1{%
1123   \@tempcntb=\@hyxmp@count
1124   \divide\@tempcntb by #1
1125   \multiply\@tempcntb by #1
1126   \advance\@hyxmp@count by -\@tempcntb
1127 }
```

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

```
\hyxmp@big@prime@ii 1128 \def\hyxmp@big@prime{536870923}
1129 \def\hyxmp@big@prime@ii{536870027}
```

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

```
\hyxmp@one@token 1130 \def\hyxmp@seed@rng#1{%
1131   \@hyxmp@count=\hyxmp@big@prime
1132   \futurelet\hyxmp@one@token\hyxmp@seed@rng@i#1\@empty
1133 }
```

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

```
\next 1134 \def\hyxmp@seed@rng@i{%
1135   \ifx\hyxmp@one@token\@empty
1136     \let\next=\relax
1137   \else
1138     \def\next##1{%
1139       \multiply\@hyxmp@count by 3
1140       \advance\@hyxmp@count by '##1
1141       \hyxmp@modulo@a{\hyxmp@big@prime}%
1142       \futurelet\hyxmp@one@token\hyxmp@seed@rng@i
1143     }%
1144   \fi
1145   \next
1146 }
```

`\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 `\@hyxmp@count` and `\@tempcntb` are overwritten in the process.

```
1147 \def\hyxmp@set@rand@num{%
```



```

1148 \@hyxmp@count=\hyxmp@rand@num
1149 \multiply\@hyxmp@count by 3
1150 \advance\@hyxmp@count by \hyxmp@big@prime@ii
1151 \hyxmp@modulo@a{\hyxmp@big@prime}%
1152 \xdef\hyxmp@rand@num{\the\@hyxmp@count}%
1153 }

```

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

```

1154 \def\hyxmp@append@hex#1{%
1155   \hyxmp@set@rand@num
1156   \@hyxmp@count=\hyxmp@rand@num
1157   \hyxmp@modulo@a{16}%
1158   \ifnum\@hyxmp@count<10
1159     \xdef#1{#1\the\@hyxmp@count}%
1160   \else

```

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

```

1161     \advance\@hyxmp@count by -10
1162     \ifcase\@hyxmp@count
1163       \xdef#1{#1a}%
1164       \or\xdef#1{#1b}%
1165       \or\xdef#1{#1c}%
1166       \or\xdef#1{#1d}%
1167       \or\xdef#1{#1e}%
1168       \or\xdef#1{#1f}%
1169     \fi
1170 \fi
1171 }

```

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

```

1172 \def\hyxmp@append@hex@iii#1{%
1173   \hyxmp@append@hex#1%
1174   \hyxmp@append@hex#1%
1175   \hyxmp@append@hex#1%
1176 }

```

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

```

1177 \def\hyxmp@append@hex@iv#1{%
1178   \hyxmp@append@hex@iii#1%
1179   \hyxmp@append@hex#1%
1180 }

```

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

```

1181 \def\hyxmp@create@uuid#1{%
1182   \def#1{uuid:}%
1183   \hyxmp@append@hex@iv#1%
1184   \hyxmp@append@hex@iv#1%
1185   \g@addto@macro#1{-}%
1186   \hyxmp@append@hex@iv#1%

```

```

1187 \g@addto@macro#1{-4}%
1188 \hyxmp@append@hex@iii#1%
1189 \g@addto@macro#1{-}%
Randomly select one of “8”, “9”, “a”, or “b”.
1190 \hyxmp@set@rand@num
1191 \@hyxmp@count=\hyxmp@rand@num
1192 \hyxmp@modulo@a{4}%
1193 \ifcase\@hyxmp@count
1194 \g@addto@macro#1{8}%
1195 \or\g@addto@macro#1{9}%
1196 \or\g@addto@macro#1{a}%
1197 \or\g@addto@macro#1{b}%
1198 \fi
1199 \hyxmp@append@hex@iii#1%
1200 \g@addto@macro#1{-}%
1201 \hyxmp@append@hex@iv#1%
1202 \hyxmp@append@hex@iv#1%
1203 \hyxmp@append@hex@iv#1%
1204 }

```

`\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@seed@string` `\hyxmp@DocumentID` as a random UUID.

```

1205 \newcommand*{\hyxmp@def@DocumentID}{%
1206 \edef\hyxmp@seed@string{\hyxmp@jobname:\@pdftitle:\@pdfauthor:}%
1207 \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
1208 \edef\hyxmp@rand@num{\the\@hyxmp@count}%
1209 \hyxmp@create@uuid\hyxmp@DocumentID
1210 }

```

`\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@seed@string` `\hyxmp@create@uuid` to define `\hyxmp@InstanceID` as a random UUID. For the current timestamp, we use both the document-specified timestamp from `pdfdate` and the `TEX` time. The former can be more precise (to sub-seconds) but may be less random (as it depends on manual document modifications) while the latter is typically less precise (to minutes) but may be more random (as it is updated automatically).

```

1211 \newcommand*{\hyxmp@def@InstanceID}{%
1212 \hyxmp@today@xmp@define{\hyxmp@seed@string}%
1213 \edef\hyxmp@seed@string{%
1214 \hyxmp@jobname:\@pdftitle:\@pdfauthor:\hyxmp@today@xmp:\hyxmp@seed@string
1215 }%
1216 \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
1217 \edef\hyxmp@rand@num{\the\@hyxmp@count}%
1218 \hyxmp@create@uuid\hyxmp@InstanceID
1219 }

```

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.

```

1220 \newcommand*{\hyxmp@add@to+xml}[1]{%
1221   \bgroup
1222     \hyxmp@count=0
1223     \ifhyxmp@unicodetex
1224       \@tempcntb=65536%
1225     \else
1226       \@tempcntb=256%
1227     \fi
1228     \loop
1229       \lccode\@hyxmp@count=\@hyxmp@count
1230       \advance\@hyxmp@count by 1
1231       \ifnum\@hyxmp@count<\@tempcntb
1232     \repeat
1233     \lccode\@_='\ \relax
1234     \lccode\^^C='\,\relax
1235     \lccode\^^U='\_\relax
1236     \lowercase{\xdef\hyxmp@new+xml{#1}}%
1237     \xdef\hyxmp+xml{\hyxmp+xml\hyxmp@new+xml}%
1238   \egroup
1239 }
```

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

```

1240 \bgroup
1241 \catcode\#=11
1242 \gdef\hyxmp@hash{#}
1243 \egroup
```

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

```

1244 \bgroup
1245 \xdef\hyxmp+xml{}%
1246 \hyxmp@add@to+xml{%
1247 -----^^J%
1248 }
```

```

1249 \xdef\hyxmp@padding{\hyxmp@xml}%
1250 \egroup
1251 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
1252 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
1253 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
1254 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
1255 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}

```

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

```

1256 \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 `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.

```

1257 \newcommand*{\hyxmp@define@pdfproducer}{%
1258   \gdef\@pdfproducer{TeX}
1259   \ifLuaTeX
1260     \expandafter\hyxmp@banner@to@producer\expandafter{\luatexbanner}%
1261   \else
1262     \ifPDFTeX
1263       \expandafter\hyxmp@banner@to@producer\expandafter{\pdfptexbanner}%
1264     \else
1265       \ifXeTeX
1266         \edef\@pdfproducer{XeTeX version \the\XeTeXversion\XeTeXrevision}%
1267       \fi
1268     \fi
1269   \fi
1270 }

```

`\@pdfproducer` Define `\@pdfproducer` as the `TEX` engine’s banner string (e.g., “This is `\hyxmp@banner@to@producer` LuaHBTeX, Version 1.15.0 (TeX Live 2022)”), removing the initial “This is” if possible (specifically, when `ε-TEX`’s `\scantokens` primitive is available).

```

1271 \def\hyxmp@banner@to@producer#1{%
1272   \ifx\scantokens\relax
1273     \gdef\@pdfproducer{#1}%
1274   \else
1275     {\scantokens{\makeatletter\hyxmp@remove@this#1\relax}}%
1276   \fi
1277 }

```

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

```

1278 \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.

```
1279 \AtBeginDocument{%
1280   \ifx\pdfproducer\relax
1281     \hyxmp@define@pdfproducer
1282   \fi
1283 }
```

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

`\hyxmp@major@minor`

```
1284 \newcommand*{\hyxmp@assign@major@minor}{%
1285   \@ifundefined{pdfvariable}{%
1286     \@ifundefined{pdfminorversion}{%
Case 1: Neither \pdfvariable nor \pdfminorversion is defined (XYLATEX
and regular LATEX).
1287     }{%
Case 2: \pdfminorversion is defined (pdfLATEX and pre-0.85 LuaLATEX).
1288     \xdef\hyxmp@major@minor{\the\pdfminorversion}%
1289     \@ifundefined{pdfmajorversion}{%
Case 2(a): \pdfmajorversion is not defined (older versions of pdfLATEX and
LuaLATEX).
1290     \xdef\hyxmp@major@minor{1.\hyxmp@major@minor}%
1291     }{%
Case 2(b): \pdfmajorversion is defined (pdfLATEX 1.40.21+).
1292     \xdef\hyxmp@major@minor{\the\pdfmajorversion.\hyxmp@major@minor}%
1293     }%
1294     }%
1295     }{%
Case 3: \pdfvariable is defined (LuaLATEX 0.85+).
1296     \xdef\hyxmp@major@minor{\the\pdfvariable majorversion.\the\pdfvariable minorversion}%
1297     }%
1298 }
```

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

```
1299 \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.

```
1300 \hyxmp@add@simple@var{pdf:Producer}{@pdfproducer}%
1301 \hyxmp@add@simple@var{pdf:Keywords}{@pdfkeywords}%
```

```

1302 \hyxmp@add@simple{pdf:Trapped}{\@pdftrapped}%
1303 \hyxmp@assign@major@minor
1304 \hyxmp@add@simple@var{pdf:PDFVersion}{hyxmp@major@minor}%
1305 }

```

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.

```

1306 \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.

```

1307 \newcommand*{\hyxmp@rdf@dc}[3][\iffalse]{%

```

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

```

1308 \@ifmtargexp{#3}{\@tempswafalse}{\@tempswatru}%
1309 #1
1310 \@tempswatru
1311 \fi

```

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

```

1312 \if@tempswa
1313 \hyxmp@xmlify{#3}%

```

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

```

1314 \let\hyxmp@value=\hyxmp@xmlified
1315 \hyxmp@add@to@xml{%
1316 _____<dc:#2>^^J%
1317 _____<rdf:Alt>^^J%
1318 }%

```

Record whether property #2 has definitions in multiple languages.

```

1319 \@if@def@and@nonempty{hyxmp@alt@#2}{%
1320 \hyxmp@multi@langstrue
1321 }{%
1322 \hyxmp@multi@langsfalse
1323 }%

```

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.

```

1324 \ifx\@pdfmetalang\hyxmp@x@default
1325 \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.

```

1326     \PackageError{hyperxmp}%
1327     {\string\XMPLangAlt\space was used without specifying
1328     pdfmetalang\MessageBreak
1329     or pdflang}%
1330     {Be sure to assign a language code to the pdfmetalang key and/or a
1331     document\MessageBreak
1332     language to the pdflang key (e.g., \string\hypersetup{pdfmetalang={en}}).}%
1333     \else

```

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

```

1334     \hyxmp@add@to@xml{%
1335     -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@value</rdf:li>^^J%
1336     }%
1337     \fi
1338     \else
1339     \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.

```

1340     \hyxmp@xmlify{\@pdfmetalang}%
1341     \hyxmp@add@to@xml{%
1342     -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@value</rdf:li>^^J%
1343     -----<rdf:li xml:lang="\hyxmp@xmlified">\hyxmp@value</rdf:li>^^J%
1344     }%
1345     \def\do##1##2{
1346     \hyxmp@xmlify{##2}%
1347     \hyxmp@add@to@xml{%
1348     -----<rdf:li xml:lang="##1">\hyxmp@xmlified</rdf:li>^^J%
1349     }%
1350     }%
1351     \csname hyxmp@alt@#2\endcsname
1352     \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.

```

1353     \hyxmp@xmlify{\@pdfmetalang}%
1354     \hyxmp@add@to@xml{%
1355     -----<rdf:li xml:lang="\hyxmp@xmlified">\hyxmp@value</rdf:li>^^J%
1356     }%
1357     \fi
1358     \fi

```

Complete this XMP element.

```

1359     \hyxmp@add@to@xml{%
1360     -----</rdf:Alt>^^J%
1361     -----</dc:#2>^^J%
1362     }%
1363     \fi
1364 }%

```

`\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.

```
1365 \newcommand*\hyxmp@list@to@xml}[4][\iffalse]{%
```

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

```
1366 \ifmtargexp{#4}{\@tempwafalse}{\@tempwattrue}%
```

```
1367 #1
```

```
1368 \@tempwattrue
```

```
1369 \fi
```

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

```
1370 \if@tempswa
```

```
1371 \hyxmp@add@to@xml{%
```

```
1372 -----<dc:#2>^^J%
```

```
1373 -----<rdf:#3>^^J%
```

```
1374 }%
```

```
1375 \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`.

```
1376 \hyxmp@xmlify{#4}%
```

```
1377 \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
```

```
1378 \def\@elt##1{%
```

```
1379 \hyxmp@add@to@xml{%
```

```
1380 -----<rdf:li>##1</rdf:li>^^J%
```

```
1381 }%
```

```
1382 }%
```

```
1383 \hyxmp@list
```

```
1384 \egroup
```

```
1385 \hyxmp@add@to@xml{%
```

```
1386 -----</rdf:#3>^^J%
```

```
1387 -----</dc:#2>^^J%
```

```
1388 }%
```

```
1389 \fi
```

```
1390 }
```

`\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.

```
1391 \newcommand{\hyxmp@singleton@dc}[3][Bag]{%
```

```
1392 \ifnotmtargexp{#3}{%
```

```
1393 \hyxmp@xmlify{#3}%
```

```
1394 \hyxmp@add@to@xml{%
```

```
1395 -----<dc:#2>^^J%
```

```
1396 -----<rdf:#1>^^J%
```

```
1397 -----<rdf:li>\hyxmp@xmlified</rdf:li>^^J%
```

```
1398 -----</rdf:#1>^^J%
```

```
1399 -----</dc:#2>^^J%
```

```
1400 }%
```

```
1401 }
```

```
1402 }
```


`\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).

```

1403 \newcommand*{\hyxmp@cond@dc@identifier}[2]{%
1404   \ifx\hyxmp@xmlified\@empty
1405     \ifnotmtargexp{#2}{%
1406       \hyxmp@add@simple@pfx{dc:identifier}{#1}{#2}%
1407     }%
1408   \fi
1409 }
```

`\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^AT_EX—unless the author specified `pdfdate`, in which case we use that.

```

1410 \newcommand*{\hyxmp@dc@schema}{%
1411   \hyxmp@add@simple{dc:format}{application/pdf}%
1412   \hyxmp@rdf@dc[\ifHy@pdfa]{title}{\@pdftitle}%
1413   \hyxmp@rdf@dc[\ifHy@pdfa]{description}{\@pdfsubject}%
1414   \hyxmp@rdf@dc{rights}{\@pdfcopyright}%
1415   \hyxmp@singleton@dc{publisher}{\@pdfpublisher}%
1416   \@ifmtargexp{\@pdfdatetime}{%
1417     \hyxmp@singleton@dc[Seq]{date}{\hyxmp@today@xmp}%
1418   }{%
1419     \hyxmp@singleton@dc[Seq]{date}{\@pdfdatetime}%
1420   }%
1421   \hyxmp@singleton@dc{type}{\@pdftype}%
1422   \hyxmp@list@to@xml[\ifHy@pdfa]{creator}{Seq}{\hyxmp@pdfauthor}%
1423   \hyxmp@list@to@xml{subject}{Bag}{\hyxmp@pdfkeywords}%
1424   \ifx\@pdfsource\@empty
1425   \else
1426     \hyxmp@add@simple{dc:source}{\@pdfsource}%
1427   \fi
1428   \hyxmp@list@to@xml{language}{Bag}{\hyxmp@dc@lang}%
1429 % If |\@pdfidentifier| is empty, try setting it to each of |\@pdfdoi|,
1430 % |\@pdfeissn|, |\@pdfissn|, and |\@pdfisbn|, in turn, with proper
1431 % syntactic adjustments.
1432 %   \begin{macrocode}
1433   \@ifmtargexp{\@pdfidentifier}{%
1434     \let\hyxmp@xmlified=\@empty
1435     \hyxmp@cond@dc@identifier{info:doi/}{\@pdfdoi}%
1436     \hyxmp@cond@dc@identifier{urn:ISSN:}{\@pdfeissn}%
1437     \hyxmp@cond@dc@identifier{urn:ISSN:}{\@pdfissn}%
1438     \hyxmp@cond@dc@identifier{urn:ISBN:}{\@pdfisbn}%
1439   }{%
1440     \hyxmp@add@simple{dc:identifier}{\@pdfidentifier}%

```

```

1441 }%
1442 }

```

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.

```

1443 \newcommand*{\hyxmp@xmpRights@schema}{%

```

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

```

1444 \let\hyxmp@rights=\@empty
1445 \ifx\@pdflicenseurl\@empty
1446 \else
1447 \def\hyxmp@rights{YES}%
1448 \fi
1449 \ifx\@pdfcopyright\@empty
1450 \else
1451 \def\hyxmp@rights{YES}%
1452 \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.

```

1453 \ifx\hyxmp@rights\@empty
1454 \else
1455 \ifx\@pdfcopyright\@empty
1456 \else
1457 \hyxmp@add@simple{xmpRights:Marked}{True}%
1458 \fi
1459 \hyxmp@add@simple{xmpRights:WebStatement}{\@pdflicenseurl}%
1460 \fi
1461 }

```

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.

```

1462 \expandafter\hyxmp@aep@toks\expandafter=\expandafter{%
1463 \the\hyxmp@aep@toks
1464 \AtEndPreamble{%
1465 \@ifmtargexp{\hyxmp@DocumentID}{\hyxmp@def@DocumentID}{}%
1466 \@ifmtargexp{\hyxmp@InstanceID}{\hyxmp@def@InstanceID}{}%
1467 }%
1468 }

```

`\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_EX-based workflow. We additionally support the `xmpMM:VersionID` property, whose value is supplied by the author using `pdfversionid`.

```

1469 \gdef\hyxmp@mm@schema{%
1470   \hyxmp@add@simple{xmpMM:DocumentID}{\hyxmp@DocumentID}%
1471   \hyxmp@add@simple{xmpMM:InstanceID}{\hyxmp@InstanceID}%
1472   \hyxmp@add@simple{xmpMM:VersionID}{\pdfversionid}%
1473   \hyxmp@add@simple{xmpMM:RenditionClass}{\pdfrendition}%
1474 }

```

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`.

```

1475 \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`.

```

1476   \@ifmtargexp{\pdfcreationdate}{%
1477     \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@today@xmp}%
1478   }{%
1479     \hyxmp@add@simple{xmp:CreateDate}{%
1480       \expandafter\hyxmp@as@xmp@date\expandafter{\pdfcreationdate}}%
1481   }%

```

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

```

1482   \@ifmtargexp{\pdfmoddate}{%
1483     \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@today@xmp}%
1484   }{%
1485     \hyxmp@add@simple{xmp:ModifyDate}{%
1486       \expandafter\hyxmp@as@xmp@date\expandafter{\pdfmoddate}}%
1487   }%

```

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

```

1488   \@ifmtargexp{\pdfmetadatetime}{%
1489     \hyxmp@add@simple{xmp:MetadataDate}{\hyxmp@today@xmp}%
1490   }{%
1491     \hyxmp@add@simple{xmp:MetadataDate}{\pdfmetadatetime}%
1492   }%

```

Define the creation tool and the base URL.

```

1493   \hyxmp@add@simple{xmp:CreatorTool}{\pdfcreator}%
1494   \hyxmp@add@simple{xmp:BaseURL}{\@baseurl}%
1495 }

```

3.6.7 The Photoshop schema

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

```
1496 \gdef\hyxmp@photoshop@schema{%
1497   \edef\hyxmp@photoshop@data{\@pdfauthor\@pdfcaptionwriter}%
1498   \hyxmp@add@simple{photoshop:AuthorsPosition}{\@pdfauthor}%
1499   \hyxmp@add@simple{photoshop:CaptionWriter}{\@pdfcaptionwriter}%
1500 }
```

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`.

```
1501 \newcommand*\hyxmp@pdfa@id@schema{%
1502   \ifHy@pdfa
1503     \hyxmp@add@simple{pdfaid:part}{\@pdfapart}%
1504     \hyxmp@add@simple{pdfaid:conformance}{\@pdfaconformance}%
1505   \fi
1506 }
```

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

```
1507 \newcommand*\hyxmp@pdfua@id@schema{%
1508   \hyxmp@add@simple{pdfuaid:part}{\@pdfuapart}%
1509 }
```

`\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.

```
1510 \newcommand*\hyxmp@pdfx@id@schema{%
1511   \@hyxmp@count=0\hyxmp@pdfx@major\relax
1512   \ifnum\@hyxmp@count=0
1513     \else
1514       \ifnum\@hyxmp@count=1
1515         \hyxmp@add@simple{pdfx:GTS_PDFXVersion}{PDF/X-1:2001}%
1516         \hyxmp@add@simple{pdfx:GTS_PDFXConformance}{\@pdfxstandard}%
1517       \else
1518         \ifnum\@hyxmp@count<4
1519           \hyxmp@add@simple{pdfx:GTS_PDFXVersion}{\@pdfxstandard}%
1520         \else
1521           \hyxmp@add@simple{pdfxid:GTS_PDFXVersion}{\@pdfxstandard}%
1522         \fi
1523       \fi
1524     \fi
1525 }
```

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.

```
1526 \begingroup
1527 \catcode'\&=12
1528 \catcode'\#=12
1529 \gdef\xmplinesep{&#xA;}
1530 \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.

```
1531 \newcommand*\hyxmp@list@to@lines}[2]{%
1532 \ifnotmtargexp{#2}{%
1533 \bgroup
1534 \hyxmp@add@to+xml{%
1535 \hyxmp@extra@indent_____<#1>%
1536 }%
```

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

```
1537 \def\@elt@first##1{%
1538 \hyxmp@add@to+xml{##1}%
1539 \let\@elt=\@elt@rest
1540 }%
```

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

```
1541 \def\@elt@rest##1{%
1542 \hyxmp@add@to+xml{\xmplinesep##1}%
1543 }%
```

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

```
1544 \let\@elt=\@elt@first
1545 \hyxmp@xmlify{#2}%
1546 \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
1547 \hyxmp@list
1548 \hyxmp@add@to+xml{</#1>^^J}%
1549 \egroup
1550 }%
1551 }
```

`\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.

```
1552 \gdef\hyxmp@iptc@schema{%
```

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

```
1553 \ifx\hyxmp@iptc@data\@empty
1554 \else
```

```

1555     \hyxmp@add@to@xml{%
1556     -----<Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">^^J%
1557     }%

```

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

```

1558     \bgroup
1559     \edef\hyxmp@extra@indent{\hyxmp@extra@indent\space\space}%
1560     \hyxmp@list@to@lines{Iptc4xmpCore:AdrExtadr}{\@pdfcontactaddress}%
1561     \hyxmp@add@simple{Iptc4xmpCore:AdrCity}{\@pdfcontactcity}%
1562     \hyxmp@add@simple{Iptc4xmpCore:AdrRegion}{\@pdfcontactregion}%
1563     \hyxmp@add@simple{Iptc4xmpCore:AdrPcode}{\@pdfcontactpostcode}%
1564     \hyxmp@add@simple{Iptc4xmpCore:AdrCtry}{\@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.

```

1565     \def\xmplinesep{,}%
1566     \hyxmp@list@to@lines{Iptc4xmpCore:TelWork}{\@pdfcontactphone}%
1567     \hyxmp@list@to@lines{Iptc4xmpCore:EmailWork}{\@pdfcontactemail}%
1568     \hyxmp@list@to@lines{Iptc4xmpCore:UrlWork}{\@pdfcontacturl}%
1569     \egroup
1570     \hyxmp@add@to@xml{%
1571     -----</Iptc4xmpCore:CreatorContactInfo>^^J%
1572     }%
1573     \fi
1574 }

```

3.6.10 The PRISM Basic Metadata schema

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

```

1575 \newcommand*{\hyxmp@prism@schema}{%
1576 \ifx\hyxmp@prism@data\@empty
1577 \else
1578     \hyxmp@add@simple{prism:complianceProfile}{three}%
1579 \fi
1580 \hyxmp@add@simple@lang{prism:subtitle}{\@pdfsubtitle}%
1581 \hyxmp@add@simple@lang{prism:publicationName}{\@pdfpublication}%
1582 \hyxmp@add@simple{prism:aggregationType}{\@pdfpubtype}%
1583 \hyxmp@add@simple@lang{prism:bookEdition}{\@pdfbookedition}%
1584 \hyxmp@add@simple{prism:volume}{\@pdfvolumenum}%
1585 \hyxmp@add@simple{prism:number}{\@pdfissuenum}%
1586 \hyxmp@add@simple{prism:pageRange}{\@pdfpagerange}%
1587 \hyxmp@add@simple{prism:isbn}{\@pdfisbn}%
1588 \hyxmp@add@simple{prism:issn}{\@pdfissn}%
1589 \hyxmp@add@simple{prism:eIssn}{\@pdfeissn}%
1590 \hyxmp@add@simple{prism:doi}{\@pdfdoi}%
1591 \hyxmp@add@simple{prism:url}{\@pdfurl}%
1592 \hyxmp@add@simple{prism:byteCount}{\@pdfbytes}%

```

```

1593 \hyxmp@add@simple{prism:pageCount}{\@pdfnumpages}%
1594 }

```

3.6.11 The Journal Article Versions (JAV) schema

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

```

1595 \newcommand*{\hyxmp@jav@schema}{%
1596 \hyxmp@add@simple{jav:journal_article_version}{\@pdfpubstatus}%
1597 }

```

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.

```

1598 \newcommand*{\hyxmp@xmptpg@schema}{%
1599 \ifLuaTeX
1600 \luadirect{write_xmp_font_list(\the\hyxmp@cct)}%
1601 \fi
1602 \hyxmp@add@simple{xmpTPg:NPages}{\@pdfnumpages}%
1603 }

```

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

```

1604 \ifLuaTeX
1605 \newcatcodetable\hyxmp@cct
1606 \savecatcodetable\hyxmp@cct
1607 \fi

```

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

```

1608 \bgroup
1609 \catcode'\_ =11
1610 \gdef\hyxmp@prot@us{_%}
1611 \egroup

```

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

```

1612 \ifLuaTeX
1613 \begin{luacode*}
1614 function write_xmp_font_list (cct)
1615     local function show_field(name, ...)
1616         for i = 1, select("#", ...) do
1617             local val = select(i, ...)
1618             if val then
1619                 local xml = string.gsub(val, "&", "&amp;")
1620                 xml = string.gsub(xml, "<", "&lt;")
1621                 xml = string.gsub(xml, ">", "&gt;")
1622                 xml = string.gsub(xml, "_", "\\hyxmp@prot@us ")

```

```

1623         tex.print(cct, "_____<stFnt:" .. name .. ">" ..
1624             xml .. "</stFnt:" .. name .. ">^^J%")
1625     return
1626 end
1627 end
1628 end
1629 tex.print(cct, "\\hyxmp@add@to@xml{%")
1630 tex.print(cct, "_____<xmpTPg:Fonts>^^J%")
1631 tex.print(cct, "_____<rdf:Bag>^^J%")
1632 for i, f in font.each() do
1633     tex.print(cct, "_____<rdf:li rdf:parseType=\"Resource\">^^J%")
1634     if f.filename then
1635         local fname = string.gsub(f.filename, "^harfloaded:(.*)", "%1")
1636         local info = fontloader.info(fname)
1637         if info then
1638             show_field("fontFace", info.fullname)
1639             show_field("fontFamily", info.familyname)
1640             show_field("fontName", info.fontname)
1641             show_field("versionString", info.version)
1642         end
1643         local baseName = string.gsub(f.filename, ".*[/\\](.*)", "%1")
1644         show_field("fontFileName", baseName)
1645     else
1646         show_field("fontName", f.psname, f.fullname, f.name)
1647     end
1648     if f.format and f.format ~= "unknown" then
1649         show_field("fontType", f.format)
1650     end
1651     tex.print(cct, "_____</rdf:li>^^J%")
1652 end
1653 tex.print(cct, "_____</rdf:Bag>^^J%")
1654 tex.print(cct, "_____</xmpTPg:Fonts>^^J%")
1655 tex.print(cct, "}")
1656 end
1657 \end{luacode*}
1658 \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.

```

1659 \newcommand*{\hyxmp@check@iptc@data}{%
1660 \edef\hyxmp@iptc@data{%
1661 \@pdfcontactaddress
1662 \@pdfcontactcity
1663 \@pdfcontactregion
1664 \@pdfcontactpostcode

```



```

1665 \pdfcontactcountry
1666 \pdfcontactphone
1667 \pdfcontactemail
1668 \pdfcontacturl
1669 }%
1670 }%

```

`\hyxmp@check@prism@data` Define `\hyxmp@prism@data` as the concatenation of all PRISM metadata supplied by the document.

```

1671 \newcommand*{\hyxmp@check@prism@data}{%
1672 \edef\hyxmp@prism@data{%
1673 \pdfbookedition
1674 \pdfbytes
1675 \pdfdoi
1676 \pdfeissn
1677 \pdfisbn
1678 \pdfissn
1679 \pdfissuenumber
1680 \pdfnumpages
1681 \pdfpagerange
1682 \pdfpublication
1683 \pdfpubtype
1684 \pdfsubtitle
1685 \pdfurl
1686 \pdfvolumenum
1687 }%
1688 }%

```

`\hyxmp@check@jav@data` Define `\hyxmp@jav@data` as the concatenation of all JAV metadata supplied by the document.

```

1689 \newcommand*{\hyxmp@check@jav@data}{%
1690 \edef\hyxmp@jav@data{%
1691 \pdfpubstatus
1692 }%
1693 }

```

`\hyxmp@begin@extension@decls` Begin a block of XML tags that indicates we're declaring one or more extension schemata.

```

1694 \newcommand*{\hyxmp@begin@extension@decls}{%
1695 \hyxmp@add@to+xml{%
1696 -----<pdfaExtension:schemas>^^J%
1697 -----<rdf:Bag>^^J%
1698 }%
1699 }

```

`\hyxmp@end@extension@decls` End the block of XML tags begun by `\hyxmp@begin@extension@decls`.

```

1700 \newcommand*{\hyxmp@end@extension@decls}{%
1701 \hyxmp@add@to+xml{%
1702 -----</rdf:Bag>^^J%
1703 -----</pdfaExtension:schemas>^^J%
1704 }%
1705 }

```

`\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.

```
1706 \newcommand*{\hyxmp@begin@ext@decl}[3]{%
1707   \hyxmp@add@to@xml{%
1708     <rdf:li rdf:parseType="Resource">^^J%
1709     <pdfaSchema:schema>#1</pdfaSchema:schema>^^J%
1710     <pdfaSchema:prefix>#2</pdfaSchema:prefix>^^J%
1711     <pdfaSchema:namespaceURI>#3</pdfaSchema:namespaceURI>^^J%
1712     <pdfaSchema:property>^^J%
1713     <rdf:Seq>^^J%
1714   }%
1715 }
```

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

```
1716 \newcommand*{\hyxmp@end@ext@decl}{%
1717   \hyxmp@add@to@xml{%
1718     </rdf:Seq>^^J%
1719     </pdfaSchema:property>^^J%
1720     </rdf:li>^^J%
1721   }%
1722 }
```

`\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.

```
1723 \newcommand{\hyxmp@declare@property}[4][Text]{%
1724   \hyxmp@add@to@xml{%
1725     <rdf:li rdf:parseType="Resource">^^J%
1726     <pdfaProperty:name>}%
1727   \xdef\hyxmp@xml{\hyxmp@xml#2}%
1728   \hyxmp@add@to@xml{</pdfaProperty:name>^^J%
1729     <pdfaProperty:valueType>#1</pdfaProperty:valueType>^^J%
1730     <pdfaProperty:category>#3</pdfaProperty:category>^^J%
1731     <pdfaProperty:description>#4</pdfaProperty:description>^^J%
1732     </rdf:li>^^J%
1733   }%
1734 }
```

`\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.

```
1735 \newcommand{\hyxmp@declare@field}[3][Text]{%
1736   \hyxmp@add@to@xml{%
1737     <rdf:li rdf:parseType="Resource">^^J%
1738     <pdfaField:name>#2</pdfaField:name>^^J%
1739     <pdfaField:valueType>#1</pdfaField:valueType>^^J%
1740     <pdfaField:description>#3</pdfaField:description>^^J%
1741     </rdf:li>^^J%
1742   }%
1743 }
```

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

```
1744 \newcommand*{\hyxmp@pdf@extensions}{%
```

```

1745 \hyxmp@begin@ext@decl
1746     {Adobe PDF Schema}%
1747     {pdf}%
1748     {http://ns.adobe.com/pdf/1.3/}%
1749 \hyxmp@declare@property
1750     {Trapped}%
1751     {internal}%
1752     {Indication if the document has been modified to include trapping information}%
1753 \hyxmp@end@ext@decl
1754 }%

```

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

```

1755 \newcommand*{\hyxmp@mm@extensions}{%
1756 \hyxmp@begin@ext@decl
1757     {XMP Media Management Schema}%
1758     {xmpMM}%
1759     {http://ns.adobe.com/xap/1.0/mm/}%
1760 \hyxmp@declare@property
1761     [URI]
1762     {DocumentID}%
1763     {internal}%
1764     {UUID based identifier for all versions and renditions of a document}%
1765 \hyxmp@declare@property
1766     [URI]
1767     {InstanceID}%
1768     {internal}%
1769     {UUID based identifier for specific incarnation of a document}%
1770 \hyxmp@declare@property
1771     {VersionID}%
1772     {internal}%
1773     {Document version identifier}%
1774 \hyxmp@declare@property
1775     [RenditionClass]%
1776     {RenditionClass}%
1777     {internal}%
1778     {The manner in which a document is rendered}%
1779 \hyxmp@end@ext@decl
1780 }%

```

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

```

1781 \newcommand*{\hyxmp@pdfa@id@extensions}{%
1782 \hyxmp@begin@ext@decl
1783     {PDF/A Identification Schema}%
1784     {pdfaid}%
1785     {http://www.aiim.org/pdfa/ns/id/}%
1786 \hyxmp@declare@property
1787     [Integer]%
1788     {part}%
1789     {internal}%
1790     {Part of PDF/A standard}%
1791 \hyxmp@declare@property
1792     {conformance}%
1793     {internal}%
1794     {Conformance level of PDF/A standard}%

```

```
1795 \hyxmp@end@ext@decl
1796 }%
```

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

```
1797 \newcommand*{\hyxmp@pdfua@id@extensions}{%
1798   \hyxmp@begin@ext@decl
1799     {PDF/UA Universal Accessibility Schema}%
1800     {pdfuaid}%
1801     {http://www.aiim.org/pdfua/ns/id/}%
1802   \hyxmp@declare@property
1803     [Integer]%
1804     {part}%
1805     {internal}%
1806     {Part of ISO 14289 standard}%
1807   \hyxmp@end@ext@decl
1808 }%
```

`\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.

```
1809 \newcommand*{\hyxmp@pdfx@id@extensions}{%
1810   \ifx\hyxmp@pdfx@major\empty
1811   \else
1812     \hyxmp@begin@ext@decl
1813       {Adobe Document Info PDF/X eXtension Schema}%
1814       {pdfx}%
1815       {http://ns.adobe.com/pdfx/1.3/}%
1816     \hyxmp@declare@property
1817       {GTS_PDFXVersion}%
1818       {internal}%
1819       {ID of PDF/X standard}%
1820     \hyxmp@declare@property
1821       {GTS_PDFXConformance}%
1822       {internal}%
1823       {Conformance level of PDF/X standard}%
1824     \hyxmp@end@ext@decl
1825   \fi
```

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

```
1826 \@hyxmp@count=0\hyxmp@pdfx@major\relax
1827 \ifnum\@hyxmp@count>3
1828   \hyxmp@begin@ext@decl
1829     {PDF/X ID Schema}%
1830     {pdfxid}%
1831     {http://www.npes.org/pdfx/ns/id/}%
1832   \hyxmp@declare@property
1833     {GTS_PDFXVersion}%
1834     {internal}%
1835     {ID of PDF/X standard}%
1836   \hyxmp@end@ext@decl
1837 \fi
1838 }%
```

`\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.

```
1839 \newcommand*{\hyxmp@iptc@extensions}{%
1840   \hyxmp@begin@ext@decl
1841     {IPTC Core Schema}%
1842     {Iptc4xmpCore}%
1843     {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}%
1844   \hyxmp@declare@property
1845     [ContactInfo]
1846     {CreatorContactInfo}
1847     {external}
1848     {Document creator's contact information}
```

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

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

```

1888 \hyxmp@add@to+xml{%
1889 -----</rdf:Seq>^^J%
1890 -----</pdfaType:field>^^J%
1891 -----</rdf:li>^^J%
1892 -----</rdf:Seq>^^J%
1893 -----</pdfaSchema:valueType>^^J%
1894 -----</rdf:li>^^J%
1895 }%
1896 }

```

\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.

```

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

```

```

1937     {ISSN for the electronic publication in which the document was published}%
1938 \hyxmp@declare@property
1939     {isbn}%
1940     {external}%
1941     {ISBN for the publication in which the document was published}%
1942 \hyxmp@declare@property
1943     {doi}%
1944     {external}%
1945     {Digital Object Identifier for the document}%
1946 \hyxmp@declare@property
1947     [URL]
1948     {url}%
1949     {external}%
1950     {URL at which the document can be found}%
1951 \hyxmp@declare@property
1952     [Integer]
1953     {byteCount}%
1954     {internal}%
1955     {Approximate file size in octets}%
1956 \hyxmp@declare@property
1957     [Integer]
1958     {pageCount}%
1959     {internal}%
1960     {Number of pages in the print version of the document}%
1961 \hyxmp@declare@property
1962     {subtitle}%
1963     {external}%
1964     {Document's subtitle}%
1965 \hyxmp@end@ext@decl
1966 }%

```

`\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.

```

1967 \newcommand*{\hyxmp@jav@extensions}{%
1968   \hyxmp@begin@ext@decl
1969     {NISO/ALPSP Journal Article Versions}%
1970     {jav}%
1971     {http://www.niso.org/schemas/jav/1.0/}%
1972   \hyxmp@declare@property
1973     [Closed Choice of Text]%
1974     {journal_article_version}%
1975     {external}%
1976     {Article status, one of "A0", "SMUR", "AM", "P", "VoR", "CVoR", or "EVoR"}%
1977   \hyxmp@end@ext@decl
1978 }%

```

`\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.

```

1979 \newcommand*{\hyxmp@declare@extensions}{%
1980   \hyxmp@begin@extension@decls

```

Declare the Adobe PDF schema (always present).

```

1981 \hyxmp@pdf@extensions
Declare the XMP Media Management extensions (always present).
1982 \hyxmp@mm@extensions
Declare the PDF/A Identification extensions, but only when generating a PDF/A
document.
1983 \ifHy@pdfa
1984 \hyxmp@pdfa@id@extensions
1985 \fi
Conditionally declare the PDF/UA Universal Accessibility extensions.
1986 \ifx\@pdfuapart\@empty
1987 \else
1988 \hyxmp@pdfua@id@extensions
1989 \fi

```

`\next` Conditionally declare the PDF/X extensions.

```

1990 \ifx\@pdfxversion\@empty
1991 \else
1992 \hyxmp@pdfx@id@extensions
1993 \fi

```

Conditionally declare IPTC photo metadata extensions.

```

1994 \ifx\hyxmp@iptc@data\@empty
1995 \else
1996 \hyxmp@iptc@extensions
1997 \fi

```

Conditionally declare PRISM basic metadata extensions.

```

1998 \ifx\hyxmp@prism@data\@empty
1999 \else
2000 \hyxmp@prism@extensions
2001 \fi

```

Conditionally declare JAV metadata.

```

2002 \ifx\hyxmp@jav@data\@empty
2003 \else
2004 \hyxmp@jav@extensions
2005 \fi

```

```

2006 \hyxmp@end@extension@decls
2007 }

```

3.6.14 Combining schemata into an XMP packet

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

```

2008 \begingroup
2009 \ifhyxmp@unicodetex
2010 \lccode\!="FEFF %
2011 \lowercase{%
2012 \gdef\hyxmp@bom{!}
2013 }%
2014 \else
2015 \catcode\^^ef=12
2016 \catcode\^^bb=12

```



```

2017 \catcode'\^^bf=12
2018 \gdef\hyxmp@bom{^^ef^^bb^^bf}%
2019 \fi
2020 \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.

```

2021 \def\hyxmp@construct@packet{%
2022 \gdef\hyxmp+xml{%
2023 \hyxmp@add@to+xml{<?xpacket begin="\hyxmp@bom" %
2024 id="W5M0MpCehiHzreSzNTczkc9d"?>^^J%
2025 <x:xmpmeta xmlns:x="adobe:ns:meta/">^^J%
2026 __<rdf:RDF %
2027 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\hyxmp@hash">^^J%
2028 ____<rdf:Description rdf:about="">^^J%

```

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

```

2029 _____xmlns:pdf="http://ns.adobe.com/pdf/1.3/"^^J%
2030 _____xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/"^^J%
2031 _____xmlns:dc="http://purl.org/dc/elements/1.1/"^^J%
2032 _____xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/"^^J%
2033 _____xmlns:xmp="http://ns.adobe.com/xap/1.0/"^^J%
2034 _____xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"^^J%
2035 _____xmlns:stEvt="http://ns.adobe.com/xap/1.0/sType/ResourceEvent\hyxmp@hash"^^J%
2036 _____xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/"^^J%
2037 _____xmlns:pdfuaid="http://www.aiim.org/pdfua/ns/id/"^^J%
2038 _____xmlns:pdfx="http://ns.adobe.com/pdfx/1.3/"^^J%
2039 }%

```

We make one exception to the rule of including every namespace we can potentially use: We don't define the pdfxid namespace unless the PDF/X version (specified by the pdfxstandard) option is 4 or greater. Otherwise, Adobe Acrobat—at least Adobe Acrobat DC 2020—alters the way it displays color. (I believe it renders color in a printer gamut instead of a screen gamut.)

```

2040 \ifnum\hyxmp@pdfx@major>3
2041 \hyxmp@add@to+xml{%
2042 _____xmlns:pdfxid="http://www.npes.org/pdfx/ns/id/"^^J%
2043 }%
2044 \fi

```

Revert to “include every namespace” mode.

```

2045 \hyxmp@add@to+xml{%
2046 _____xmlns:prism="http://prismstandard.org/namespaces/basic/3.0/"^^J%
2047 _____xmlns:jav="http://www.niso.org/schemas/jav/1.0/"^^J%
2048 _____xmlns:xmpTPg="http://ns.adobe.com/xap/1.0/t/pg/"^^J%
2049 _____xmlns:stFnt="http://ns.adobe.com/xap/1.0/sType/Font\hyxmp@hash"^^J%
2050 _____xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"^^J%
2051 _____xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"^^J%
2052 _____xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema\hyxmp@hash"^^J%
2053 _____xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property\hyxmp@hash"^^J%
2054 _____xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type\hyxmp@hash"^^J%
2055 _____xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field\hyxmp@hash">^^J%
2056 }%

```

Declare non-standard schemata.

```
2057 \hyxmp@check@iptc@data
2058 \hyxmp@check@prism@data
2059 \hyxmp@check@jav@data
2060 \hyxmp@declare@extensions
```

Insert all the metadata we know how to insert.

```
2061 \hyxmp@pdf@schema
2062 \hyxmp@xmpRights@schema
2063 \hyxmp@dc@schema
2064 \hyxmp@photoshop@schema
2065 \hyxmp@xmp@basic@schema
2066 \hyxmp@pdfa@id@schema
2067 \hyxmp@pdfua@id@schema
2068 \hyxmp@pdfx@id@schema
2069 \hyxmp@mm@schema
2070 \hyxmp@iptc@schema
2071 \hyxmp@prism@schema
2072 \hyxmp@jav@schema
2073 \hyxmp@xmptpg@schema
2074 \hyxmp@add@to+xml{%
2075 ----</rdf:Description>^^J%
2076 __</rdf:RDF>^^J%
2077 </x:xmpmeta>^^J%
2078 \hyxmp@padding
2079 <?xpacket end="w"?>^^J%
2080 }%
2081 }
```

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
\hyxmp@driver function.
2082 \newcommand*{\hyxmp@embed@packet}{%
2083 \hyxmp@construct@packet
2084 \def\hyxmp@driver{hpdfTeX}%
2085 \ifx\hyxmp@driver\Hy@driver
2086 \hyxmp@embed@packet@pdfTeX
2087 \else
2088 \def\hyxmp@driver{hLaTeX}%
2089 \ifx\hyxmp@driver\Hy@driver
2090 \hyxmp@embed@packet@LaTeX
2091 \else
2092 \def\hyxmp@driver{hdvipdfm}%
2093 \ifx\hyxmp@driver\Hy@driver
2094 \hyxmp@embed@packet@dvipdfm
2095 \else
2096 \def\hyxmp@driver{hXeTeX}%
2097 \ifx\hyxmp@driver\Hy@driver
2098 \hyxmp@embed@packet@XeTeX
```

```

2099     \else
2100     \@ifundefined{pdfmark}{%
2101     \PackageWarningNoLine{hyperxmp}{%
2102     Unrecognized hyperref driver ‘\Hy@driver’.\MessageBreak
2103     \hyxmp@jobname.tex’s XMP metadata will *not* be\MessageBreak
2104     embedded in the resulting file}%
2105     }{%
2106     \hyxmp@embed@packet@pdfmark
2107     }%
2108     \fi
2109 \fi
2110 \fi
2111 \fi
2112 }

```

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.

```
2113 \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.

```

2114 \newcommand*{\hyxmp@embed@packet@pdftex}{%
2115 \bgroup
2116 \ifluatex
2117 \else
2118 \pdfcompresslevel=0
2119 \fi
2120 \immediate\pdfobj \ifluatex uncompressed\fi stream attr {%
2121 /Type /Metadata
2122 /Subtype /XML
2123 }{\hyxmp@xml}%
2124 \pdfcatalog {/Metadata \the\pdflastobj\space 0 R}%
2125 \egroup
2126 }

```

3.7.2 Embedding using LuaTeX 0.85+

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

```

2127 \newcommand*{\hyxmp@embed@packet@luatex}{%
2128 \immediate\pdfextension obj uncompressed stream attr {%
2129     /Type /Metadata
2130     /Subtype /XML
2131 }{\hyxmp@xml}%
2132 \pdfextension catalog {/Metadata \the\numexpr\pdffeedback lastobj\relax\space 0 R}%
2133 }

```

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`, `dvipdfone`, `dvips`, `dviwindo`, `nativepdf`, `pdfmark`, `ps2pdf`, `textures`, and `vtexpdfmark` options to `hyperref`, but I've tested only a few of those.

```

2134 \newcommand*{\hyxmp@embed@packet@pdfmark}{%
2135 \pdfmark{%
2136     pdfmark=/NamespacePush
2137 }%
2138 \pdfmark{%
2139     pdfmark=/OBJ,
2140     Raw={/_objdef \string{hyxmp@Metadata\string} /type /stream}%
2141 }%
2142 \pdfmark{%
2143     pdfmark=/PUT,
2144     Raw={\string{hyxmp@Metadata\string}
2145         2 dict begin
2146             /Type /Metadata def
2147             /Subtype /XML def
2148             currentdict
2149         end
2150     }%
2151 }%
2152 \pdfmark{%
2153     pdfmark=/PUT,
2154     Raw={\string{hyxmp@Metadata\string} (\hyxmp@xml)}%
2155 }%
2156 \pdfmark{%
2157     pdfmark=/Metadata,
2158     Raw={\string{Catalog\string} \string{hyxmp@Metadata\string}}%
2159 }%
2160 \pdfmark{%
2161     pdfmark=/NamespacePop
2162 }%
2163 }

```

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.

```

2164 \newcommand*{\hyxmp@embed@packet@dvipdfm}{%
2165 \hyxmp@string@len{\hyxmp@xml}%
2166 \special{pdf: object @hyxmp@Metadata
2167     <<

```

```

2168     /Type /Metadata
2169     /Subtype /XML
2170     /Length \the\@hyxmp@count
2171   >>
2172   stream^^J\hyxmp+xml endstream%
2173 }%
2174 \special{pdf: docview
2175   <<
2176     /Metadata @hyxmp@Metadata
2177   >>
2178 }%
2179 }

```

`\hyxmp@string@len` Set `\@hyxmp@count` 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.

```

2180 \newcommand*\hyxmp@string@len}[1]{%
2181   \@hyxmp@count=0
2182   \expandafter\hyxmp@count@spaces#1 {} %
2183   \expandafter\hyxmp@count@non@spaces#1{}}%
2184 }

```

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

```

2185 \def\hyxmp@count@spaces#1 {%
2186   \def\hyxmp@one@token{#1}%
2187   \ifx\hyxmp@one@token\empty
2188     \advance\@hyxmp@count by -1
2189   \else
2190     \advance\@hyxmp@count by 1
2191     \expandafter\hyxmp@count@spaces
2192   \fi
2193 }

```

`\hyxmp@count@non@spaces` Count the number of non-spaces in a given string. Ideally, we'd count both spaces and non-spaces but `TEX` won't bind `#1` to a space character (category code 10). Hence, in each iteration, `#1` is bound to the next non-space character only.

```

2194 \newcommand*\hyxmp@count@non@spaces}[1]{%
2195   \def\hyxmp@one@token{#1}%
2196   \ifx\hyxmp@one@token\empty
2197   \else
2198     \advance\@hyxmp@count by 1
2199     \expandafter\hyxmp@count@non@spaces
2200   \fi
2201 }

```

3.7.5 Embedding using `XYTEX`

`\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.

```
2202 \newcommand*{\hyxmp@embed@packet@xetex}{%
2203   \special{pdf:stream @hyxmp@Metadata (\hyxmp@xml)
2204     <<
2205       /Type /Metadata
2206       /Subtype /XML
2207     >>
2208   }%
2209   \special{pdf:put @catalog
2210     <<
2211       /Metadata @hyxmp@Metadata
2212     >>
2213   }%
2214 }
```

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.

```
2215 \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.

```
2216 \catcode'\="\hyxmp@dq@code
```

4 Help Wanted

Comma handling Ideally, `\xmpquote` should automatically replace all commas with `\xmpcomma`. Unfortunately, my $\text{T}_{\text{E}}\text{X}$ 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 $\text{T}_{\text{E}}\text{X}$ engines (pdf $\text{T}_{\text{E}}\text{X}$, Lua $\text{T}_{\text{E}}\text{X}$, X $\text{T}_{\text{E}}\text{X}$, 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 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ document presented on pages 9–11. 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: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: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.
- [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.
- [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.
- [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.
- [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: Pre-defined XMP Properties in PDF/A-1*, March 20, 2008. Available from http://www.pdfa.org/wp-content/uploads/2011/08/tn0008_predefined_xmp_properties_in_pdfa-1_2008-03-20.pdf.
- [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.
- [15] Misha Wolf and Charles Wicksteed. Date and time formats. Note NOTE-datettime, World Wide Web Consortium (W3C), September 15, 1997. Available from <http://www.w3.org/TR/NOTE-datettime>.

Change History

v1.0 General: Initial version 1	regardless of the specified metadata language 62
v1.1 \hyxmp@construct@packet: Explicitly set the category codes of characters $\langle EF \rangle$, $\langle BB \rangle$, and $\langle BF \rangle$ to “letter”. Thanks to Daniel Schömer for the bug report 81	\hyxmp@xmpRights@schema: Renamed the xapRights namespace prefix to xmpRights 66
v1.2 General: Added support for the X _Y TeX backend (xdvipdfmx) . . . 1 Added support for the Photoshop schema 1 Made the package compatible with ngerman. Thanks to Tobias Mueller for the bug report. 18	v1.5 General: Made the XMP inclusion more robust. Thanks to Heiko Oberdiek for the bug report and suggested modifications. . . 18
v1.3 General: Introduced the pdfmetalang package option, which enables an author to specify the language in which he wrote the document’s metadata 32	v2.0 General: Added support for the XMP Basic schema and miscellaneous other bits of metadata 1 Heiko Oberdiek’s major rewrite of the code to better support native-Unicode T _E X implementations (X _Y TeX and LuaT _E X) 1 New \AtBeginDocument code from Heiko Oberdiek to properly encode \@pdfmetalang 32
v1.4 \hyxmp@mm@schema: Renamed the xapMM namespace prefix to xmpMM 67 \hyxmp@rdf@dc: Included metadata in the x-default language	\hyxmp@add@simple: Added this macro 53 \hyxmp@add@to+xml: Updated also to replace commas 59 \hyxmp@bom: Added by Heiko Oberdiek 80

<code>\hyxmp@comma</code> : Added this macro	42	<code>\ProcessKeyvalOptions</code> : Added this macro	29
<code>\hyxmp@construct@packet</code> : Modified by Heiko Oberdiek to use an appropriate BOM representation via <code>\hyxmp@bom</code>	81	<code>\xmpcomma</code> : Added this macro	41
<code>\hyxmp@crap@convert</code> : Added by Heiko Oberdiek	52	<code>\xmpquote</code> : Added this macro	42
<code>\hyxmp@crap@test</code> : Added by Heiko Oberdiek	52	<code>\XMPTruncateList</code> : Added this macro	42
<code>\hyxmp@dc@schema</code> : Added support for <code>dc:language</code> and <code>dc:source</code>	65	v2.1 General: Enabled <code>hyperxmp</code> and <code>hyperref</code> to be loaded in either order. This addresses a bug report by Yury Donskoy	27
<code>\hyxmp@is@unicode</code> : Added by Heiko Oberdiek	49	<code>\hypersetup</code> : Added this macro	29
<code>\hyxmp@list@to@xml</code> : Modified by Heiko Oberdiek to use the new Unicode-processing macros	64	<code>\hyxmp@hypersetup</code> : Added this macro	29
<code>\hyxmp@photoshop@schema</code> : Simplified using <code>\hyxmp@add@simple</code>	68	<code>\hyxmp@redefine@Hyp</code> : Added this macro	28
<code>\hyxmp@ProcessKeyvalOptions</code> : Added this macro	29	v2.2 General: Added support for the IPTC Photo Metadata schema	1
<code>\hyxmp@skiptorelax</code> : Added by Heiko Oberdiek	52	<code>\hyxmp@iptc@extensions</code> : Added this macro to support PDF/A generation	76
<code>\hyxmp@skipzeros</code> : Added by Heiko Oberdiek	51	<code>\hyxmp@iptc@schema</code> : Added this macro	69
<code>\hyxmp@SpaceOther</code> : Added by Heiko Oberdiek	52	<code>\hyxmp@list@to@lines</code> : Added this macro	69
<code>\hyxmp@toxml</code> : Added by Heiko Oberdiek	50	<code>\xmpcomma</code> : Changed the default from <code>\relax</code> to an ordinary comma	41
Escaped parentheses written with <code>pdfmarks</code> to prevent <code>dvips</code> from line-wrapping the XMP packet	50	<code>\xmplinesep</code> : Added this macro	69
<code>\hyxmp@toxml@unicodetex</code> : Added by Heiko Oberdiek	51	v2.3 <code>\hyxmp@iptc@extensions</code> : Gave the <code>Iptc4xmpCore:CreatorContactInfo</code> fields a unique <code>pdfaType:prefix</code> to better support conversion of the document to PDF/A	76
<code>\hyxmp@xetex@crap</code> : Added by Heiko Oberdiek	51	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>	40
<code>\hyxmp@xmlify</code> : Completely rewritten by Heiko Oberdiek to better support Unicode-enabled <code>TeX</code> programs	48	v2.3b <code>\XMPTruncateList</code> : Made all definitions local to avoid spurious <code>Too many</code> <code>unprocessed floats errors</code> when running with <code>memoir</code>	42
<code>\hyxmp@xmp@basic@schema</code> : Added this macro	67	v2.4 General: Added support for the PDF/A Identification schema, as requested by Florian Breitwieser	1
<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	66		
<code>\hyxmp@zero</code> : Added by Heiko Oberdiek	53		
<code>\ifhyxmp@unicodetex</code> : Added by Heiko Oberdiek	48		

<code>\hyxmp@add@simple@var</code> : Added this macro	53	the document date (and optionally time)	1
<code>\hyxmp@create@uuid</code> : Modified this macro to produce a proper version 4 (random or pseudorandom) UUID	57		
<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	65	<code>\hyxmp@auto@assign@data</code> : Automatically use <code>\title</code> and <code>\author</code> if <code>pdftitle</code> and <code>pdfauthor</code> are left unspecified. Thanks to Maciej Radziejewski for the suggestion	34
<code>\hyxmp@parse@time</code> : Added this macro	43		
<code>\hyxmp@parse@tz</code> : Added this macro	44		
<code>\hyxmp@parse@tz@char</code> : Added this macro	43		
<code>\hyxmp@pdf@schema</code> : Made <code>\hyxmp@pdf@schema</code> <i>always</i> include the Adobe PDF schema, even when empty. Florian Breitwieser noted that this is necessary for PDF/A-1b compliance	61	<code>\hyxmp@add@to+xml</code> : Corrected inadvertent lowercasing of non-Latin characters when run under \XeTeX or \LuaTeX (bug reported by Leonid Sinev)	59
<code>\hyxmp@pdf@to@xmp@date</code> : Added this macro	43		
<code>\hyxmp@pdfa@id@schema</code> : Added this macro	68		
<code>\hyxmp@today@xmp</code> : Modified the code to parse the time and timezone from <code>\pdfcreationdate</code> , as proposed by Florian Breitwieser	46	v2.9 General: Force inclusion of <code>dc:creator</code> , <code>dc:title</code> , and <code>dc:description</code> —even if empty—when <code>hyperref</code> is loaded with the <code>pdfa</code> option (suggested by Leonid Sinev)	1
<code>\hyxmp@today@xmp@define</code> : Added this macro	46	Introduced the <code>pdftype</code> package option, which enables an author to specify the type of document being produced	1
<code>\hyxmp@xmp@to@pdf@date</code> : Added this macro	44	<code>\hyxmp@iptc@schema</code> : Use <code>lptc4xmpCore</code> instead of <code>lptc4ContInfo</code> as the contact-information metadata prefix. Leonid Sinev reports that Acrobat’s PDF/A validator seems to prefer <code>lptc4xmpCore</code>	69
<code>\xmp tilde</code> : Added this macro	42	<code>\hyxmp@pdfa@id@schema</code> : Let the author specify the PDF/A part and conformance IDs, as requested by Leonid Sinev	68
v2.5 General: Enabled “_” to work within email addresses, as requested by Leonid Sinev	1		
<code>\hyxmp@add@to+xml</code> : Updated also to replace underscores and to modify only the text being added, not the already-modified text	59		
<code>\hyxmp@textunderscore</code> : Added this macro	20		
<code>\hyxmp@uscore</code> : Added this macro	42		
v2.6 General: Added support for a new <code>pdfdate</code> key to explicitly specify		v3.0 General: Made the code compatible with \LuaTeX 0.85. Thanks to Robert Schlicht, Leonid Sinev, and David Carlisle for bug reports and to Leonid Sinev for helping test the new <code>hyperxmp</code> code	1
		<code>\hyxmp@embed@packet@luatex</code> : Added this macro	83
		<code>\hyxmp@today@xmp@define</code> : Modified to accept the name of a macro to define	46
		<code>\hyxmp@xmp@basic@schema</code> : Made the XMP <code>xmp:CreateDate</code> , <code>xmp:ModifyDate</code> , and	

	xmp:MetadataDate match the PDF CreationDate	67		spaces. Bug reported by Gaëtan Leurent	58
v3.1	<code>\hyxmp@embed@packet@luatex:</code> Updated to use <code>\pdfextensionobj uncompressed</code> as suggested by Hans Hagen	83	v3.5	<code>\hyxmp@DocumentID:</code> Added the <code>pdfdocumentid</code> option, at Michael Osipov's request	23
	<code>\hyxmp@embed@packet@pdftex:</code> Leave the XMP packet—and only the XMP packet—uncompressed in both pdf \TeX and pre-0.85 Lua \TeX	83		<code>\hyxmp@InstanceID:</code> Added the <code>pdfinstanceid</code> option, at Michael Osipov's request	24
v3.2	<code>\hyxmp@as@pdf@date:</code> Added this macro	44		<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 <code>pdfdocumentid</code> and <code>pdfinstanceid</code> options	67
	<code>\hyxmp@as@xmp@date:</code> Added this macro	43		<code>\hyxmp@seed@string:</code> Seed with the \TeX timestamp in addition to the document-specified timestamp	58
	<code>\hyxmp@today@xmp@define:</code> Modified to include hours and minutes	46	v4.0	General: Include all metadata within a single <code>rdf:Description</code> block	1
	<code>\hyxmp@xmp@basic@schema:</code> Honor <code>hyperref</code> 's <code>pdfcreationdate</code> and <code>pdfmoddate</code> options plus a new <code>pdfmetadate</code> option. Leonid Sinev requested this additional control and helped test the resulting <code>hyperxmp</code> code	67		<code>\hyxmp@add@simple@lang:</code> Added this macro	54
v3.3	General: Don't overwrite an existing <code>pdfmetalang</code> with <code>pdflang</code> or <code>x-default</code> . This addresses a bug report by Niklas Beisert	32		<code>\hyxmp@begin@ext@decl:</code> Added this macro	74
	<code>\@pdfsource:</code> Added this macro and the corresponding <code>pdfsource</code> option, at Niklas Beisert's request	23		<code>\hyxmp@declare@field:</code> Replaced <code>\hyxmp@declare@resource</code> with this macro	74
	<code>\hyxmp@rdf@dc:</code> Bug fix: Output the metadata language as correct XML even when <code>hyperref</code> is loaded with the <code>unicode</code> option	62		<code>\hyxmp@declare@property:</code> Added this macro	74
	<code>\XMPLangAlt:</code> Added this macro based on a request—and some code—by Niklas Beisert to support metadata expressed in multiple languages	55		<code>\hyxmp@end@ext@decl:</code> Added this macro	74
v3.4	General: Use <code>ifmtarg</code> to test for empty arguments, including non-empty but all spaces	1		<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>	76
	<code>\hyxmp@seed@string:</code> Correctly handle an author field of all			Rewrote to more closely honor the XMP specification	76
				<code>\hyxmp@iptc@schema:</code> Moved the definition of <code>\hyxmp@iptc@data</code> from here into <code>\hyxmp@check@iptc@data</code>	69
				Renamed this macro to <code>\hyxmp@iptc@schema</code> from <code>\hyxmp@photometa@schema</code>	69
				Rewrote this macro entirely to correct the use of fields within a structure	69

<code>\hyxmp@mm@extensions</code> : Added this macro	75	<code>\hyxmp@no@info@lists</code> : Renamed this macros from	
<code>\hyxmp@mm@schema</code> : Include xmpMM:VersionID in the XMP packet	67	<code>\hyxmp@suppress@pdf@metadata</code> and rewrote it to replace, if possible, only Author and Keywords	27
<code>\hyxmp@no@info@lists</code> : Added this macro	27	<code>\hyxmp@pdf@extensions</code> : Added this macro	74
<code>\hyxmp@pdfa@id@extensions</code> : Added this macro	75	<code>\hyxmp@pdf@schema</code> : Honor pdftrapped	61
<code>\hyxmp@prism@extensions</code> : Added this macro	78	<code>\hyxmp@pdfua@id@extensions</code> : Added this macro	76
<code>\hyxmp@prism@schema</code> : Added this macro	70	<code>\hyxmp@pdfua@id@schema</code> : Added this macro	68
<code>\XMPTruncateList</code> : Deprecated this macro	42	<code>\hyxmp@pdfx@id@extensions</code> : Added this macro	76
v4.1		<code>\hyxmp@pdfx@id@schema</code> : Added this macro	68
General: Updated the documentation to refer to <code>\pdfnumpages</code> by its correct name. Thanks to Volker RW Schaa for catching the discrepancy	1	<code>\hyxmp@today@pdf</code> : Added this macro	47
<code>\hyxmp@aep@toks</code> : Invoke <code>\hyxmp@no@info@lists</code> at the beginning of the document, for compatibility with both newer and older versions of hyperref	32	<code>\hyxmp@today@xmp</code> : Support X _Y TeX's <code>\filemoddate</code>	46
<code>\hyxmp@singleton@dc</code> : Added this macro	64	<code>\hyxmp@today@xmp@define</code> : Modified to specify UTC	46
v5.0		v5.1	
General: Added support for PDF/UA standards, as requested by Robin Schwab	1	<code>\hyxmp@banner@to@producer</code> : Prevent the category code of “@” from propagating past the <code>\begin{document}</code> . Thanks to Robert Schlicht for noticing this catcode “leak” and providing a correction	60
Added support for PDF/X standards, as requested by Robin Schwab	1	<code>\hyxmp@define@pdfproducer</code> : Check for LuaTeX before checking for pdfTeX to work around luatex85's confusing <code>iftex</code> by defining <code>\pdfTeXversion</code> . Thanks to Robin Schwab for the bug report	60
Define a default producer	61	<code>\hyxmp@timestamp</code> : Don't rely on <code>\jobname.aux</code> existing to query the current time under X _Y TeX. Instead, use <code>\jobname.log</code> . Thanks to Ulrike Fischer for the bug report and for her suggestion to use the log file.	47
Don't set any document dates (creation, modification, or metadata) from <code>pdfdate</code>	1	v5.2	
<code>\@pdfrendition</code> : Added the <code>pdfrendition</code> option	24	General: Introduced the <code>pdfidentifier</code> package option, which enables an author to specify a unique identifier for the document	1
<code>\@pdfxstandard</code> : Added this macro	23		
<code>\hyxmp@add@simple</code> : Insert the tag name (#1) verbatim	53		
<code>\hyxmp@check@standards</code> : Added this macro	31		
<code>\hyxmp@check@std</code> : Added this macro	23		
<code>\hyxmp@declare@property</code> : Insert the property name (#2) verbatim	74		
<code>\hyxmp@define@pdfproducer</code> : Added this macro	60		

<code>\hyxmp@add@simple@pfx</code> : Added this macro	54	only if <code>\@pdfdatetime</code> is undefined	65
<code>\hyxmp@assign@major@minor</code> : Added this macro. hyperxmp 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 \TeX ecosystem and informing me how to activate it	61	<code>\hyxmp@detect@langs</code> : Added support for <code>babel</code>	40
<code>\hyxmp@cond@dc@identifier</code> : Added this macro	65	Refactored language detection into a separate command	40
<code>\next</code> : Define <code>\ifdraft</code> locally, at Niklas Beisert's request	24	<code>\hyxmp@parse@acmart</code> : Bug fix: Correct a missing "else" argument in two invocations of <code>\@if@def@and@nonempty</code>	39
v5.3		v5.5	
<code>\@if@def@and@nonempty</code> : Added this macro	20	General: Automatically assign <code>pdfnumpages</code> and <code>pdfbytes</code> under <code>pdf\LaTeX</code> and <code>Lua\LaTeX</code>	1
<code>\hyxmp@at@end</code> : Use <code>\AtEndDocument</code> in all \TeX back ends that provide it. Thanks to Nelson Posse Lago for pointing out why <code>atenddvi</code> is best avoided if possible	19	Correctly handle source files with spaces in their name. Thanks to Peter Dybala for the bug report	19
<code>\hyxmp@auto@assign@data</code> : Consider other author-provided sources of metadata. Thanks to Robin Schwab for proposing that hyperxmp use the KOMA letter classes's metadata	34	Defer <code>\AtEndPreamble</code> execution until the end of the document. This enables hyperxmp itself to be loaded from <code>\AtEndPreamble</code> , as is done by <code>doclicense v2.2.0</code> . Thanks to Tommaso Pecorella for the bug report and help testing	1
<code>\hyxmp@dc@schema</code> : Include all languages used in the document in <code>dc:language</code>	65	Introduced the <code>pdfpubstatus</code> package option, which enables an author to specify the document's publication status. Thanks to Robin Schwab for pointing me to the Journal Article Versions recommendation [1]	1
<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	40	Move most of the <code>\AtEndPreamble</code> code to <code>\hyxmp@at@end</code>	32
<code>\hyxmp@parse@acmart</code> : Added this macro	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	32
<code>\hyxmp@set@koma@phones</code> : Added this macro	33	Load <code>hyperref</code> automatically if the document does not do so explicitly, as requested by Robin Schwab	32
<code>\hyxmp@use@first@valid</code> : Added this macro	33	v5.4	
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>	32	<code>\hyxmp@auto@assign@data</code> : Moved the language-detection and \XeTeX date-detection code here from the <code>\hyxmp@at@end</code> block	34
<code>\hyxmp@dc@schema</code> : Bug fix: Use <code>\hyxmp@today@xmp</code> as the date		Moved title and author autodetection to the <code>\AtEndPreamble</code>	34

Use Lua _T _E X mechanisms, when available, to automatically compute the page count	34	v5.7	General: Do not automatically compute the PDF file size under pdf _L _A _T _E X because this confuses <code>latexmk</code> . Thanks to John Collins, Nelson Posse Lago, Derek Dreyer, and the other contributors to acmart issue #413 , “Latexmk goes into an infinite loop even on sample files from ACM”	35
<code>\hyxmp@detect@langs</code> : Set the language(s) immediately instead of deferring them to <code>\hyxmp@set@dc@lang</code>	40		<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	66
Store the main language in <code>\@pdflang</code> . Thanks to Javier Bezos for his help with the hyperxmp code and for modifying <code>babel</code> for hyperxmp’s benefit	40			
<code>\hyxmp@jav@extensions</code> : Added this macro	79	v5.8	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	17
<code>\hyxmp@jav@schema</code> : Added this macro	71		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	35
<code>\hyxmp@mm@extensions</code> : Corrected the type of <code>xmpMM:RenditionClass</code> . Thanks to Thorsten Wißmann for the bug report and patch	75			
<code>\hyxmp@query@self</code> : Added this macro	37	v5.9	General: At Karl Berry’s request, rename <code>add_byteCount</code> to the less generic-sounding <code>hyperxmp-add-bytecount</code>	17
<code>\hyxmp@rdf@dc</code> : List x-default alternatives before language-specific alternatives, as dictated by the XMP specification [5]	62			
Rewrite the core part of this macro to divide it into four, cleanly defined cases	62	v5.10	<code>\hyxmp+xml</code> : Include the <code>pdfxid</code> namespace only if the PDF/X version is 4 or greater. Thanks to John Lienhard for the bug report	81
<code>\hyxmp@set@koma@phones</code> : Support hyperlinks and other markup in <code>frommobilephone</code> and <code>fromphone</code> , as requested by Robin Schwab	33	v5.11	General: Disable <code>hyperxmp</code> if <code>L^AT_EX₃</code> document metadata is available. Document metadata implies the presence of PDF management, which completely breaks <code>hyperxmp</code>	18
<code>\hyxmp@xmptpg@schema</code> : Added this macro	71		<code>\@hyxmp@count</code> : Added this macro to fix a bug with <code>pdfapart</code> . Thanks to John H. Lienhard and Kartik Singhal for their bug reports	22
v5.6			<code>\hyxmp@at@end</code> : Use <code>\AddToHook</code>	
General: Don’t inadvertently replace underscores in filenames when writing font-related metadata	71			
Make <code>write_xmp_font_list</code> robust to fonts loaded using <code>HarfBuzz</code> . Thanks to John Lienhard for the bug report	71			
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			

when available. This addresses a bug reported on T _E X StackExchange by joHub and solved by Ulrike Fischer	19	Fischer for the bug report . . .	29
<code>\hyxmp@ProcessKeyvalOptions:</code> Bug fix: Restore		<code>\hyxmp@query@self:</code> Use <code>\thetotalpages</code> to compute the page count in an engine-independent manner. Thanks to Ulrike Fischer for recommending this mechanism	37
<code>\ProcessKeyvalOptions</code> after first use. Thanks to Ulrike			

Index

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

Symbols			
<code>\@acmBooktitle</code>	606	432, 483, 818,	<code>\@pdfcontactemail</code> ..
<code>\@acmConference</code> . . .	607	1308, 1366, 1416, <u>214</u> , 323,
<code>\@acmDOI</code> . .	577, 578, 580	1433, 1465, 1466,	465, 536, 1567, 1667
<code>\@acmISBN</code> .	587, 588, 591	1476, 1482, 1488	<code>\@pdfcontactphone</code> ..
<code>\@acmNumber</code>	621	<code>\@ifnextchar</code> ..	20, 1077
<code>\@acmVolume</code>	618	<code>\@ifnotmtarg</code> <u>212</u> ,
<code>\@author</code>	403, <u>533</u>	. 38, 85, 1071, 1080	324, 469, 1566, 1666
<code>\@baseurl</code> . . .	314, 1494	<code>\@ifnotmtargexp</code> . . .	<code>\@pdfcontactpostcode</code>
<code>\@elt</code> <u>679</u> , <u>1376</u> , <u>1539</u> , <u>1544</u>	 <u>37</u> , 380, <u>208</u> ,
<code>\@elt@first</code>	<u>1537</u>	444, 1053, 1094,	325, 566, 1563, 1664
<code>\@elt@rest</code> . .	<u>1539</u> , <u>1541</u>	1392, 1405, 1532	<code>\@pdfcontactregion</code> .
<code>\hyxmp@count</code> . .	<u>89</u> ,	<code>\@journalName</code>	605
92, 93, 101, 102,		<code>\@latex@warning@no@line</code> <u>206</u> ,
107, 110, 801,	 396	326, 554, 1562, 1663
802, 803, 804,		<code>\@pdfaconformance</code> ..	<code>\@pdfcontacturl</code> <u>216</u> ,
806, 808, 809,	 <u>95</u> , 371, 1504	327, 472, 1568, 1668
810, 811, 813,		<code>\@pdfapart</code>	<code>\@pdfcopyright</code>
1123, 1126, 1131,		364, 370, 376, 1503 <u>72</u> , 328,
1139, 1140, 1148,		<code>\@pdfauthor</code> 229, <u>253</u> ,	1414, 1449, 1455
1149, 1150, 1152,		315, 401, 1206, 1214	<code>\@pdfcreationdate</code> ..
1156, 1158, 1159,		<code>\@pdfauthortitle</code> <u>329</u> ,
1161, 1162, 1191,		78, 316, 1497, 1498	483, 484, 1476, 1480
1193, 1208, 1217,		<code>\@pdfbookedition</code> . .	<code>\@pdfcreator</code>
1222, 1229, 1230,		<u>176</u> , 317, 1583, 1673	1493
1231, 1511, 1512,		<code>\@pdfbytes</code>	<code>\@pdfdatetime</code>
1514, 1518, 1826,		<u>166</u> ,	<u>50</u> , 330, 1416, 1419
1827, 2170, 2181,		318, 527, 1592, 1674	<code>\@pdfdoi</code>
2188, 2190, 2198		<code>\@pdfcaptionwriter</code> .	<u>186</u> ,
<code>\@if@def@and@nonempty</code>	<u>80</u> , 319, 1497, 1499	331, 579, 1429,
.	<u>39</u> , 417,	<code>\@pdfcontactaddress</code>	1435, 1590, 1675
418, 424, 453,	 <u>196</u> ,	<code>\@pdfeissn</code>
523, 576, 586, 1319		320, 542, 1560, 1661	<u>172</u> , 332, 1430,
<code>\@ifclassloaded</code> . . .	624	<code>\@pdfcontactcity</code> <u>204</u> ,	1436, 1589, 1676
<code>\@ifmtarg</code>	<u>37</u> , 1082	321, 548, 1561, 1662	<code>\@pdfidentifier</code> . . .
<code>\@ifmtargexp</code>		<code>\@pdfcontactcountry</code> <u>190</u> , 333,
<u>37</u> , 41, 364, 381,	 <u>210</u> ,	1429, 1433, 1440
		322, 560, 1564, 1665	<code>\@pdfisbn</code>
			<u>174</u> ,
			334, 593, 1430,
			1438, 1587, 1677
			<code>\@pdfissn</code>
			<u>170</u> , 335, 1430,

1437, 1588, 1678	\@publishers 479	CreationDate 35, 95
\@pdfissuenumber 182,	\@subtitle 476	
336, 620, 1585, 1679	\@tempwafalse 1308, 1366	
\@pdfkeywords	\@tempwatrue . 1308,	D
. 235, 274, 337	1310, 1366, 1368	Date 46
\@pdflang	\@title 399	\day 796, 797, 799
. 338, 454, 459,	_ 953, 988, 1233	dc:creator . . . 2, 11, 65, 94
462, 629, 631, 646		dc:date 2, 65
\@pdflicenseurl	A	dc:description
76, 339, 1445, 1459	ACM 15, 33, 39	. . . 3, 11, 55, 65, 94
\@pdfmetadatetimestamp	acmart (class) 15, 35, 37–40	dc:format 2
61, 340, 1488, 1491	acmart (package) 98	dc:identifier 3, 65
\@pdfmetalang 82, 458,	add_byteCount 98	dc:language 2,
460, 462, 1077,	\addresses 574	15, 29, 65, 93, 94, 97
1324, 1340, 1353	\AddToHook 17	dc:publisher 3
\@pdfmoddate	Adobe PDF schema 59–62	dc:rights 2, 16, 55, 65
. . . 341, 1482, 1486	\affiliation 570	dc:source 2, 65, 93
\@pdfnumpages	ALPSP 7, 71	dc:subject 2, 65
. 168, 342, 525,	\and 253	dc:title . . . 3, 11, 55, 65, 94
1593, 1602, 1680	ASCII 20, 49	dc:type 3, 65
\@pdfpagerange	\AtBeginDocument . 1279	DCMI 8
184, 343, 1586, 1681	\AtEndDocument 14	\define@key 51, 62, 73,
\@pdfproducer . 1257,	\AtEndDvi 12	75, 77, 79, 81, 83,
1271, 1278, 1280	atenddvi (package) 19, 97	91, 96, 100, 119,
\@pdfpublication 162,	\AtEndPreamble	137, 139, 141,
344, 604, 1581, 1682	. . . 300, 392, 1464	143, 161, 163,
\@pdfpublisher	Author 11, 27, 96	165, 167, 169,
178, 478, 601, 1415		171, 173, 175,
\@pdfpubstatus	B	177, 179, 181,
. . . 194, 1596, 1691	babel (package) . . . 15,	183, 185, 187,
\@pdfpubtype . . . 164,	18, 29, 34, 40, 97, 98	189, 191, 193,
345, 614, 1582, 1683	\BabelEnsureInfo . . 631	195, 197, 205,
\@pdfrendition 154, 1473	Bag 94	207, 209, 211,
\@pdfsource	baseurl (option)	213, 215, 217,
. . . 136, 1424, 1426	5, 7, 15, 20, 25, 67	241, 253, 274, 1109
\@pdfsubject . 346, 1413	\bbl@main@language . 640	doclicense (package) . 97
\@pdfsubtitle . . 192,	BOM 80, 93	DOI 2, 7, 25, 38, 39
347, 475, 1580, 1684		DOS 36
\@pdftitle	C	draft (option) 8
. 348, 381, 397,	CiAdrCity 2	Dublin Core schema
1206, 1214, 1412	CiAdrCtry 2 2, 59, 62–66
\@pdftrapped 1302	CiAdrExtadr 2	DVI 36
\@pdftype 74, 1421	CiAdrPcode 2	dvipdf (option) 84
\@pdfuapart 99,	CiAdrRegion 2	Dvipdfm 11
349, 378, 1508, 1986	CiEmailWork 2	dvipdfm 84
\@pdfurl	CiTelWork 3	dvips (option) 84
188, 350, 1591, 1685	\city 546	dvips 11, 50, 93
\@pdfversionid 142, 1472	CiUrlWork 3	dvipsone (option) 84
\@pdfvolumenum . 180,	classes	dviwindo (option) . . . 84
351, 617, 1584, 1686	ACM 15, 33	
\@pdfxstandard	acmart . 15, 35, 37–40	E
. 117, 352, 377,	Koma 15, 33, 97	ε -TeX 60
1516, 1519, 1521	scrlltr2 15	\EdefEscapeHex 876, 889
\@pdfxversion 1990	\country 558	\EdefUnescapeHex . . 893
		\EdefUnescapeString 863

<code>\email</code>	534	1584 , 1585 , 1586 ,	<code>\hyxmp@as@xmp@date</code> .
<code>\empty</code>	1810	1587 , 1588 , 1589 , 56 , 67 ,
<code>\endinput</code>	6	1590 , 1591 , 1592 ,	695 , 833 , 1480 , 1486
<code>\equal</code>	113	1593 , 1596 , 1602	<code>\hyxmp@assign@major@minor</code>
etoolbox (package) . .	20	<code>\hyxmp@add@simple@lang</code> 1284 , 1303
ETX	42 , 48 1070 ,	<code>\hyxmp@at@end</code> 9 , 408
		1580 , 1581 , 1583	<code>\hyxmp@auto@assign@data</code>
		<code>\hyxmp@add@simple@lang@i</code> 409 , 452
	 1073 , 1076	<code>\hyxmp@banner@to@producer</code>
		<code>\hyxmp@add@simple@lang@ii</code> 1260 , 1263 , 1271
	 1077 , 1079	<code>\hyxmp@begin@ext@decl</code>
		<code>\hyxmp@add@simple@pfx</code> 1706 ,
	 1093 , 1406	1745 , 1756 , 1782 ,
		<code>\hyxmp@add@simple@var</code>	1798 , 1812 , 1828 ,
	 1061 ,	1840 , 1898 , 1968
		1300 , 1301 , 1304	<code>\hyxmp@begin@extension@decls</code>
		<code>\hyxmp@add@to@xml</code> 1694 , 1980
	 1055 ,	<code>\hyxmp@big@prime</code> . .
		1057 , 1065 , 1083 , 1128 ,
		1087 , 1095 , 1099 ,	1131 , 1141 , 1151
		1101 , 1220 , 1246 ,	<code>\hyxmp@big@prime@ii</code>
		1315 , 1334 , 1341 , 1128 , 1150
		1347 , 1354 , 1359 ,	<code>\hyxmp@bom</code> 2008 , 2023
		1371 , 1379 , 1385 ,	<code>\hyxmp@cct</code> 1600 , 1604
		1394 , 1534 , 1538 ,	<code>\hyxmp@check@iptc@data</code>
		1542 , 1548 , 1555 , 1659 , 2057
		1570 , 1695 , 1701 ,	<code>\hyxmp@check@jav@data</code>
		1707 , 1717 , 1724 , 1689 , 2059
		1728 , 1736 , 1849 ,	<code>\hyxmp@check@prism@data</code>
		1888 , 2023 , 1671 , 2058
		2041 , 2045 , 2074	<code>\hyxmp@check@standards</code>
		<code>\hyxmp@address@val</code> 362 , 410
	 534 , 540 ,	<code>\hyxmp@check@std</code> . .
		546 , 552 , 558 , 564 112 , 124 , 125 ,
		<code>\hyxmp@aep@toks</code> 25 ,	126 , 127 , 128 ,
		297 , 390 , 1462 , 2215	129 , 130 , 131 , 132
		<code>\hyxmp@alt@description</code>	<code>\hyxmp@comma</code>
	 1105 , 1115 198 , 254 , 275 , 666
		<code>\hyxmp@alt@rights</code> . .	<code>\hyxmp@commas@to@list</code>
	 1105 , 1116 650 , 686 , 1377 , 1546
		<code>\hyxmp@alt@title</code> . .	<code>\hyxmp@commas@to@list@i</code>
	 1105 , 1114 652 , 654
		<code>\hyxmp@and</code> 253	<code>\hyxmp@concat@metadata</code>
		<code>\hyxmp@append@hex</code> 297 , 311
	 1154 , 1173 ,	<code>\hyxmp@cond@dc@identifier</code>
		1174 , 1175 , 1179 1403 , 1435 ,
		<code>\hyxmp@append@hex@iii</code>	1436 , 1437 , 1438
	 1172 ,	<code>\hyxmp@construct@packet</code>
		1178 , 1188 , 1199 2021 , 2083
		<code>\hyxmp@append@hex@iv</code>	<code>\hyxmp@count@non@spaces</code>
	 1177 , 2183 , 2194
		1183 , 1184 , 1186 ,	<code>\hyxmp@count@spaces</code>
		1201 , 1202 , 1203 2182 , 2185
		<code>\hyxmp@as@pdf@date</code> . . 723	

<code>\hyxmp@crap@convert</code> 979, 1013	<code>\hyxmp@embed@packet@pdftex</code> 2086, 2114	<code>\hyxmp@lang@name</code>	.. 631
<code>\hyxmp@crap@result</code> 969, 1005	<code>\hyxmp@embed@packet@xetex</code> 2098, 2202	<code>\hyxmp@lang@tag</code>	... 631
<code>\hyxmp@crap@test</code> 976, 1001	<code>\hyxmp@end@ext@decl</code> 1716 , 1753, 1779, 1795, 1807, 1824, 1836, 1965, 1977	<code>\hyxmp@legal</code> 1444
<code>\hyxmp@create@uuid</code>	.. 1181 , 1209, 1218	<code>\hyxmp@list</code>	... 1377, 1383, 1546, 1547	<code>\hyxmp@list@to@lines</code>	... 1531 , 1560, 1566, 1567, 1568
<code>\hyxmp@cur@lang</code> 1111 , 1119	<code>\hyxmp@end@extension@decl</code> 1700 , 2006	<code>\hyxmp@list@to@xml</code> 1365 ,
<code>\hyxmp@dc@lang</code>	. 454, 625 , 631 , 647 , 1428	<code>\hyxmp@extra@indent</code> 1051 , 1055, 1066, 1095, 1535, 1559	<code>\hyxmp@list@to@xml</code> 1365 ,
<code>\hyxmp@dc@schema</code> 1410 , 2063	<code>\hyxmp@first@char</code>	.. 693	<code>\hyxmp@major@minor</code>	1284
<code>\hyxmp@declare@extensions</code> 1979 , 2060	<code>\hyxmp@first@char@i</code> 693 , 696, 724	<code>\hyxmp@mm@extensions</code> 1755 , 1982
<code>\hyxmp@declare@field</code> 1735 , 1864, 1867, 1870, 1873, 1876, 1879, 1882, 1885	<code>\hyxmp@gobbletwo</code>	763 , 776	<code>\hyxmp@mm@schema</code> 1469 , 2069
<code>\hyxmp@declare@property</code>	... 1723 , 1749, 1760, 1765, 1770, 1774, 1786, 1791, 1802, 1816, 1820, 1832, 1844, 1902, 1906, 1910, 1914, 1918, 1922, 1926, 1930, 1934, 1938, 1942, 1946, 1951, 1956, 1961, 1972	<code>\hyxmp@hash</code> 1240 , 2027, 2035, 2049, 2052, 2053, 2054, 2055	<code>\hyxmp@modulo@a</code> 1122 , 1141, 1151, 1157, 1192
<code>\hyxmp@def@DocumentID</code> 1205 , 1465	<code>\hyxmp@Hyp@pdfauthor</code> 247	<code>\hyxmp@multi@langsfalse</code> 1306 , 1322
<code>\hyxmp@def@InstanceID</code> 1211 , 1466	<code>\hyxmp@Hyp@pdfkeywords</code> 268	<code>\hyxmp@multi@langstrue</code> 1306 , 1320
<code>\hyxmp@define@pdfproducer</code> 1257 , 1281	<code>\hyxmp@Hyp@pdfkeywords</code> 268	<code>\hyxmp@new+xml</code>	1236 , 1237
<code>\hyxmp@detect@langs</code> 456 , 626	<code>\hyxmp@hypersetup</code>	.. 292	<code>\hyxmp@no@bad@parts</code> 84 , 92, 101
<code>\hyxmp@DocumentID</code> 138 , 1205 , 1465, 1470	<code>\hyxmp@InstanceID</code> 140 , 1211 , 1466, 1471	<code>\hyxmp@no@info@lists</code> 218 , 242, 394
<code>\hyxmp@dq@code</code>	. 7 , 2216	<code>\hyxmp@iprefix</code>	1097, 1098	<code>\hyxmp@num</code> 1013
<code>\hyxmp@driver</code>	... 2082	<code>\hyxmp@iptc@data</code>	.. 1553, 1659 , 1994	<code>\hyxmp@one@token</code> 1130 , 1134 , 2186, 2187, 2195, 2196
<code>\hyxmp@embed@packet</code> 412 , 2082	<code>\hyxmp@iptc@extensions</code> 1839 , 1996	<code>\hyxmp@padding</code>	1244 , 2078
<code>\hyxmp@embed@packet@dvipdfm</code> 2094, 2164	<code>\hyxmp@iptc@schema</code> 1552 , 2070	<code>\hyxmp@parse@acmart</code> 481 , 531 , 624
<code>\hyxmp@embed@packet@luatex</code> 2090 , 2127	<code>\hyxmp@is@unicode</code> 865 , 882, 897	<code>\hyxmp@parse@time</code> 704 , 706
<code>\hyxmp@embed@packet@pdfmark</code> 2106 , 2134	<code>\hyxmp@jav@data</code> 1689 , 2002	<code>\hyxmp@parse@tz</code> 713 , 716, 720
<code>\hyxmp@koma@phones</code> 414 , 470	<code>\hyxmp@jav@extensions</code> 1967 , 2004	<code>\hyxmp@parse@tz@char</code> 708 , 710
<code>\hyxmp@LA@accept</code> 1108 ,	<code>\hyxmp@jav@schema</code> 1595 , 2072	<code>\hyxmp@pdf@extensions</code> 1744 , 1981
		<code>\hyxmp@jobname</code> 22 , 23, 136,	<code>\hyxmp@pdf@schema</code> 1299 , 2061
		<code>\hyxmp@jobname</code> 356, 515, 831, 1206, 1214, 2103	<code>\hyxmp@pdf@to@xmp@date</code>	. 697, 702 , 825, 828
		<code>\hyxmp@jobname</code> 356, 515, 831, 1206, 1214, 2103	<code>\hyxmp@pdfa@id@extensions</code> 1781 , 1984
		<code>\hyxmp@jobname</code> 356, 515, 831, 1206, 1214, 2103	<code>\hyxmp@pdfa@id@schema</code> 1501 , 2066

<code>\hyxmp@pdfauthor</code> ..	<code>\hyxmp@redefine@Hyp</code>	<code>\hyxmp@today@xmp</code> ..
... 244 , 253 , 1422	... 246 , 289 , 294	... 818 , 823 ,
<code>\hyxmp@pdfkeywords</code> .	<code>\hyxmp@remove@this</code> .	841 , 1214 , 1417 ,
... 244 , 274 , 1423	... 1275 , 1278	1477 , 1483 , 1489
<code>\hyxmp@pdfstringdef</code>	<code>\hyxmp@rights</code> . 1444 ,	<code>\hyxmp@today@xmp@define</code>
... 44 ,	1447 , 1451 , 1453	... 789 , 838 , 1212
55 , 66 , 73 , 75 ,	<code>\hyxmp@seed@rng</code> ...	<code>\hyxmp@toxml</code> .. 891 , 914
77 , 79 , 81 , 83 , 93 ,	.. 1130 , 1207 , 1216	<code>\hyxmp@toxml@unicodetex</code>
97 , 102 , 120 , 137 ,	<code>\hyxmp@seed@rng@i</code> 879 , 939
139 , 141 , 143 ,	... 1132 , 1134	<code>\hyxmp@trimb</code> .. 847 , 850
161 , 163 , 165 ,	<code>\hyxmp@seed@string</code> .	<code>\hyxmp@trimc</code> .. 850 , 851
167 , 169 , 171 ,	... 1205 , 1211	<code>\hyxmp@trimspaces</code> ..
173 , 175 , 177 ,	<code>\hyxmp@set@jobname</code> 659 , 843
179 , 181 , 183 ,	... 19 , 24	<code>\hyxmp@try</code> ... 969
185 , 187 , 189 ,	<code>\hyxmp@set@jobname@dbl</code>	<code>\hyxmp@try@today</code> 817 ,
191 , 193 , 195 ,	... 20 , 22	824 , 827 , 830 , 837
200 , 205 , 207 ,	<code>\hyxmp@set@jobname@plain</code>	<code>\hyxmp@unicodetexfalse</code>
209 , 211 , 213 ,	... 20 , 23	... 853
215 , 217 , 419 ,	<code>\hyxmp@set@koma@phones</code>	<code>\hyxmp@unicodetextrue</code>
421 , 425 , 1097 , 1110	... 414 , 468	... 853
<code>\hyxmp@pdfua@id@extensions</code>	<code>\hyxmp@set@pdfx@major</code>	<code>\hyxmp@uscore</code> .. 46 , 670
... 1797 , 1988	... 104 , 134	<code>\hyxmp@use@first@valid</code>
<code>\hyxmp@pdfua@id@schema</code>	<code>\hyxmp@set@pdfx@major@i</code>	.. 397 , 401 , 431 ,
... 1507 , 2067	... 104 , 105	465 , 469 , 472 ,
<code>\hyxmp@pdfx@id@extensions</code>	<code>\hyxmp@set@pdfx@major@ii</code>	475 , 478 , 527 ,
... 1809 , 1992	... 106 , 109	536 , 542 , 548 ,
<code>\hyxmp@pdfx@id@schema</code>	<code>\hyxmp@set@rand@num</code>	554 , 560 , 566 ,
... 1510 , 2068	.. 1147 , 1155 , 1190	579 , 593 , 601 ,
<code>\hyxmp@pdfx@major</code> ..	<code>\hyxmp@singleton@dc</code>	604 , 614 , 617 , 620
... 109 ,	... 1391 , 1415 ,	<code>\hyxmp@use@first@valid@i</code>
118 , 134 , 1511 ,	1417 , 1419 , 1421	... 433 , 437
1810 , 1826 , 2040	<code>\hyxmp@skiptorelax</code> .	<code>\hyxmp@value</code> 1110 , 1314
<code>\hyxmp@photoshop@data</code>	... 1006 , 1012	<code>\hyxmp@warn@if@no@metadata</code>
... 1496	<code>\hyxmp@skipzeros</code> .. 964	... 311 , 411
<code>\hyxmp@photoshop@schema</code>	<code>\hyxmp@space@other</code> ..	<code>\hyxmp@x@default</code> ..
... 1496 , 2064	... 973 , 986	... 460 , 1256 ,
<code>\hyxmp@prev@pdf@size</code>	<code>\hyxmp@standards</code> .. 375	1324 , 1335 , 1342
... 513 , 528	<code>\hyxmp@string@len</code> ..	<code>\hyxmp@xetex@crap</code> ..
<code>\hyxmp@prism@data</code> 2165 , 2180	... 870 , 969
.. 1576 , 1671 , 1998	<code>\hyxmp@strip@isbn@date</code>	<code>\hyxmp+xml</code> 1056 , 1058 ,
<code>\hyxmp@prism@extensions</code>	... 586	1096 , 1102 , 1237 ,
... 1897 , 2000	<code>\hyxmp@sublist</code> ...	1244 , 1727 , 2021 ,
<code>\hyxmp@prism@schema</code>	.. 655 , 656 , 659 , 660	2123 , 2131 , 2154 ,
... 1575 , 2071	<code>\hyxmp@suppress@pdf@info</code>	2165 , 2172 , 2203
<code>\hyxmp@ProcessKeyvalOptions</code>	... 219	<code>\hyxmp+xmlified</code> ...
... 286	<code>\hyxmp@temp@list</code> .. 679	... 861 , 1057 ,
<code>\hyxmp@prot@us</code> ... 1608	<code>\hyxmp@temp@str</code> ... 679	1066 , 1073 , 1084 ,
<code>\hyxmp@query@self</code> ..	<code>\hyxmp@text</code> ...	1088 , 1099 , 1101 ,
... 488 , 522	... 861 , 939 , 969 , 1013	1314 , 1343 , 1348 ,
<code>\hyxmp@rand@num</code> ...	<code>\hyxmp@text@underscore</code>	1355 , 1377 , 1397 ,
... 1147 , 1156 ,	... 44	1404 , 1434 , 1546
1191 , 1208 , 1217	<code>\hyxmp@timestamp</code> .. 830	<code>\hyxmp+xmlify</code> ...
<code>\hyxmp@rdf@dc</code> . 1307 ,	<code>\hyxmp@today@pdf</code> 484 , 840	... 861 , 1054 ,
1412 , 1413 , 1414		1064 , 1072 , 1081 ,

1098, 1100, 1313,
1340, 1346, 1353,
1376, 1393, 1545
\hyxmp@xmp@basic@schema
..... 1475, 2065
\hyxmp@xmp@to@pdf@date
..... 727, 730, 841
\hyxmp@xmp@to@pdf@date@i
..... 731, 733
\hyxmp@xmp@to@pdf@date@ii
..... 736, 739
\hyxmp@xmp@to@pdf@date@iii
..... 742, 745
\hyxmp@xmp@to@pdf@date@iv
..... 748, 751
\hyxmp@xmp@to@pdf@date@v
..... 754, 757
\hyxmp@xmp@to@pdf@date@vi
..... 760, 764
\hyxmp@xmp@to@pdf@date@vii
..... 767, 770, 780
\hyxmp@xmp@to@pdf@date@viii
..... 783, 786
\hyxmp@xmpRights@schema
..... 1443, 2062
\hyxmp@xmptpg@schema
..... 1598, 2073
\hyxmp@zero
..... 1022, 1029,
1036, 1042, 1047

I

IETF 6
\if@ACM@journal ... 609
\if@tempswa . 1312, 1370
\IfDocumentMetadataTF
..... 1, 6
ifdraft (package) 18, 24, 98
\ifdraft 144, 154
\iffalse ... 1307, 1365
\ifHy@pdfa
363, 1412, 1413,
1422, 1502, 1983
\ifhyxmp@multi@langs
.. 1306, 1325, 1339
\ifhyxmp@unicodetex
853, 864, 1223, 2009
\ifLuaTeX
. 34, 490, 1259,
1599, 1604, 1612
ifluatex (package) ... 83
\ifluatex .. 2116, 2120
ifmtarg (package) . 20, 95
\ifPDFTeX 1262

J

JAV 73, 79, 80
jav:journal_article_ver-
sion 2
\jobname 24
Journal Article Versions
schema ... 59, 71

K

keeppdfinfo (option) 12, 27
Keywords . 11, 27, 61, 96
Koma (class) . 15, 33, 97
\KV@Hyp@pdfauthor .. 253
\KV@Hyp@pdfkeywords 274
kvoptions (package) 20, 29

L

latexmk .. 17, 18, 35, 98
LF 69
\LocaleForEach 632
Lua 71
luacode (package) ... 20
\luadirect ... 514, 1600
LuaL^AT_EX 9,
11, 13, 14, 17, 35,
46, 47, 61, 94, 97
LuaT_EX 20,
35, 48, 51, 71,
83, 86, 92, 94–96, 98

luatex85 (package) ... 96
\luatexbanner 1260

M

\makeatletter 1275
memoir (package) ... 93
Metadata 82, 85
\month 791, 792, 794

N

NAK 20, 42, 48
nativepdf (option) ... 84
\newcatcodetable . 1605
\newcount 89
\newif 853, 1306
\newtoks 25
\next ... 54, 65, 114,
121, 144, 159,
219, 438, 440,
446, 450, 654,
832, 835, 1134, 1990
ngerman (package) 18, 92
NISO 7, 71
\number 1016,
1018, 1020, 1025,
1027, 1032, 1034
\numexpr 2132

O

options
baseurl
5, 7, 15, 20, 25, 67
draft 8
dviPDF 84
dvips 84
dvipsone 84
dviwindo 84
keeppdfinfo ... 12, 27
nativepdf 84
pdfa ... 9, 22, 31, 94
pdfaconformance .
..... 5, 9, 68
pdfapart 5, 9, 22, 68, 98
pdfauthor 5,
6, 11, 14, 15, 20,
27, 28, 32, 65, 94, 97
pdfauthortitle . 5, 6, 15
pdfbookedition .. 5, 7
pdfbytes . 5, 9, 37, 97
pdfcaptionwriter . 5, 6
pdfcontactaddress .
..... 5, 6, 12, 13
pdfcontactcity .. 5, 6
pdfcontactcountry 5, 6

pdfcontactemail . 5, 6
pdfcontactphone . 5, 6
pdfcontactpostcode
. 5, 6
pdfcontactregion . 5, 6
pdfcontacturl . 5, 6, 15
pdfcopyright
. 5, 6, 65, 66, 93
pdfcreationdate
. 5, 8, 35, 95
pdfdate 5, 7, 8,
15, 21, 58, 65, 94, 96
pdfdocumentid . 5, 6, 95
pdfdoi 5, 7
pdffeissn 5, 7
pdfidentifier . 5, 7, 65, 96
pdfinstanceid . 5, 6, 95
pdfisbn 5–7
pdfissn 5, 7
pdfissuenum 5, 7
pdfkeywords 5,
11, 14, 20, 27, 28, 65
pdflang 5–7,
15, 20, 40, 55, 65, 95
pdflicenseurl
. 5, 6, 15, 66, 93
pdfmark 84
pdfmetadate . 5, 8, 21, 95
pdfmetalang . 5, 6,
15, 55, 62, 63, 92, 95
pdfmoddate . . 5, 8, 95
pdfnumpages . 5, 9, 97
pdfpagerange
. 5, 7, 16, 17
pdfproducer . 5, 20, 61
pdfpublication 5–7
pdfpublisher 5, 7
pdfpubstatus . 5, 7, 8, 97
pdfpubtype 5, 7
pdfrendition . . 5, 8, 96
pdfsource 5, 9, 95
pdfsubject . 5, 11, 20, 65
pdfsubtitle 5
pdftitle 5, 11,
15, 20, 32, 65, 94, 97
pdftrapped
. 5, 8, 9, 20, 96
pdftype 5, 8, 65, 94
pdfuapart . 5, 9, 22, 68
pdfurl 5, 7
pdfversionid . . 5, 6, 67
pdfvolumenum . . 5, 7
pdfxstandard
. 5, 9, 22, 23, 68, 81
ps2pdf 84
textures 84
unicode 15, 95
vtxpdfmark 84

P

\PackageError 1326
packages
acmart 98
atenddvi 19, 97
babel 15,
18, 29, 34, 40, 97, 98
doclicense 97
etoolbox 20
gitver 6
hyperref 1,
4–6, 8, 9, 11, 15,
18–20, 22, 27–29,
31, 32, 40, 41,
60, 61, 82–84, 93–97
hyperxmp
1, 2, 4–9, 11–20,
22, 23, 26–29, 32,
33, 35, 40, 41, 43,
48, 56, 60, 61, 72,
83, 86, 93–95, 97, 98
ifdraft 18, 24, 98
ifluatex 83
ifmtarg 20, 95
iftex 20, 96
ifthen 20
intcalc 20
kvoptions 20, 29
luacode 20
luatex85 96
memoir 93
ngerman 18, 92
pdfescape 20
pdfx 4
polyglossia . 15, 18,
29, 33, 34, 40, 41, 97
stringenc 20
texdate 16
totpages 16, 17
xmpincl 4
\PackageWarning
. 2, 86, 122, 680
\PackageWarningNoLine
. 221,
355, 365, 382, 2101
\patchcmd 227, 233
PDF 1–5, 8,
11–14, 16–18, 21,
27, 31–33, 35, 36,
41, 43, 44, 47, 49,
50, 58–61, 71, 74,
79, 81–83, 86, 94,
95, 97, 98, 100, 107
Author 11, 27, 96
CreationDate 35, 95
Info 11,
12, 27, 32, 35, 60
Keywords 11, 27, 61, 96
Metadata 82, 85
Producer 61
Subject 11
Title 11
PDF/A 3, 9, 11,
12, 22, 27, 31, 60,
61, 68, 72, 75–80,
93, 94, 105, 107
PDF/A Identification
schema 59, 68
PDF/UA . 3, 9, 22, 31, 68,
76, 80, 96, 105, 107
PDF/UA Identification
schema 59, 68
PDF/X 3, 9, 22,
23, 31, 68, 76, 80,
81, 96, 98, 105, 107
PDF/X Identification
schema 59, 68
pdf:Keywords 2, 11, 61
pdf:PDFVersion . . 3, 61, 97
pdf:Producer 3, 60, 61
pdf:trapped 3
\PDF@FinishDoc
. 220, 228, 234
pdfa (option) . 9, 22, 31, 94
pdfaconformance (op-
tion) 5, 9, 68
pdfaid:conformance 3
pdfaid:part 3
pdfapart (option)
. 5, 9, 22, 68, 98
pdfaType:prefix 93
pdfauthor (option) . 5,
6, 11, 14, 15, 20,
27, 28, 32, 65, 94, 97
pdfauthortitle (option)
. 5, 6, 15
pdfbookedition (option)
. 5, 7
pdfbytes (option)
. 5, 9, 37, 97
pdfcaptionwriter (op-
tion) 5, 6
\pdfcatalog 2124

`\pdfcompresslevel` . 2118
`pdfcontactaddress` (option) . 5, 6, 12, 13
`pdfcontactcity` (option) 5, 6
`pdfcontactcountry` (option) 5, 6
`pdfcontactemail` (option) 5, 6
`pdfcontactphone` (option) 5, 6
`pdfcontactpostcode` (option) 5, 6
`pdfcontactregion` (option) 5, 6
`pdfcontacturl` (option) 5, 6, 15
`pdfcopyright` (option) 5, 6, 65, 66, 93
`pdfcreationdate` (option) 5, 8, 35, 95
`\pdfcreationdate` 825
`pdfdate` (option) 5, 7, 8, 15, 21, 58, 65, 94, 96
`PDFDocEncoding` 27, 48, 49
`pdfdocumentid` (option) 5, 6, 95
`pdfdoi` (option) 5, 7
`pdfeissn` (option) 5, 7
`pdfescape` (package) 20
`\pdfextension` 2128, 2132
`\pdffeedback` 828, 2132
`pdfidentifier` (option) 5, 7, 65, 96
`pdfinstanceid` (option) 5, 6, 95
`pdfisbn` (option) 5–7
`pdfissn` (option) 5, 7
`pdfissuenum` (option) 5, 7
`pdfkeywords` (option) 5, 11, 14, 20, 27, 28, 65
`pdflang` (option) . 5–7, 15, 20, 40, 55, 65, 95
`\pdflastobj` 2124
`pdfLATEX` 4, 9, 11, 14, 35, 46, 61, 97, 98
`pdflicenseurl` (option) 5, 6, 15, 66, 93
`\pdfmajorversion` . 1292
`pdfmark` (option) 84
`\pdfmark` 2135, 2138, 2142, 2152, 2156, 2160
`pdfmetadate` (option) 5, 8, 21, 95
`pdfmetalang` (option) 5, 6, 15, 55, 62, 63, 92, 95
`\pdfminorversion` . 1288
`pdfmoddate` (option) 5, 8, 95
`pdfnumpages` (option) 5, 9, 97
`\pdfobj` 2120
`pdfpagerange` (option) 5, 7, 16, 17
`pdfproducer` (option) 5, 20, 61
`pdfpublication` (option) 5–7
`pdfpublisher` (option) 5, 7
`pdfpubstatus` (option) 5, 7, 8, 97
`pdfpubtype` (option) . 5, 7
`pdfrendition` (option) 5, 8, 96
`pdfsource` (option) 5, 9, 95
`\pdfstringdef` 47
`pdfsubject` (option) 5, 11, 20, 65
`pdfsubtitle` (option) 5
`pdfTEX` 50, 83, 86, 95, 96
`\pdftexbanner` 1263
`pdftitle` (option) 5, 11, 15, 20, 32, 65, 94, 97
`pdftrapped` (option) 5, 8, 9, 20, 96
`pdftype` (option) 5, 8, 65, 94
`pdfuaid:part` 3
`pdfuapart` (option) 5, 9, 22, 68
`pdfurl` (option) 5, 7
`\pdfvariable` 1296
`pdfversionid` (option) 5, 6, 67
`pdfvolumenum` (option) 5, 7
`pdfx` (package) 4
`pdfxid:GTS_PDFXVersion` 3
`pdfxstandard` (option) 5, 9, 22, 23, 68, 81
`Perl` 17
`Photoshop schema` 59, 68
`photoshop:AuthorsPosition` 3, 68
`photoshop:CaptionWriter` 2, 68
`PI` 58, 59
`polyglossia` (package) 15, 18, 29, 33, 34, 40, 41, 97
`\postcode` 564
`PRISM` 7, 17, 59, 70, 73, 78, 80, 106, 107
`PRISM Basic Metadata schema` 17, 59, 70–71
`prism:aggregationType` . 3
`prism:bookEdition` 2
`prism:byteCount` 2, 17
`prism:doi` 2
`prism:elssn` 2
`prism:isbn` 2
`prism:issn` 2
`prism:number` 2
`prism:pageCount` 3
`prism:pageRange` 3
`prism:publicationName` . 3
`prism:subtitle` 3
`prism:url` 3
`prism:volume` 3
`\ProcessKeyvalOptions` 286
`Producer` 61
`properties, XMP`
`dc:creator` 2, 11, 65, 94
`dc:date` 2, 65
`dc:description` 3, 11, 55, 65, 94
`dc:format` 2
`dc:identifier` 3, 65
`dc:language` 2, 15, 29, 65, 93, 94, 97
`dc:publisher` 3
`dc:rights` 2, 16, 55, 65
`dc:source` 2, 65, 93
`dc:subject` 2, 65
`dc:title` 3, 11, 55, 65, 94
`dc:type` 3, 65
`lptc4xmpCore:ContactInfo` 69, 77
`lptc4xmpCore:CreatorContactInfo` 2, 3, 69, 70, 93
`jav:journal_article_version` 2
`pdf:Keywords` 2, 11, 61
`pdf:PDFVersion` 3, 61, 97
`pdf:Producer` 3, 60, 61
`pdf:trapped` 3

pdfaid:conformance	3	rdf:li	2	\StringEncodingConvert	866,
pdfaid:part	3	rdf:Seq	2		872, 883, 886, 981
pdfaType:prefix	93	\renewcommand	287	Subject	11
pdfauid:part	3	\RequirePackage	11, 26, 27, 28,		
pdfxid:GTS_PDFXVersion	3		29, 30, 31, 32, 33,		
photoshop:Author-Position	3, 68		35, 149, 393, 2113		
photoshop:Caption-Writer	2, 68	S		T	
prism:aggregation-Type	3	\savecatcodetable	1606	TeX	16, 19, 20,
prism:bookEdition	2	\scantokens	1272, 1275		46–48, 50, 51, 56,
prism:byteCount	2, 17	schemata			58, 60, 67, 85, 86, 97
prism:doi	2	Adobe PDF	59–62	texdate (package)	16
prism:elssn	2	Dublin Core	2, 59, 62–66	Text	74
prism:isbn	2	IPTC Photo Meta- data	59, 69–70	\textunderscore	45, 46, 48
prism:issn	2	Journal Article Ver- sions	59, 71	textures (option)	84
prism:number	2	PDF/A Identifica- tion	59, 68	\thetotalpages	525
prism:pageCount	3	PDF/UA Identifica- tion	59, 68	\time	801, 809
prism:pageRange	3	PDF/X Identifica- tion	59, 68	Title	11
prism:publication- Name	3	Photoshop	59, 68	totpages (package)	16, 17
prism:subtitle	3	PRISM Basic Meta- data	17, 59, 70–71		
prism:url	3	XMP Basic	59, 67	U	
prism:volume	3	XMP Media Manage- ment	59, 66–67	\undefined	442
xmp:BaseURL	2	XMP Paged-Text	59, 71–72	Unicode	15, 20, 48–52,
xmp:CreateDate	2, 35, 94	XMP Rights Manage- ment	59, 66		64, 69, 80, 86, 93
xmp:CreatorTool	3	\scr@fromemail@var	466	unicode (option)	15, 95
xmp:MetadataDate	2, 95	\scr@frommobilephone@var	419, 421	URI	7
xmp:ModifyDate	2, 94	\scr@fromname@var	402	URL	2, 3, 6, 7, 15,
xmpMM:Document- tID	3, 56, 67, 79	\scr@fromphone@var	419, 425		21, 25, 27, 66, 67, 70
xmpMM:InstanceID	3, 56, 67, 79	\scr@fromurl@var	473	UTF-16BE	49
xmpMM:Rendition- Class	3, 98	\scr@subject@var	398	UTF-32BE	49
xmpMM:VersionID	3, 67, 96	scrltr2 (class)	15	UTF-8	49
xmpRights:Marked	2, 66, 93	\SE->pdfdoc@03	859	UUID	3, 6,
xmpRights:WebState- ment	2, 66, 93	\SE->pdfdoc@15	860		23, 24, 56–58, 66, 94
ps2pdf (option)	84	\setbox	574	V	
ps2pdf	11	\setkeys	1120	\vfuzz	851
		\special	2166,	vtexpdfmark (option)	84
Q			2174, 2203, 2209		
\Q	843, 852	\state	552	X	
		\streetaddress	540	\x	969
R		stringenc (package)	20	x-default	6, 15, 34, 55,
RDF	64				60, 62, 63, 92, 95, 98
rdf:Description	95			xdvipdfmx	14, 35, 85

