

The hyperxmp package*

Scott Pakin
scott+hyxmp@pakin.org

October 5, 2020

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>
```

*This document corresponds to `hyperxmp` v5.6, dated 2020/10/05.

```

    <rdf:li>Harvey Dent</rdf:li>
  </rdf:Seq>
</dc:creator>

```

In the preceding code, the `dc` namespace refers to the Dublin Core schema, a collection of metadata properties. The `dc:creator` property surrounds the list of authors. The `rdf` namespace is the Resource Description Framework, which defines `rdf:Seq` as an ordered list of values. Each author is represented by an individual list item (`rdf:li`), making it easy for an XML parser to separate the authors' names.

Remember that XMP code is stored as *metadata*. It does not appear when viewing or printing the PDF file. Rather, it is intended to make it easy for computer applications to identify and categorize the document.

1.1 Supported metadata

hyperxmp knows how to embed all of the following types of metadata within a document:

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

- journal article version (jav:journal_article_version)
- keywords (pdf:Keywords and dc:subject)
- language used (dc:language)
- license URL (xmpRights:WebStatement)
- metadata writer (photoshop:CaptionWriter)
- page count (prism:pageCount)
- page range(s) (prism:pageRange)
- PDF version (pdf:PDFVersion)
- PDF-generating tool (pdf:Producer and xmp:CreatorTool)
- PDF/A version and conformance level (pdfaid:part and pdfaid:conformance)
- PDF/UA version (pdfuaid:part)
- PDF/X standard compliance (pdfxid:GTS_PDFXVersion)
- position/title of primary author (photoshop:AuthorsPosition)
- publication name of parent publication (prism:publicationName)
- publisher of the document (dc:publisher)
- rendition variation of the document (xmpMM:RenditionClass)
- summary (dc:description)
- subtitle (prism:subtitle)
- telephone number(s) of primary author
(Iptc4xmpCore:CreatorContactInfo.CiTelWork)
- title (dc:title)
- trapping of colors (pdf:trapped)
- type of document (dc:type)
- type of parent publication (prism:aggregationType)
- unique identifier for the document (dc:identifier)
- URL of the document (prism:url)
- URL(s) of the primary author (Iptc4xmpCore:CreatorContactInfo.CiUrlWork)
- UUID for the document (xmpMM:DocumentID)

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

Figure 1: Comparison of pdfx and hyperxmp

- UUID for the document instance (`xmpMM:InstanceID`)
- version identifier for the document (`xmpMM:VersionID`)
- volume number of parent publication (`prism:volume`)

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

1.2 Comparisons with similar packages

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

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

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

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

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

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

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

2 Usage

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

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

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

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

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

2.1 Option descriptions

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

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

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

`pdfauthor` `hyperref`’s `pdfauthor` option specifies the document’s author(s). See Note 4 on page 15 for a discussion of the correct syntax. If `pdfauthor` is not specified it will inherit its value from the document’s `\author`. `pdfauthortitle` indicates the primary author’s position or title. `pdfcaptionwriter` specifies the name of the person who added the metadata to the document.

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

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

`pdfmetalang` indicates the natural language in which certain metadata—specifically, the document’s title, subject, and copyright statement—are written. The language should be specified using an IETF language tag [11], for example, “en” for English, “en-US” for specifically United States English, “de” for German, and so forth. If `pdfmetalang` is not specified, `hyperxmp` assumes the metadata language is the same as the document language (`hyperref`’s `pdflang` option). If neither `pdfmetalang` nor `pdflang` is specified, `hyperxmp` uses only “x-default” as the metadata language.

`pdfdocumentid` XMP can include a universally unique identifier (UUID) for each document and for each instance of a given document. By default, `hyperxmp` assigns a version 4 (i.e., pseudorandom) UUID [12] for each of these. However, a document can

pdfinstanceid	alternatively specify a particular document identifier using <code>pdfdocumentid</code> and (not normally recommended) a particular instance identifier using <code>pdfinstanceid</code> . These should be of the form <code>uuid:xxxxxx-xxx-xxx-xxx-xxxxxxxxxx</code> , where “x” is a lowercase hexadecimal number. For example, <code>uuid:53ab7f19-a48c-5177-8bb2-403ad907f632</code> is a valid argument to <code>pdfdocumentid</code> (or <code>pdfinstanceid</code>). See Leach, Mealling, and Salz’s UUID specification document for details on how to produce the various forms of UUIDs [12]. A more freeform mechanism than <code>pdfinstanceid</code>
pdfversionid	for versioning documents is available via <code>pdfversionid</code> . The version specified by <code>pdfversionid</code> can be incremented as 1, 2, 3, ...; identified with a hierarchical numbering scheme (e.g., this document is versioned 5.6 to match the package version); or labeled using any other approach. One possibility is to use a revision number or commit hash from the version-control software maintaining the document. For example, the <code>\gitVer</code> macro from the <code>gitver</code> package is an expandable (see Note 8 on page 17) version of the current Git hash that can suitably be passed to <code>pdfversionid</code> . If not specified, <code>pdfversionid</code> defaults to 1.
pdfisbn	Already-published documents can be identified in a number of ways. <code>pdfisbn</code> specifies the ISBN. <code>pdfissn</code> refers to the ISSN of the <i>print</i> version of the document while <code>pdfeissn</code> refers to the ISSN of the <i>electronic</i> version of the document. <code>pdfdoi</code> specifies the DOI and should include only the DOI name without any URL prefix. For example, specify <code>pdfdoi={10.1145/3149526.3149532}</code> , <i>not</i> <code>pdfdoi={https://doi.org/10.1145/3149526.3149532}</code> . <code>pdfurl</code> points to the complete URL for the document. In contrast, <code>baseurl</code> points one level up and is used to resolve relative URLs.
pdfissn	
pdfeissn	
pdfdoi	
pdfurl	
baseurl	
pdfidentifier	<code>pdfidentifier</code> provides an alternative mechanism to uniquely identify a document. Its advantage relative to <code>pdfisbn</code> , <code>pdfissn</code> , <code>pdfdoi</code> , etc. is its flexibility; any of a wide variety of identification types can be used. 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.
pdfpublication	Already-published documents can further be identified by the publication in which they appear. <code>pdfpublication</code> specifies the title of the journal, magazine, or other parent document. The title language is assumed to be the same as the document language (<code>pdflang</code>) but can be overridden by preceding the text with a bracketed ISO 639-1 two-letter language code and an optional ISO 3166-1 two-letter region code. For example, <code>pdfpublication={[fr]Charlie Hedbo}</code> indicates a French-language title. Were the language or pronunciation differences significant, <code>fr-FR</code> would indicate specifically the French spoken in France, as opposed to that spoken in, say, Canada (<code>fr-CA</code>) or Belgium (<code>fr-BE</code>). The publisher itself can be

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

`pdfpublisher` named using `pdfpublisher`.

`pdfpubtype` `pdfpubtype` indicates the type of publication in which the document was published. This should be one of the PRISM aggregation types [9] such as `book`, `journal`, `magazine`, `manual`, `report`, or `whitepaper`.

`pdfvolumenum` For publications in journals, magazines, and similar periodicals, a document can specify the volume number with `pdfvolumenum` and the issue number within the volume with `pdfissuenum`. `pdfpagerange` 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 `pdfpagerange={1,4-5}`. See Note 9 on page 17 for advice on how to assign `pdfpagerange` semi-automatically. A journal article’s publication status can be indicated with `pdfpubstatus`. 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.

`pdfpubstatus`

`pdfbookedition` For books, `pdfbookedition` names the edition of the book. This is specified as text, not a number. As with `pdfpublication` (above), `pdfbookedition` accepts a bracketed language code, as in `pdfbookedition={ [en] Second edition }`.

`pdfdate` XMP metadata can include a number of dates (in fact, timestamps, as they include both date and time components). `pdfdate` specifies the document date. It is analogous to the \LaTeX `\date` command, and, like `\date`, defaults to the date the document was built. It must be specified in either XMP format [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

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

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

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

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

Rarely needed options

pdfsource	pdfsource overrides the name of the L ^A T _E X source file. It defaults to <code>\jobname.tex</code> but can be replaced by any other string. If pdfsource is given an empty argument, no document source will be specified at all.
pdfnumpages	The number of pages in the published, print version of the document can be expressed with pdfnumpages. This is computed automatically when the document is built using either pdfL ^A T _E X or LuaL ^A T _E X.
pdfbytes	The pdfbytes option expresses the document's file size in bytes. The intention is for this to be used to display an estimate of download time to a user or to serve as a quick check on whether a file was transmitted correctly between systems. pdfbytes is computed automatically by both pdfL ^A T _E X and LuaL ^A T _E X, using the file size from the previous build of the document.

It is usually more convenient to provide values for all of the options presented in this section using hyperref's `\hypersetup` command than on the `\usepackage` command line. See the hyperref manual for more information.

2.2 A complete example

The following is a sample L^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},
pdfidentifier={info:lcnn/50013519}
}
\XMLLangAlt{de}{pdftitle={Über einen die Erzeugung und Verwandlung des
  Lichtes betreffenden heuristischen Gesichtspunkt}}

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

```

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

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

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

Once the document is compiled, the resulting PDF file will contain an XMP packet that looks something like that shown in Appendix A. Figure 2 is a screenshot of the XMP metadata as it appears in Adobe Acrobat's "Advanced" metadata dialog box. Further clicking on the "Advanced" item within that dialog box displays all of the document's metadata sorted by schema as shown in Figure 3.

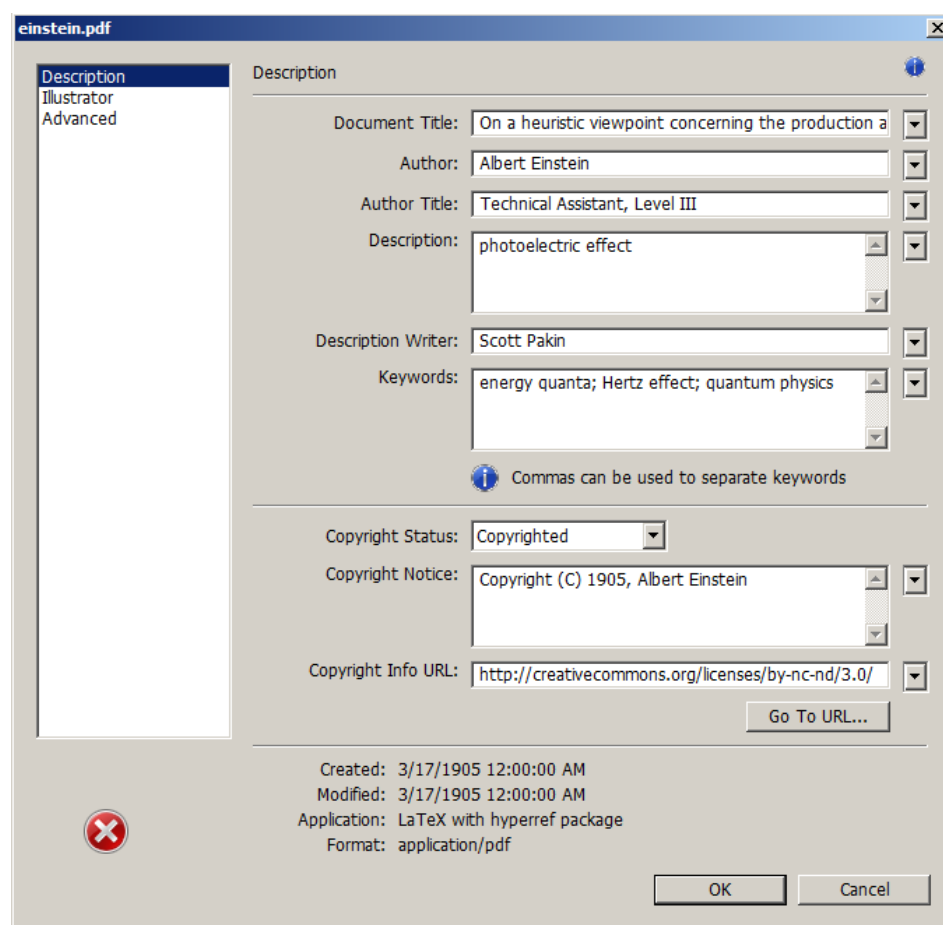


Figure 2: XMP metadata as it appears in Adobe Acrobat

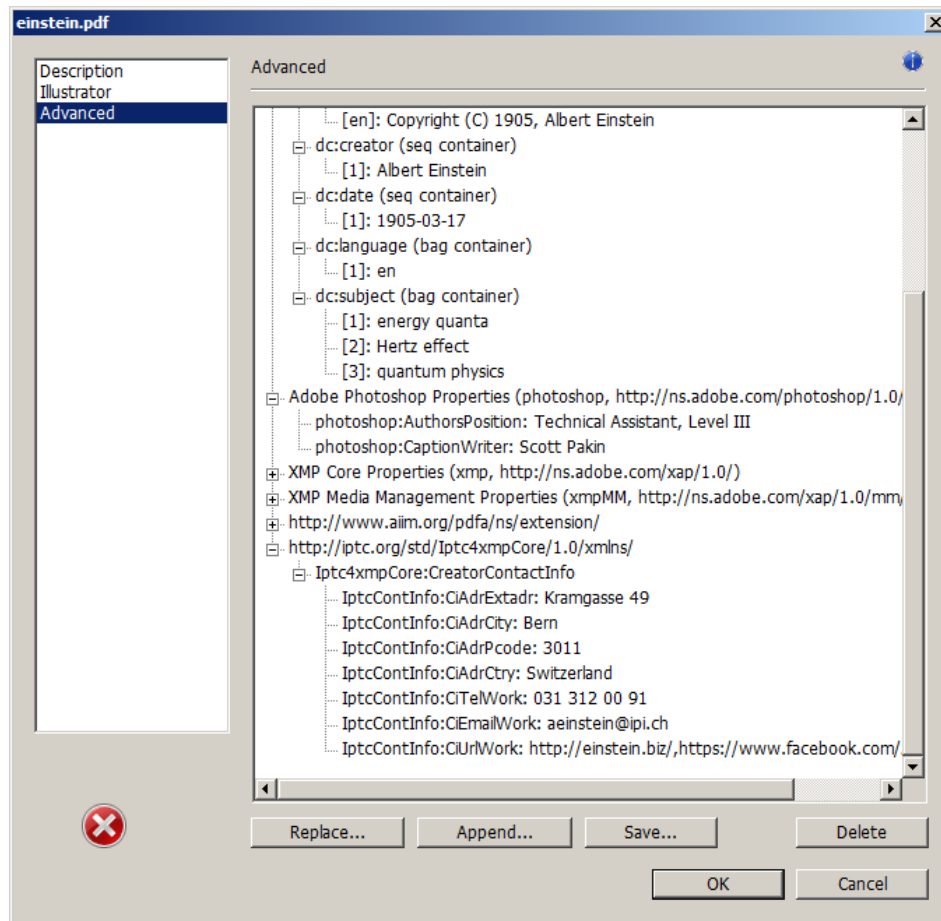


Figure 3: Additional XMP metadata as it appears in Adobe Acrobat

2.3 Usage notes

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

```

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

```

while in PDF, the author and keywords are specified as flat strings. Alas, there is no definition of how a list should be collapsed to a flat string: “Curly Howard, Larry Fine, Moe Howard” or “Curly Howard; Larry Fine; Moe Howard” or something else. I have not yet found a form of flat string that passes all PDF/A validators. Furthermore, when Adobe Acrobat—at least Adobe Acrobat DC (2019) and earlier versions—converts a PDF file to PDF/A format, it does so by discarding all but the first author, which is an unsatisfying solution.

Starting with version 4.0, `hyperxmp`’s solution is to suppress writing metadata to the PDF Info dictionary and write it only to the XMP packet. (`hyperxmp` v5.0+ is more sophisticated. It suppresses only the author and keyword lists.) This appears to pacify PDF/A validators yet retains the author and keyword lists in their non-truncated form. If desired, the Info dictionary can be retained by passing the `keeppdfinfo` option to `\hypersetup`.

`keeppdfinfo`

Note 2: Acrobat multiline-field bug The IPTC Photo Metadata schema states that “the [contact] address is a multiline field” [10]. `hyperxmp` converts commas in `pdfcontactaddress`’s argument to line breaks in the generated XML. Unfortunately, a bug in Adobe Acrobat—at least in Adobe Acrobat DC (2019) and earlier versions—causes that PDF reader to discard line breaks in the contact address. Interestingly, Adobe Illustrator CS5 correctly displays the contact address. If you find Adobe Acrobat’s behavior bothersome, you can redefine the `\xmplinesep` macro as a string to use as an address-line separator. For example, the following replaces all commas appearing in `pdfcontactaddress`’s argument with semicolons:

`\xmplinesep`

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

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

1. When run with versions of `LuaLATEX` earlier than 0.85, `hyperxmp` leaves all PDF objects uncompressed. This is due to `LuaLATEX` treating object compression as a global parameter, unlike `pdfLATEX`, which treats it as a local parameter. Hence, when `hyperxmp` requests that the XMP packet be left uncompressed, `LuaLATEX` in fact leaves *all* PDF streams uncompressed. Beginning with version 3.0, `hyperxmp` includes a workaround that correctly leaves only the

XMP metadata uncompressed, but this workaround is implemented only for Lua^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., Lua^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` If you need to include a literal comma within an author or keyword list (where
`\xmpquote` commas normally separate list items) or a street address (where commas normally
 separate lines), use the `\xmpcomma` macro to represent it, and wrap the entire entry
 containing the comma within `\xmpquote{...}` as shown below:

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

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

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

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

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

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

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

Note 6: Automatically specified metadata `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 `babel` or `polyglossia` package, it is recommended that it *not* explicitly set `pdflang`. `pdflang` accepts only a single language name while `hyperxmp` can automatically query `babel` and `polyglossia` for a list of all languages used in the document and include this list in an XMP `dc:language` element.

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

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

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

```
\hypersetup{%
```



```

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

```

Note 8: Expandable arguments All arguments passed to `hyperxmp` options must be expandable, in $\text{T}_{\text{E}}\text{X}$ 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=02020, Scott Pakin”. The garbage in that line corresponds to the remnants of the `\printfdate` code after expanding all of the $\text{T}_{\text{E}}\text{X}$ 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}}
  }%
}
```

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

`\hyxmp@dq@code` The `ngerman` package redefines “ ” as an active character, which causes problems for `hyperxmp` when it tries to use that character. We therefore save the double-quote character’s current category code in `\hyxmp@dq@code` and mark the character as category code 12 (“other”). The original category code is restored at the end of the package code (Section 3.8).

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

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

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

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

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

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

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

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

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

Define `\hyxmp@jobname` as a sanitized version of `\jobname`. The problem with using `\jobname` directly is that it surrounds the filename with double quotes if it contains a space character. For example, a source file named `my-file.tex` results in a `\jobname` of “my-file”, but a source file named `my file.tex` results in a `\jobname` of “my file”. Trying to access “my file”.log (as is done on page 48) will fail because the filename does not in fact contain literal double quotes.

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

`\hyxmp@aep@toks` In order for `hyperxmp` to be loaded safely during `\AtEndPreamble` we need to ensure that we perform no `\AtEndPreamble` actions until all top-level macro definitions have been made. The most straightforward approach would be to move all of `hyperxmp`’s `\AtEndPreamble` stanzas to the end of the package. However, this degrades readability of the source code. For instance, an `\AtEndPreamble` stanza related to integration with `hyperref` could no longer appear in the “Integration with `hyperref`” section (Section 3.2). Hence, we instead store in a token list, `\hyxmp@aep@toks`, each `\AtEndPreamble` stanza as we encounter it. This token list is evaluated as one of the package’s final actions (Section 3.8).

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

3.2 Integration with `hyperref`

An important design decision underlying `hyperxmp` is that the package should integrate seamlessly with `hyperref`. To that end, `hyperxmp` takes XMP metadata from `hyperref`’s `baseurl`, `pdfauthor`, `pdfkeywords`, `pdflang`, `pdfproducer`, `pdfsubject`, `pdftrapped`, and `pdftitle` options. It also introduces a number of new options, which are listed on pages 5–6. For consistency with `hyperref`’s document-metadata naming conventions (which are in turn based on L^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.

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

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

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

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

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

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

```

29 \newcommand*{\@if@def@and@nonempty}[3]{%
30   \@ifundefined{#1}{#3}{%
31     \expandafter\@ifmtargexp\expandafter{\csname#1\endcsname}{#3}{#2}%
32   }%
33 }

```

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

`\hyxmp@textunderscore`

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

55   \edef\next{%

```

```

56      \noexpand\hyxmp@pdfstringdef\noexpand\@pdfmetadatettime{%
57      \noexpand\hyxmp@as@xmp@date{#1}}}%
58    }%
59    \next
60  \endgroup
61 }

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

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

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

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

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

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

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

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

```

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

```

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

```

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

```

88 \def\@pdfuapart{}
89 \define@key{Hyp}{pdfuapart}{%
90   \afterassignment\hyxmp@no@bad@parts\@tempcnta=0#1\relax
91   \hyxmp@pdfstringdef\@pdfuapart{the\@tempcnta}%
92 }

```

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

```

93 \newcommand*\hyxmp@set@pdfx@major[1]{\hyxmp@set@pdfx@major@i#1!}

```

`\hyxmp@set@pdfx@major@i` This is the first helper macro for `\hyxmp@set@pdfx@major`. It stores the PDF/X major version in `\@tempcnta`.

```

94 \def\hyxmp@set@pdfx@major@i PDF/X-{%
95   \afterassignment\hyxmp@set@pdfx@major@ii
96   \@tempcnta=%
97 }

```

`\hyxmp@set@pdfx@major@ii` This is the second helper macro for `\hyxmp@set@pdfx@major`. It copies the PDF/X major version from `\@tempcnta` to `\@hyxmp@pdfx@major` and discards the rest of the PDF/X standard string.

```

98 \def\hyxmp@set@pdfx@major@ii#1!{%
99   \edef\hyxmp@pdfx@major{the\@tempcnta}%
100 }

```

`\hyxmp@check@std` Compare a user-provided string to a fixed string. (Assumption: Both are names of PDF/X standard versions.) If they match, undefine `\next`, which we assume was previously defined to issue an “unrecognized standard” warning message.

```

101 \newcommand*\hyxmp@check@std[2]{%
102   \ifthenelse{\equal{#1}{#2}}{%
103     {\global\let\next=\relax}%
104     {}%
105 }%

```

`\@pdfxstandard` Prepare to store the PDF/X standard.

```

106 \def\@pdfxstandard{}
107 \def\hyxmp@pdfx@major{}
108 \define@key{Hyp}{pdfxstandard}{%
109   \hyxmp@pdfstringdef\@pdfxstandard{#1}%

```

`\next` Issue a warning message if the PDF/X standard named by the user does not appear in a list of known PDF/X standards. This is to caution the user that `hyperxmp` generates standard-specific XMP metadata and it can only guess at the correct format for new standard versions. (See the comments on page 72 above the definition of `\hyxmp@pdfx@id@schema`, for example.)

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

`\next`


```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```

\@pdfcontacturl Prepare to store the URL of the document's contact person/institution.
205 \def\@pdfcontacturl{}
206 \define@key{Hyp}{pdfcontacturl}{\hyxmp@pdfstringdef\@pdfcontacturl{#1}}

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

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

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

```

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

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

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

```

\hyxmp@pdfauthor Prepare to store the name of the author and a list of keywords.
\hyxmp@pdfkeywords 233 \def\hyxmp@pdfauthor{}
234 \def\hyxmp@pdfkeywords{}

\hyxmp@redefine@Hyp If not already redefined, redefine hyperref's pdfauthor and pdfkeywords options to
properly handle \xmpcomma and \xmpquote.
235 \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.
236 \@ifundefined{KV@Hyp@pdfauthor}{}{%
237 \@ifundefined{hyxmp@Hyp@pdfauthor}{%
238 \expandafter\let\expandafter\hyxmp@Hyp@pdfauthor
239 \csname KV@Hyp@pdfauthor\endcsname
240 }{}%
241 }%

\KV@Hyp@pdfauthor Redefine \KV@Hyp@pdfauthor to process its argument twice. The first time,
\xmpcomma \xmpcomma is defined as a placeholder character (\hyxmp@comma) and \xmpquote
\xmpquote as the identity function. The result is stored in \hyxmp@pdfauthor for use in
\hyxmp@and structured lists (those surrounding each entry with <rdf:li>). The second time,
\xmpcomma is defined as an ordinary comma, and \xmpquote is defined as a macro
\and that puts its argument within double quotes. The result is stored in \@pdfauthor
\@pdfauthor for use in unstructured lists (those in which the entire list appears within a single
pair of tags). In case pdfauthor is left unspecified and we copy \author's argument
to pdfauthor, we temporarily redefine \and as the list separator when producing a
structured list and as "and" when producing an unstructured list.
242 \define@key{Hyp}{pdfauthor}{%
243 \let\xmpcomma=\hyxmp@comma
244 \def\xmpquote####1{####1}%
245 \let\hyxmp@and=\and
246 \def\and{,}%
247 \hyxmp@Hyp@pdfauthor{##1}%
248 \global\let\hyxmp@pdfauthor=\@pdfauthor
249 \def\and{and\space}%
250 \def\xmpcomma{,}%
251 \def\xmpquote####1{"####1"%
252 \hyxmp@Hyp@pdfauthor{##1}%
253 \def\xmpcomma{,}%
254 \let\xmpquote=\relax
255 \let\and=\hyxmp@and

```

256 }%

`\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`.

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

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

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

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

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

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

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

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

```

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

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

```

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

```

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

```

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

```

350 \newcommand*{\hyxmp@check@standards}{%

```

If the pdfa option was passed to hyperref but `\pdfapart` is not set, set it to 1 and `\pdfaconformance` to B.

```

351 \ifHy@pdfa
352 \@ifmtargexp{\pdfapart}{%
353 \PackageWarningNoLine{hyperxmp}{%
354 'pdfa' was passed to hyperref, but 'pdfapart' was\MessageBreak
355 not specified.\space\space Setting pdfapart to '1' and\MessageBreak
356 pdfaconformance to 'B'%
357 }%
358 \gdef\pdfapart{1}%
359 \gdef\pdfaconformance{B}%
360 }%
361 {}%
362 \fi

```

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

```

363 \edef\hyxmp@standards{%

```



```

364     \@pdfapart
365     \@pdfxstandard
366     \@pdfuapart
367 }%

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

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

```

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

```

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

```

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

```

382     \hyxmp@no@info@lists

```

If `pdftitle` is undefined but the author invoked `\title`, we copy the latter to the former. This addresses two problems: (1) handling 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.

```

383     \begingroup
384         \let\@latex@warning@no@line=\@gobble
385         \hyxmp@use@first@valid{pdftitle}{\@pdftitle}{%

```

```

386         \scr@subject@var,%
387         \@title
388     }%
389     \hyxmp@use@first@valid{pdfauthor}{\@pdfauthor}{%
390         \scr@fromname@var,%
391         \@author
392     }%
393 \endgroup
394 }%
395 }

```

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

```

396 \hyxmp@at@end{%

```

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

```

397 \hyxmp@auto@assign@data

```

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

```

398 \hyxmp@check@standards

```

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

```

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

```

3.3 Advanced metadata detection

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

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

`\hyxmp@set@koma@phones` Define `\hyxmp@koma@phones` as a comma-separated list of the phone numbers provided to a KOMA letter class (mobile and landline).

```

402 \newcommand*{\hyxmp@set@koma@phones}{%
403 \begingroup
404     \Hy@unicodedefalse
405     \@if@def@and@nonempty{scr@frommobilephone@var}{%
406         \@if@def@and@nonempty{scr@fromphone@var}{%

```

```

407     \hyxmp@pdfstringdef\hyxmp@koma@phones{\scr@frommobilephone@var,\scr@fromphone@var}%
408   }{%
409     \hyxmp@pdfstringdef\hyxmp@koma@phones{\scr@frommobilephone@var}%
410   }%
411 }{%
412   \@if@def@and@nonempty{\scr@fromphone@var}{%
413     \hyxmp@pdfstringdef\hyxmp@koma@phones{\scr@fromphone@var}%
414   }{%
415   }%
416 }%
417 \endgroup
418 }

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

441 \if@def@and@nonempty{@pdflang}{%
442   \let\hyxmp@dc@lang=\pdflang
443 }{%
444   \hyxmp@detect@langs
445 }%

```

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

```

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

```

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

```

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

```

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

```

469 \hyxmp@parse@acmart

```

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

```

470 \ifundefined{XeTeXversion}{}{%

```

```

471 \ifmtargexp{\@pdfcreationdate}{%
472 \let\@pdfcreationdate=\hyxmp@today@pdf
473 }%
474 {}%
475 }%

```

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

```

476 \hyxmp@query@self
477 }

```

`hyperxmp` can directly query the page count using `LuaTeX` features. When any other `TeX` engine is used, `hyperxmp` employs the `totpages` package to help tally the total number of pages.

```

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

```

Determine the size of the output file from the *previous* run of `LuaLaTeX` or `pdfLaTeX`. 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 even in the `LuaLaTeX` case because `status.output_file_name` is not defined at this point.

```

482 \ifLuaTeX
483 \edef\hyxmp@prev@pdf@size{%
484 \luadirect{%
485 local nbytes = lfs.attributes("\hyxmp@jobname.pdf", "size")
486 if nbytes then
487 tex.write(nbytes)
488 end
489 }%
490 }%
491 \else
492 \ifPDFTeX
493 \edef\hyxmp@prev@pdf@size{\pdffilesize{\hyxmp@jobname.pdf}}%
494 \fi
495 \fi

```

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

```

496 \newcommand*\hyxmp@query@self{%

```

`LuaTeX` exposes via `status.total_pages` the number of pages written. We use this mechanism when available to assign `\@pdfnumpages`. To finalize the page count we first issue a `\clearpage`.

```

497 \ifLuaTeX
498 \if@def@and@nonempty{@pdfnumpages}{%
499 }%
500 \clearpage
501 \xdef\@pdfnumpages{\luadirect{tex.print(status.total_pages)}}%
502 }%
503 \else

```

Without Lua_T_EX we rely on the `totpages` package to help count the number of pages. This may require an additional run of L^AT_EX, but the user will be notified in that case.

```
504 \pdfstringdef\@pdfnumpages{\ref*{TotPages}}%
505 \fi
```

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

```
506 \hyxmp@use@first@valid{pdfbytes}{\@pdfbytes}{%
507 \hyxmp@prev@pdf@size
508 }%
509 }
```

`\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.

```
510 \newcommand*{\hyxmp@parse@acmart}{%
511 \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.

```
512 \let\@author=\@gobble
```

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

```
513 \def\email##1##2{%
514 \def\hyxmp@address@val{##2}%
515 \hyxmp@use@first@valid{pdfcontactemail}{\@pdfcontactemail}{%
516 \hyxmp@address@val
517 }%
518 }%
```

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

```
\hyxmp@address@val 519 \def\streetaddress##1{%
520 \def\hyxmp@address@val{##1}%
521 \hyxmp@use@first@valid{pdfcontactaddress}{\@pdfcontactaddress}{%
522 \hyxmp@address@val
523 }%
524 }%
```

`\city` `\city` wraps the author's city name.

```
\hyxmp@address@val 525 \def\city##1{%
526 \def\hyxmp@address@val{##1}%
527 \hyxmp@use@first@valid{pdfcontactcity}{\@pdfcontactcity}{%
528 \hyxmp@address@val
529 }%
530 }%
```

`\state` `\state` wraps the author's state or region name.

```

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

```

`\country` `\country` wraps the author's country name.

```

\hyxmp@address@val 537    \def\country##1{%
                    538    \def\hyxmp@address@val{##1}%
                    539    \hyxmp@use@first@valid{pdfcontactcountry}{\@pdfcontactcountry}{%
                    540    \hyxmp@address@val
                    541    }%
                    542    }%

```

`\postcode` `\postcode` wraps the author's postal code.

```

\hyxmp@address@val 543    \def\postcode##1{%
                    544    \def\hyxmp@address@val{##1}%
                    545    \hyxmp@use@first@valid{pdfcontactpostcode}{\@pdfcontactpostcode}{%
                    546    \hyxmp@address@val
                    547    }%
                    548    }%

```

`\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.

```

549    \def\affiliation##1##2{%
550    ##2%
551    \let\affiliation=\@gobbletwo
552    }%

```

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.

```

553    \setbox0=\hbox{\addresses}%
554    \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.

```

555    \@if@def@and@nonempty{\acmDOI}{%

```

```

556 \IfSubStr{\@acmDOI}{10.1145/1122445.1122456}{-}{%
557 \IfSubStr{\@acmDOI}{10.1145/nnnnnnn.nnnnnnn}{-}{%
558 \hyxmp@use@first@valid{pdfdoi}{\@pdfdoi}{%
559 \@acmDOI
560 }%
561 }%
562 }%
563 }%
564 {}%

```

`\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.

```

565 \@if@def@and@nonempty{\@acmISBN}{%
566 \IfSubStr{\@acmISBN}{XXXX}{-}{%
567 \IfSubStr{\@acmISBN}{xxxx}{-}{%
568 \def\hyxmp@strip@isbn@date##1/##2!{##1}%
569 \edef\hyxmp@acm@isbn{%
570 \expandafter\hyxmp@strip@isbn@date\@acmISBN/!%
571 }%
572 \hyxmp@use@first@valid{pdfisbn}{\@pdfisbn}{%
573 \hyxmp@acm@isbn
574 }%
575 }%
576 }%
577 }%
578 {}%

```

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

```

579 \def\hyxmp@acm@publisher{Association for Computing Machinery}%
580 \hyxmp@use@first@valid{pdfpublisher}{\@pdfpublisher}{%
581 \hyxmp@acm@publisher
582 }%

```

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

```

583 \hyxmp@use@first@valid{pdfpublication}{\@pdfpublication}{%
584 \@journalName,%
585 \@acmBooktitle,%
586 \@acmConference
587 }%

```

`\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).

```

588 \if@ACM@journal
589 \def\hyxmp@acm@pubtype{journal}%
590 \else
591 \def\hyxmp@acm@pubtype{book}%

```



```

592 \fi
593 \hyxmp@use@first@valid{pdfpubtype}{\@pdfpubtype}{%
594 \hyxmp@acm@pubtype
595 }%

```

Journal articles have a volume and issue number.

```

596 \hyxmp@use@first@valid{pdfvolumenum}{\@pdfvolumenum}{%
597 \@acmVolume
598 }%
599 \hyxmp@use@first@valid{pdfissuenum}{\@pdfissuenum}{%
600 \@acmNumber
601 }%
602 }

```

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

```

603 \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.

```

604 \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.

```

605 \newcommand*{\hyxmp@detect@langs}{%
606 \ifundefined{mainbcp47id}{%
607 \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.

```

608 \let\@pdflang=\@empty
609 }%

```

`\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
610 \BabelEnsureInfo
611 \LocaleForEach{%
612 \getlocaleproperty\hyxmp@lang@tag{##1}{identification/tag.bcp47}%
613 \ifx\hyxmp@dc@lang\@empty
614 \xdef\hyxmp@dc@lang{\hyxmp@lang@tag}%
615 \else
616 \xdef\hyxmp@dc@lang{\hyxmp@dc@lang,\hyxmp@lang@tag}%
617 \fi
618 \def\hyxmp@lang@name{##1}%

```

```

619         \ifx\hyxmp@lang@name\bb1@main@language
620             \edef\@pdflang{\hyxmp@lang@tag}%
621         \fi
622     }%
623 }%
624 }{%
```

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

```

625     \xdef\@pdflang{\csname mainbcp47id\endcsname}%
626     \edef\hyxmp@dc@lang{\xpg@bcp@loaded}%
627 }%
628 }
```

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.)

629 \newcommand*{\hyxmp@commas@to@list}[2]{%
630     \gdef#1{%
631         \expandafter\hyxmp@commas@to@list@i\expandafter#1#2,,%
632     }

\hyxmp@commas@to@list@i Recursively construct macro #1 from comma-separated list #2. Stop if #2 is empty.
\next
633 \def\hyxmp@commas@to@list@i#1#2,{%
634     \gdef\hyxmp@sublist{#2}%
635     \ifx\hyxmp@sublist@empty
636         \let\next=\relax
637     \else
638         \hyxmp@trimspaces\hyxmp@sublist
639         \@cons{#1}{\hyxmp@sublist}%
640     \def\next{\hyxmp@commas@to@list@i{#1}}%
```

```

641 \fi
642 \next
643 }

\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.
644 \def\xmpcomma{,}%

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

645 \bgroup
646 \catcode'^^C=11
647 \gdef\hyxmp@comma{^^C}
648 \egroup

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

649 \bgroup
650 \catcode'^^U=11
651 \gdef\hyxmp@uscore{^^U}
652 \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.
653 \let\xmpquote=\relax

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

654 \bgroup
655 \catcode'\~=12%
656 \gdef\xmptilde{~}%
657 \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.

```

```

658 \newcommand{\XMPTruncateList}[1]{%
659   \PackageWarning{hyperxmp}{%
660     \noexpand\XMPTruncateList has been deprecated since\MessageBreak
661     hyperxmp 4.0 and may be removed in future\MessageBreak
662     versions of the package. \noexpand\XMPTruncateList\MessageBreak
663     was found}%
664   \edef\hyxmp@temp@str{\csname hyxmp@#1\endcsname}%
665   \hyxmp@commas@to@list{\hyxmp@temp@list}{\hyxmp@temp@str}%
666   \def\@elt##1{%
667     \expandafter\gdef\csname @#1\endcsname{##1}%
668     \let\@elt=\@gobble
669   }
670   \hyxmp@temp@list
671 }

```

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:20201005202728-06’00’) [4], while XMP timestamps are of the form “YYYY-MM-DDThh:mm:ss+TT:tt” (e.g., 2020-10-05T20:27:28-06:00) [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 672 \def\hyxmp@first@char#1{\hyxmp@first@char@i#1\relax}
673 \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.

```

674 \def\hyxmp@as@xmp@date#1{%
675   \expandafter\ifnum\expandafter'\hyxmp@first@char@i#1\relax='D
676   \hyxmp@pdf@to@xmp@date{#1}%
677   \else
678     #1%
679   \fi
680 }

```

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

```

681 \def\hyxmp@pdf@to@xmp@date#1:#2#3#4#5#6#7#8#9{%
682   #2#3#4#5-#6#7-#8#9%
683   \hyxmp@parse@time
684 }

```

`\hyxmp@parse@time` This is a helper function for `\hyxmp@pdf@to@xmp@date`. `\hyxmp@pdf@to@xmp@date` proper parses only the year, month, and day

then calls `\hyxmp@parse@time`. `\hyxmp@parse@time` parses the hours, minutes, and seconds then calls `\hyxmp@parse@tz@char`.

```
685 \def\hyxmp@parse@time#1#2#3#4#5#6{%
686   T#1#2:#3#4:#5#6%
687   \hyxmp@parse@tz@char
688 }
```

`\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.

```
689 \def\hyxmp@parse@tz@char#1{%
690   #1%
691   \ifx#1-%
692     \expandafter\hyxmp@parse@tz
693   \else
694     \ifx#1+%
695       \expandafter\hyxmp@parse@tz
696     \fi
697   \fi
698 }
```

`\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.

```
699 \def\hyxmp@parse@tz#1'#2' {%
700   #1:#2%
701 }
```

`\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.

```
702 \def\hyxmp@as@pdf@date#1{%
703   \expandafter\ifx\hyxmp@first@char@i#1\relax D%
704   #1%
705   \else
706     \hyxmp@xmp@to@pdf@date{#1}%
707   \fi
708 }
```

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

```
709 \def\hyxmp@xmp@to@pdf@date#1{%
710   D:\hyxmp@xmp@to@pdf@date@i#1\relax\relax
711 }
```

```

\hyxmp@xmp@to@pdf@date@i Parse the year for \hyxmp@xmp@to@pdf@date.
712 \def\hyxmp@xmp@to@pdf@date@i#1#2#3#4#5#6{%
713   #1#2#3#4%
714   \ifx#5-%
715     \expandafter\hyxmp@xmp@to@pdf@date@ii\expandafter#6%
716   \fi
717 }

\hyxmp@xmp@to@pdf@date@ii Parse the month for \hyxmp@xmp@to@pdf@date.
718 \def\hyxmp@xmp@to@pdf@date@ii#1#2#3#4{%
719   #1#2%
720   \ifx#3-%
721     \expandafter\hyxmp@xmp@to@pdf@date@iii\expandafter#4%
722   \fi
723 }

\hyxmp@xmp@to@pdf@date@iii Parse the day for \hyxmp@xmp@to@pdf@date.
724 \def\hyxmp@xmp@to@pdf@date@iii#1#2#3#4{%
725   #1#2%
726   \ifx#3T%
727     \expandafter\hyxmp@xmp@to@pdf@date@iv\expandafter#4%
728   \fi
729 }

\hyxmp@xmp@to@pdf@date@iv Parse the hour for \hyxmp@xmp@to@pdf@date.
730 \def\hyxmp@xmp@to@pdf@date@iv#1#2#3#4{%
731   #1#2%
732   \ifx#3:%
733     \expandafter\hyxmp@xmp@to@pdf@date@v\expandafter#4%
734   \fi
735 }

\hyxmp@xmp@to@pdf@date@v Parse the minute for \hyxmp@xmp@to@pdf@date.
736 \def\hyxmp@xmp@to@pdf@date@v#1#2#3#4{%
737   #1#2%
738   \ifx#3:%
739     \expandafter\hyxmp@xmp@to@pdf@date@vi\expandafter#4%
740   \fi
741 }

\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.
742 \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).

```

```

743 \def\hyxmp@xmp@to@pdf@date@vi#1#2#3#4{%
744   #1#2%
745   \ifx#3+%
746     +\expandafter\hyxmp@xmp@to@pdf@date@vii
747   \fi
748   \ifx#3-%
749     -\expandafter\hyxmp@xmp@to@pdf@date@vii
750   \fi
751   \ifx#3Z%
752     Z%
753   \fi
754   \ifx#3\relax
755     \expandafter\hyxmp@gobbletwo
756   \fi
757   \@gobbletwo #4%
758 }

```

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

```

759 \def\hyxmp@xmp@to@pdf@date@vii#1\@gobbletwo#2#3#4#5{%
760   #2#3%
761   \ifx#4:%
762     \expandafter\hyxmp@xmp@to@pdf@date@viii\expandafter#5%
763   \fi
764 }

```

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

```

765 \def\hyxmp@xmp@to@pdf@date@viii#1#2#3#4{%
766   '#1#2'%
767 }

```

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

```

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

```

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

```

769   \xdef#1{\the\year}%
770   \ifnum\month<10
771     \xdef#1{#1-0\the\month}%
772   \else
773     \xdef#1{#1-\the\month}%
774   \fi
775   \ifnum\day<10
776     \xdef#1{#1-0\the\day}%
777   \else
778     \xdef#1{#1-\the\day}%
779   \fi

```

T_EX does not provide the time in terms of separate hours and minutes but rather as the total number of minutes since midnight (\time). There's no mechanism in

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

```

780 \@tempcnta=\time
781 \divide\@tempcnta by 60
782 \ifnum\@tempcnta<10
783   \xdef#1{#1T0\the\@tempcnta}%
784 \else
785   \xdef#1{#1T\the\@tempcnta}%
786 \fi
787 \multiply\@tempcnta by -60
788 \advance\@tempcnta by \time
789 \ifnum\@tempcnta<10
790   \xdef#1{#1:0\the\@tempcnta}%
791 \else
792   \xdef#1{#1:\the\@tempcnta}%
793 \fi
794 \xdef#1{#1Z}%
795 }

```

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

```

796 \def\hyxmp@try@today#1#2{%
797   \@ifmtargexp{\hyxmp@today@xmp}{%
798     \@ifundefined{#1}{#2}%
799   }%
800   {}%
801 }

```

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

```

802 \def\hyxmp@today@xmp{}
      Case 1: \pdfcreationdate is defined (pdfLaTeX and pre-0.85 LuaLaTeX).
803 \hyxmp@try@today{\pdfcreationdate}{%
804   \edef\hyxmp@today@xmp{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
805 }
      Case 2: \pdffeedback is defined (LuaLaTeX 0.85+).
806 \hyxmp@try@today{\pdffeedback}{%
807   \edef\hyxmp@today@xmp{\expandafter\hyxmp@pdf@to@xmp@date\pdffeedback creationdate}%
808 }

```

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

```

809 \hyxmp@try@today{\filemoddate}{%
810   \edef\hyxmp@today@xmp{\filemoddate{\hyxmp@jobname.log}}%
811   \edef\next{%
812     \edef\noexpand\hyxmp@today@xmp{\noexpand\hyxmp@as@xmp@date{\hyxmp@today@xmp}}%
813   }%
814   \next

```



```

815 }%
      Case 4: None of the above. Do the best we can using the available TEX primitives
      (\year, \month, \day, and \time.
816 \hyxmp@try@today{year}{%
817   \hyxmp@today@xmp@define\hyxmp@today@xmp
818 }

```

```

\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.
819 \expandafter\edef\expandafter\hyxmp@today@pdf\expandafter{%
820   \expandafter\hyxmp@xmp@to@pdf@date\expandafter{\hyxmp@today@xmp}%
821 }

```

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.

```

```

822 \catcode'\Q=3
      \hyxmp@trimspaces\x redefines \x to have the same replacement text sans leading
      and trailing space tokens.
823 \newcommand{\hyxmp@trimspaces}[1]{%
      Use grouping to emulate a multi-token afterassignment queue.
824   \begingroup
      Put “\toks 0 {” into the afterassignment queue.
825   \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.
826   \expandafter\hyxmp@trimb\expandafter\noexpand#1Q Q}%
      Transfer the trimmed text back into #1.
827   \edef#1{\the\toks0}%
828 }

```

```

\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.
829 \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.

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

```
832 \newif\ifhyxmp@unicodetex
833 \ifnum64='^^^^0040\relax
834   \hyxmp@unicodetextrue
835 \else
836   \hyxmp@unicodetexfalse
837 \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.

```
838 \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.

```
839 \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 but with all occurrences of “<” replaced with `<`; all occurrences of “>” replaced with `>`; and all occurrences of “&” replaced with `&`;

```
840 \newcommand*{\hyxmp@xmlify}[1]{%
841   \gdef\hyxmp@xmlified{%
      Escaped PDF string → PDFDocEncoding/Unicode
842   \EdefUnescapeString\hyxmp@text{#1}%
843   \ifhyxmp@unicodetex
```

PDFDocEncoding/Unicode → UTF-32BE

```

844 \hyxmp@is@unicode\hyxmp@text{%
845 \StringEncodingConvert
846 \hyxmp@text\hyxmp@text{utf16be}{utf32be}%
847 }{%
848 \ifXeTeX
849 \hyxmp@xetex@crap
850 \else
851 \StringEncodingConvert
852 \hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
853 \fi
854 }%

```

UTF-32BE → UTF-32BE as hex string

```

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

```

UTF-32BE → XML in ASCII

```

856 \edef\hyxmp@text{%
857 \expandafter
858 }\expandafter\hyxmp@toxml@unicodetex\hyxmp@text
859 \relax\relax\relax\relax\relax\relax\relax\relax
860 \else

```

PDFDocEncoding/Unicode → UTF-8

```

861 \hyxmp@is@unicode\hyxmp@text{%
862 \StringEncodingConvert
863 \hyxmp@text\hyxmp@text{utf16be}{utf8}%
864 }{%
865 \StringEncodingConvert
866 \hyxmp@text\hyxmp@text{pdfdoc}{utf8}%
867 }%

```

UTF-8 → UTF-8 as hex string

```

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

```

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

```

869 \edef\hyxmp@text{%
870 \expandafter\hyxmp@toxml\hyxmp@text\@empty\@empty
871 }%

```

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

```

872 \EdefUnescapeHex\hyxmp@text{\hyxmp@text}%
873 \fi
874 \global\let\hyxmp@xmlified\hyxmp@text
875 }

```

\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.

```

876 \begingroup
877 \lccode'\<=254 %
878 \lccode'\>=255 %
879 \catcode254=12 %

```

```

880 \catcode255=12 %
881 \lowercase{\endgroup
882 \def\hyxmp@is@unicode#1{%
883   \expandafter\hyxmp@is@unicode#1<>\@nil
884 }%
885 \def\hyxmp@@is@unicode#1<>#2\@nil{%
886   \ifx\#1\%
887     \expandafter\@firstoftwo
888   \else
889     \expandafter\@secondoftwo
890   \fi
891 }%
892 }

```

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

```

893 \def\hyxmp@toxml#1#2{%
894   \ifx#1\@empty
895   \else
896     \ifnum"#1#2='& %
897       26616D703B% &
898     \else\ifnum"#1#2='< %
899       266C743B% <
900     \else\ifnum"#1#2='> %
901       2667743B% >
902     \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.

```

903   \@ifundefined{pdfmark}{%
904     #1#2%
905   }{%
906     \ifnum"#1#2='( %
907       5C28% \(
908     \else\ifnum"#1#2='\) %
909       5C29% \)
910     \else
911       #1#2%
912     \fi\fi
913   }%

```

```

914     \fi\fi\fi
915     \expandafter\hyxmp@toxml
916   \fi
917 }

\hyxmp@toxml@unicodetex Replace the characters “<”, “&”, and “>” with XML entities when using a native-
\hyxmp@text Unicode TEX (XƎTEX or LuaTEX).
918 \def\hyxmp@toxml@unicodetex#1#2#3#4#5#6#7#8{%
919   \ifx#1\relax
920   \else
921     \ifnum"#1#2#3#4#5#6#7#8>127 %
922       \uccode'\*="#1#2#3#4#5#6#7#8\relax
923       \uppercase{%
924         \edef\hyxmp@text{\hyxmp@text *}%
925       }%
926     \else\ifnum"#7#8='< %
927       \edef\hyxmp@text{\hyxmp@text &lt;}%
928     \else\ifnum"#7#8='& %
929       \edef\hyxmp@text{\hyxmp@text &amp;%
930     \else\ifnum"#7#8='> %
931       \edef\hyxmp@text{\hyxmp@text &gt;}%
932     \else\ifnum"#7#8='\ %
933       \edef\hyxmp@text{\hyxmp@text\space}%
934     \else
935       \uccode'\*="#7#8\relax
936       \uppercase{%
937         \edef\hyxmp@text{\hyxmp@text *}%
938       }%
939     \fi\fi\fi\fi\fi
940     \expandafter\hyxmp@toxml@unicodetex
941   \fi
942 }

\hyxmp@skipzeros Skip over leading zeroes in the input argument.
943 \def\hyxmp@skipzeros#1{%
944   \ifx#10%
945     \expandafter\hyxmp@skipzeros
946   \fi
947 }

\hyxmp@xetex@crap \x In the case of XƎTEX, the strings defined by \pdfstringdef can contain big
\hyxmp@text characters. In this case, the string is treated as Unicode.
\hyxmp@try 948 \begingroup
\hyxmp@crap@result 949 \def\x#1{\endgroup
\hyxmp@text 950 \def\hyxmp@xetex@crap{%
951   \edef\hyxmp@try{%
952     \expandafter\hyxmp@SpaceOther\hyxmp@text#1\@nil
953   }%
954   \let\hyxmp@crap@result=N%

```

```

955 \expandafter\hyxmp@crap@test\hyxmp@try\relax
956 \ifx\hyxmp@crap@result Y%
957 \let\hyxmp@text\@empty
958 \expandafter\hyxmp@crap@convert\hyxmp@try\relax
959 \else
960 \StringEncodingConvert\hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
961 \fi
962 }%
963 }
964 \x{ }

```

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

```

965 \begingroup
966 \catcode'\~=12 %
967 \lccode'\~=\' %
968 \lowercase{\endgroup
969 \def\hyxmp@SpaceOther#1 #2\@nil{%
970 #1%
971 \ifx\relax#2\relax
972 \expandafter\@gobble
973 \else
974 ~%
975 \expandafter\@firstofone
976 \fi
977 {\hyxmp@SpaceOther#2\@nil}%
978 }%
979 }

```

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

```

980 \def\hyxmp@crap@test#1{%
981 \ifx#1\relax
982 \else
983 \ifnum'#1>127 %
984 \let\hyxmp@crap@result=Y%
985 \expandafter\expandafter\expandafter\hyxmp@skiptorelax
986 \else
987 \expandafter\expandafter\expandafter\hyxmp@crap@test
988 \fi
989 \fi
990 }

```

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

```

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

```

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

```

\hyxmp@num 992 \def\hyxmp@crap@convert#1{%
\hyxmp@text 993 \ifx#1\relax
994 \else
995 \edef\hyxmp@num{\number'#1}%

```

```

996     \ifnum\hyxmp@num>"FFFFFF %
997         \lccode'\!=\intcalcDiv{\hyxmp@num}{\number"1000000}\relax
998         \lowercase{\edef\hyxmp@text{\hyxmp@text!}}}%
999         \edef\hyxmp@num{\intcalcMod{\hyxmp@num}{\number"1000000}}}%
1000     \else
1001         \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
1002     \fi
1003     \ifnum\hyxmp@num>"FFFF %
1004         \lccode'\!=\intcalcDiv{\hyxmp@num}{\number"10000}\relax
1005         \lowercase{\edef\hyxmp@text{\hyxmp@text!}}}%
1006         \edef\hyxmp@num{\intcalcMod{\hyxmp@num}{\number"10000}}}%
1007     \else
1008         \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
1009     \fi
1010     \ifnum\hyxmp@num>"FF %
1011         \lccode'\!=\intcalcDiv{\hyxmp@num}{\number"100}\relax
1012         \lowercase{\edef\hyxmp@text{\hyxmp@text!}}}%
1013         \edef\hyxmp@num{\intcalcMod{\hyxmp@num}{\number"100}}}%
1014     \else
1015         \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
1016     \fi
1017     \ifnum\hyxmp@num>0 %
1018         \lccode'\!=\hyxmp@num\relax
1019         \lowercase{\edef\hyxmp@text{\hyxmp@text!}}}%
1020     \else
1021         \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
1022     \fi
1023     \expandafter\hyxmp@crap@convert
1024 \fi
1025 }

```

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

```

1026 \begingroup
1027   \catcode0=12 %
1028   \gdef\hyxmp@zero{^^00}%
1029 \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.

```

1030 \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.

```

1031 \newcommand*{\hyxmp@add@simple}[2]{%
1032   \ifnotmtargexp{#2}{%
1033     \hyxmp@xmllify{#2}%
1034     \hyxmp@add@to@xml{\hyxmp@extra@indent_____<}%
1035     \xdef\hyxmp@xml{\hyxmp@xml#1}%
1036     \hyxmp@add@to@xml{>\hyxmp@xmllified</>%
1037     \xdef\hyxmp@xml{\hyxmp@xml#1>^^J}%
1038   }%
1039 }
```

`\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.

```

1040 \newcommand*{\hyxmp@add@simple@var}[2]{%
1041   \expandafter\ifx\csname#2\endcsname\relax
1042   \else
1043     \hyxmp@xmllify{\csname#2\endcsname}%
1044     \hyxmp@add@to@xml{%
1045       \hyxmp@extra@indent_____<#1>\hyxmp@xmllified</#1>^^J%
1046     }%
1047   \fi
1048 }
```

`\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.

```

1049 \newcommand*{\hyxmp@add@simple@lang}[2]{%
1050   \ifnotmtarg{#2}{%
1051     \hyxmp@xmllify{#2}%
1052     \expandafter\hyxmp@add@simple@lang@i\hyxmp@xmllified\relax{#1}%
1053   }%
1054 }
```

`\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.

```

1055 \newcommand*{\hyxmp@add@simple@lang@i}{%
1056   \ifnextchar[\hyxmp@add@simple@lang@ii{\hyxmp@add@simple@lang@ii[\pdfmetalang]}%
1057 }
```


`\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.

```

1058 \def\hyxmp@add@simple@lang@ii[#1]#2\relax#3{%
1059   \ifnotmtarg{#2}{%
1060     \hyxmp+xmlify{#2}%
1061     \ifmtarg{#1}{%
1062       \hyxmp@add@to+xml{%
1063         <#3>\hyxmp+xmlified</#3>^^J%
1064       }%
1065     }%
1066     \hyxmp@add@to+xml{%
1067       <#3 xml:lang="#1">\hyxmp+xmlified</#3>^^J%
1068     }%
1069   }%
1070 }%
1071 }

```

`\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.

```

1072 \newcommand*{\hyxmp@add@simple@pfx}[3]{%
1073   \ifnotmtargexp{#3}{%
1074     \hyxmp@add@to+xml{\hyxmp@extra@indent_____<}%
1075     \xdef\hyxmp+xml{\hyxmp+xml#1}%
1076     \hyxmp@pdfstringdef\hyxmp@iprefix{#2}%
1077     \hyxmp+xmlify{\hyxmp@iprefix}%
1078     \hyxmp@add@to+xml{>\hyxmp+xmlified}%
1079     \hyxmp+xmlify{#3}%
1080     \hyxmp@add@to+xml{\hyxmp+xmlified</}%
1081     \xdef\hyxmp+xml{\hyxmp+xml#1>^^J}%
1082   }%
1083 }

```

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}`”.

```

1084 \def\hyxmp@alt@title{}

```

```

1085 \def\hyxmp@alt@description{}
1086 \def\hyxmp@alt@rights{}

\hyxmp@LA@accept This macro wraps \define@key to make the option “#1=⟨value⟩” append ⟨value⟩
to list #2.
1087 \newcommand{\hyxmp@LA@accept}[2]{%
1088   \define@key{hyxmp@LA}{#1}{%

\hyxmp@value As Niklas Beisert observed, if the option passed to the current key contains LATEX
code, this code will be included in the XMP packet, which is undesirable. Hence,
we first clean up the string using \hyxmp@pdfstringdef.
1089   \hyxmp@pdfstringdef\hyxmp@value{##1}%
1090   \xdef#2{#2\noexpand\do {\hyxmp@cur@lang} {\hyxmp@value}}%
1091 }
1092 }

Define ⟨key⟩=⟨value⟩ options for appending to each of the \hyxmp@alt⟨tag⟩
lists.
1093 \hyxmp@LA@accept{pdftitle}{\hyxmp@alt@title}
1094 \hyxmp@LA@accept{pdfsubject}{\hyxmp@alt@description}
1095 \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.
1096 \newcommand{\XMPLangAlt}[2]{%
1097   \let\do=\relax

\hyxmp@cur@lang Store the provided language, which will be used during option processing.
1098   \edef\hyxmp@cur@lang{#1}%
1099   \setkeys{hyxmp@LA}{#2}%
1100 }

```

3.5 UUID generation

We use a linear congruential generator to produce pseudorandom version 4 UUIDs [12]. True, this method has its flaws but it’s simple to implement in T_EX and is good enough for producing the XMP xmpMM:DocumentID and xmpMM:InstanceID fields.

```

\hyxmp@modulo@a Replace the contents of \@tempcnta with the contents modulo #1. Note that
\@tempcntb is overwritten in the process.
1101 \def\hyxmp@modulo@a#1{%
1102   \@tempcntb=\@tempcnta
1103   \divide\@tempcntb by #1
1104   \multiply\@tempcntb by #1

```

```

1105 \advance\@tempcnta by -\@tempcntb
1106 }

\hyxmp@big@prime Define a couple of large prime numbers that can still be stored in a TEX counter.
\hyxmp@big@prime@ii 1107 \def\hyxmp@big@prime{536870923}
1108 \def\hyxmp@big@prime@ii{536870027}

\hyxmp@seed@rng Seed hyperxmp's random-number generator from a given piece of text.
\hyxmp@one@token 1109 \def\hyxmp@seed@rng#1{%
1110 \@tempcnta=\hyxmp@big@prime
1111 \futurelet\hyxmp@one@token\hyxmp@seed@rng@i#1\@empty
1112 }

\hyxmp@seed@rng@i Do all of the work for \hyxmp@seed@rng. For each character code  $c$  of the input
\hyxmp@one@token text, assign  $\@tempcnta \leftarrow 3 \cdot \@tempcnta + c \pmod{\hyxmp@big@prime}$ .
\next 1113 \def\hyxmp@seed@rng@i{%
1114 \ifx\hyxmp@one@token\@empty
1115 \let\next=\relax
1116 \else
1117 \def\next##1{%
1118 \multiply\@tempcnta by 3
1119 \advance\@tempcnta by '##1
1120 \hyxmp@modulo@a{\hyxmp@big@prime}%
1121 \futurelet\hyxmp@one@token\hyxmp@seed@rng@i
1122 }%
1123 \fi
1124 \next
1125 }

\hyxmp@set@rand@num Advance \hyxmp@rand@num to the next pseudorandom number in the se-
\hyxmp@rand@num quence. Specifically, we assign  $\hyxmp@rand@num \leftarrow 3 \cdot \hyxmp@rand@num + \hyxmp@big@prime@ii \pmod{\hyxmp@big@prime}$ . Note that both  $\@tempcnta$ 
and  $\@tempcntb$  are overwritten in the process.
1126 \def\hyxmp@set@rand@num{%
1127 \@tempcnta=\hyxmp@rand@num
1128 \multiply\@tempcnta by 3
1129 \advance\@tempcnta by \hyxmp@big@prime@ii
1130 \hyxmp@modulo@a{\hyxmp@big@prime}%
1131 \xdef\hyxmp@rand@num{\the\@tempcnta}%
1132 }

\hyxmp@append@hex Append a randomly selected hexadecimal digit to macro #1. Note that both
\@tempcnta and \@tempcntb are overwritten in the process.
1133 \def\hyxmp@append@hex#1{%
1134 \hyxmp@set@rand@num
1135 \@tempcnta=\hyxmp@rand@num
1136 \hyxmp@modulo@a{16}%
1137 \ifnum\@tempcnta<10
1138 \xdef#1{#1\the\@tempcnta}%
1139 \else

```

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

```

1140     \advance\@tempcnta by -10
1141     \ifcase\@tempcnta
1142         \xdef#1{#1a}%
1143         \or\xdef#1{#1b}%
1144         \or\xdef#1{#1c}%
1145         \or\xdef#1{#1d}%
1146         \or\xdef#1{#1e}%
1147         \or\xdef#1{#1f}%
1148     \fi
1149 \fi
1150 }
```

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

```

1151 \def\hyxmp@append@hex@iii#1{%
1152     \hyxmp@append@hex#1%
1153     \hyxmp@append@hex#1%
1154     \hyxmp@append@hex#1%
1155 }
```

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

```

1156 \def\hyxmp@append@hex@iv#1{%
1157     \hyxmp@append@hex@iii#1%
1158     \hyxmp@append@hex#1%
1159 }
```

`\hyxmp@create@uuid` As per the definition of a version 4 UUID [12], define macro `#1` as a UUID of the form “`uuid:xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx`” in which each “*x*” is a lowercase hexadecimal digit and “*y*” is one of “8”, “9”, “a”, or “b”. We assume that the random-number generator is already seeded. Note that `\hyxmp@create@uuid` overwrites both `\@tempcnta` and `\@tempcntb`.

```

1160 \def\hyxmp@create@uuid#1{%
1161     \def#1{uuid:}%
1162     \hyxmp@append@hex@iv#1%
1163     \hyxmp@append@hex@iv#1%
1164     \g@addto@macro#1{-}%
1165     \hyxmp@append@hex@iv#1%
1166     \g@addto@macro#1{-4}%
1167     \hyxmp@append@hex@iii#1%
1168     \g@addto@macro#1{-}%

```

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

```

1169     \hyxmp@set@rand@num
1170     \@tempcnta=\hyxmp@rand@num
1171     \hyxmp@modulo@a{4}%
1172     \ifcase\@tempcnta
1173         \g@addto@macro#1{8}%
1174         \or\g@addto@macro#1{9}%
1175         \or\g@addto@macro#1{a}%

```

```

1176     \or\g@addto@macro#1{b}%
1177   \fi
1178   \hyxmp@append@hex@iii#1%
1179   \g@addto@macro#1{-}%
1180   \hyxmp@append@hex@iv#1%
1181   \hyxmp@append@hex@iv#1%
1182   \hyxmp@append@hex@iv#1%
1183 }

```

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

```

1184 \newcommand*{\hyxmp@def@DocumentID}{%
1185   \edef\hyxmp@seed@string{\hyxmp@jobname:\@pdftitle:\@pdfauthor}%
1186   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
1187   \edef\hyxmp@rand@num{\the\@tempcnta}%
1188   \hyxmp@create@uuid\hyxmp@DocumentID
1189 }

```

`\hyxmp@def@InstanceID` Seed the random-number generator with a function of the current filename, PDF document title, PDF author, and the current timestamp, then invoke `\hyxmp@create@uuid` to define `\hyxmp@InstanceID` as a random UUID. For the current timestamp, we use both the document-specified timestamp from `pdfdate` and the `TeX` time. The former can be more precise (to sub-seconds) but may be less random (as it depends on manual document modifications) while the latter is typically less precise (to minutes) but may be more random (as it is updated automatically).

```

1190 \newcommand*{\hyxmp@def@InstanceID}{%
1191   \hyxmp@today@xmp@define{\hyxmp@seed@string}%
1192   \edef\hyxmp@seed@string{%
1193     \hyxmp@jobname:\@pdftitle:\@pdfauthor:\hyxmp@today@xmp:\hyxmp@seed@string
1194   }%
1195   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
1196   \edef\hyxmp@rand@num{\the\@tempcnta}%
1197   \hyxmp@create@uuid\hyxmp@InstanceID
1198 }

```

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.

```

1199 \newcommand*{\hyxmp@add@to@xml}[1]{%
1200   \bgroup
1201     \@tempcnta=0
1202     \ifhyxmp@unicodetex
1203       \@tempcntb=65536%
1204     \else
1205       \@tempcntb=256%
1206     \fi
1207     \loop
1208       \lccode\@tempcnta=\@tempcnta
1209       \advance\@tempcnta by 1
1210       \ifnum\@tempcnta<\@tempcntb
1211         \repeat
1212         \lccode'\_='\ \relax
1213         \lccode'\^C='\,\relax
1214         \lccode'\^U='\_\relax
1215         \lowercase{\xdef\hyxmp@new@xml{#1}}%
1216         \xdef\hyxmp@xml{\hyxmp@xml\hyxmp@new@xml}%
1217     \egroup
1218 }
```

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

```

1219 \bgroup
1220 \catcode'\#=11
1221 \gdef\hyxmp@hash{#}
1222 \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@xml` `\hyxmp@padding` is defined to contain 32 lines of 63 spaces and a newline apiece for a total of 2048 characters of whitespace.

```

1223 \bgroup
1224 \xdef\hyxmp@xml{%
1225   \hyxmp@add@to@xml{%
1226     -----^^J%
1227   }
1228   \xdef\hyxmp@padding{\hyxmp@xml}%
1229 \egroup
1230 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
1231 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
```

```

1232 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
1233 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
1234 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}

```

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

```

1235 \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.

```

1236 \newcommand*{\hyxmp@define@pdfproducer}{%
1237   \gdef\@pdfproducer{TeX}
1238   \ifLuaTeX
1239     \expandafter\hyxmp@banner@to@producer\expandafter{\luatexbanner}%
1240   \else
1241     \ifPDFTeX
1242       \expandafter\hyxmp@banner@to@producer\expandafter{\pdftexbanner}%
1243     \else
1244       \ifXeTeX
1245         \edef\@pdfproducer{XeTeX version \the\XeTeXversion\XeTeXrevision}%
1246       \fi
1247     \fi
1248   \fi
1249 }

```

`\@pdfproducer` Define `\@pdfproducer` as the `TEX` engine's banner string (e.g., “This is
`\hyxmp@banner@to@producer` LuaHBTeX, Version 1.12.0 (TeX Live 2020)”), removing the initial “This is” if possible (specifically, when `ε-TEX`'s `\scantokens` primitive is available).

```

1250 \def\hyxmp@banner@to@producer#1{%
1251   \ifx\scantokens\relax
1252     \gdef\@pdfproducer{#1}%
1253   \else
1254     {\scantokens{\makeatletter\hyxmp@remove@this#1\relax}}%
1255   \fi
1256 }

```

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

```

1257 \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.

```

1258 \AtBeginDocument{%
1259   \ifx\pdfproducer\relax
1260     \hyxmp@define@pdfproducer
1261   \fi
1262 }
```

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

`\hyxmp@major@minor`

```

1263 \newcommand*{\hyxmp@assign@major@minor}{%
1264   \@ifundefined{pdfvariable}{%
1265     \@ifundefined{pdfminorversion}{%
```

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

```
1266   }{%
```

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

```

1267     \xdef\hyxmp@major@minor{\the\pdfminorversion}%
1268     \@ifundefined{pdfmajorversion}{%
```

Case 2(a): `\pdfmajorversion` is not defined (older versions of pdfL^AT_EX and LuaL^AT_EX).

```

1269       \xdef\hyxmp@major@minor{1.\hyxmp@major@minor}%
1270     }{%
```

Case 2(b): `\pdfmajorversion` is defined (pdfL^AT_EX 1.40.21+).

```

1271       \xdef\hyxmp@major@minor{\the\pdfmajorversion.\hyxmp@major@minor}%
1272     }%
1273   }%
1274 }
```

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

```

1275     \xdef\hyxmp@major@minor{\the\pdfvariable majorversion.\the\pdfvariable minorversion}%
1276   }%
1277 }
```

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

```
1278 \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.

```

1279 \hyxmp@add@simple@var{pdf:Producer}{@pdfproducer}%
1280 \hyxmp@add@simple@var{pdf:Keywords}{@pdfkeywords}%
1281 \hyxmp@add@simple{pdf:Trapped}{@pdftrapped}%
1282 \hyxmp@assign@major@minor
1283 \hyxmp@add@simple@var{pdf:PDFVersion}{hyxmp@major@minor}%
1284 }

```

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.

```

1285 \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.

```

1286 \newcommand*{\hyxmp@rdf@dc}[3][\iffalse]{%

```

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

```

1287 \@ifmtargexp{#3}{\@tempswafalse}{\@tempswatrue}%
1288 #1
1289 \@tempswatrue
1290 \fi

```

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

```

1291 \if@tempswa
1292 \hyxmp@xmlify{#3}%

```

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

```

1293 \let\hyxmp@value=\hyxmp@xmlified
1294 \hyxmp@add@to@xml{%
1295 -----<dc:#2>^^J%
1296 -----<rdf:Alt>^^J%
1297 }%

```

Record whether property #2 has definitions in multiple languages.

```

1298 \if@def@and@nonempty{hyxmp@alt@#2}{%
1299 \hyxmp@multi@langstrue
1300 }{%
1301 \hyxmp@multi@langsfalse
1302 }%

```

There are now four cases to consider: the cross product of $\{\text{pdfmetalang} = \text{"x-default"}, \text{pdfmetalang} \neq \text{"x-default"}\}$ and $\{\backslash\text{XMPLangAlt} \text{ was specified, } \backslash\text{XMPLangAlt} \text{ was not specified}\}$. We handle each of these in turn.

```
1303 \ifx\@pdfmetalang\hyxmp@x@default
1304 \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.

```
1305 \PackageError{hyperxmp}%
1306 {\string\XMPLangAlt\space was used without specifying
1307 pdfmetalang\MessageBreak
1308 or pdflang}%
1309 {Be sure to assign a language code to the pdfmetalang key and/or a
1310 document\MessageBreak
1311 language to the pdflang key (e.g., \string\hypersetup{pdfmetalang={en}}).}%
1312 \else
```

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

```
1313 \hyxmp@add@to@xml{%
1314 -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@value</rdf:li>^^J%
1315 }%
1316 \fi
1317 \else
1318 \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.

```
1319 \hyxmp@xmlify{\@pdfmetalang}%
1320 \hyxmp@add@to@xml{%
1321 -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@value</rdf:li>^^J%
1322 -----<rdf:li xml:lang="\hyxmp@xmlified">\hyxmp@value</rdf:li>^^J%
1323 }%
1324 \def\do##1##2{
1325 \hyxmp@xmlify{##2}%
1326 \hyxmp@add@to@xml{%
1327 -----<rdf:li xml:lang="##1">\hyxmp@xmlified</rdf:li>^^J%
1328 }%
1329 }%
1330 \csname hyxmp@alt@#2\endcsname
1331 \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.

```

1332      \hyxmp@xmlify{\pdfmetalang}%
1333      \hyxmp@add@to@xml{%
1334  -----<rdf:li xml:lang="\hyxmp@xmlified">\hyxmp@value</rdf:li>^^J%
1335      }%
1336      \fi
1337  \fi

```

Complete this XMP element.

```

1338      \hyxmp@add@to@xml{%
1339  -----</rdf:Alt>^^J%
1340  -----</dc:#2>^^J%
1341      }%
1342  \fi
1343 }%

```

`\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.

```

1344 \newcommand*{\hyxmp@list@to@xml}[4][\iffalse]{%

```

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

```

1345 \ifmtargexp{#4}{\@tempswafalse}{\@tempswatrue}%
1346 #1
1347 \if@tempswatrue
1348 \fi

```

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

```

1349 \if@tempswa
1350 \hyxmp@add@to@xml{%
1351 -----<dc:#2>^^J%
1352 -----<rdf:#3>^^J%
1353      }%
1354 \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`.

```

1355      \hyxmp@xmlify{#4}%
1356      \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
1357      \def\@elt##1{%
1358          \hyxmp@add@to@xml{%
1359  -----<rdf:li>##1</rdf:li>^^J%
1360          }%
1361      }%
1362      \hyxmp@list
1363      \egroup
1364      \hyxmp@add@to@xml{%
1365  -----</rdf:#3>^^J%

```

```

1366 -----</dc:#2>^^J%
1367     }%
1368   \fi
1369 }

```

`\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.

```

1370 \newcommand{\hyxmp@singleton@dc}[3][Bag]{%
1371   \@ifnotmtargexp{#3}{%
1372     \hyxmp@xmlify{#3}%
1373     \hyxmp@add@to@xml{%
1374       -----<dc:#2>^^J%
1375       -----<rdf:#1>^^J%
1376       -----<rdf:li>\hyxmp@xmlified</rdf:li>^^J%
1377       -----</rdf:#1>^^J%
1378       -----</dc:#2>^^J%
1379     }%
1380   }
1381 }

```

`\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).

```

1382 \newcommand*{\hyxmp@cond@dc@identifier}[2]{%
1383   \ifx\hyxmp@xmlified\@empty
1384     \@ifnotmtargexp{#2}{%
1385       \hyxmp@add@simple@pfx{dc:identifier}{#1}{#2}%
1386     }%
1387   \fi
1388 }

```

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

```

1389 \newcommand*{\hyxmp@dc@schema}{%
1390   \hyxmp@add@simple{dc:format}{application/pdf}%
1391   \hyxmp@rdf@dc[\ifHy@pdfa]{title}{\@pdftitle}%
1392   \hyxmp@rdf@dc[\ifHy@pdfa]{description}{\@pdfsubject}%
1393   \hyxmp@rdf@dc{rights}{\@pdfcopyright}%

```

```

1394 \hyxmp@singleton@dc{publisher}{\@pdfpublisher}%
1395 \@ifmtargexp{\@pdfdatetime}{%
1396   \hyxmp@singleton@dc[Seq]{date}{\hyxmp@today@xmp}%
1397 }{%
1398   \hyxmp@singleton@dc[Seq]{date}{\@pdfdatetime}%
1399 }%
1400 \hyxmp@singleton@dc{type}{\@pdftype}%
1401 \hyxmp@list@to@xml[\ifHy@pdfa]{creator}{Seq}{\hyxmp@pdfauthor}%
1402 \hyxmp@list@to@xml{subject}{Bag}{\hyxmp@pdfkeywords}%
1403 \ifx\@pdfsource\@empty
1404 \else
1405   \hyxmp@add@simple{dc:source}{\@pdfsource}%
1406 \fi
1407 \hyxmp@list@to@xml{language}{Bag}{\hyxmp@dc@lang}%
1408 % If |\@pdfidentifier| is empty, try setting it to each of |\@pdfdoi|,
1409 % |\@pdfeissn|, |\@pdfissn|, and |\@pdfisbn|, in turn, with proper
1410 % syntactic adjustments.
1411 %   \begin{macrocode}
1412 \@ifmtargexp{\@pdfidentifier}{%
1413   \let\hyxmp@xmlified=\@empty
1414   \hyxmp@cond@dc@identifier{info:doi/}{\@pdfdoi}%
1415   \hyxmp@cond@dc@identifier{urn:ISSN:}{\@pdfeissn}%
1416   \hyxmp@cond@dc@identifier{urn:ISSN:}{\@pdfissn}%
1417   \hyxmp@cond@dc@identifier{urn:ISBN:}{\@pdfisbn}%
1418 }{%
1419   \hyxmp@add@simple{dc:identifier}{\@pdfidentifier}%
1420 }%
1421 }

```

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.

```

1422 \newcommand*{\hyxmp@xmpRights@schema}{%

```

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

```

1423 \let\hyxmp@rights=\@empty
1424 \ifx\@pdflicenseurl\@empty
1425 \else
1426   \def\hyxmp@rights{YES}%
1427 \fi
1428 \ifx\@pdfcopyright\@empty
1429 \else
1430   \def\hyxmp@rights{YES}%
1431 \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.

```

1432 \ifx\hyxmp@rights\@empty
1433 \else
1434 \ifx\@pdfcopyright\@empty
1435 \else
1436 \hyxmp@add@simple{xmpRights:Marked}{True}%
1437 \fi
1438 \hyxmp@add@simple{xmpRights:WebStatement}{\@pdflicenseurl}%
1439 \fi
1440 }

```

3.6.5 The XMP Media Management schema

`\hyxmp@mm@schema` Add properties defined by the XMP Media Management schema to the `\hyxmp@xml` macro. According to the XMP specification, the `xmpMM:DocumentID` property is supposed to uniquely identify a document, and the `xmpMM:InstanceID` property is supposed to change with each save operation [5]. As seen in Section 3.5, we do what we can to honor this intention from within a TeX-based workflow. We additionally support the `xmpMM:VersionID` property, whose value is supplied by the author using `pdfversionid`.

```

1441 \gdef\hyxmp@mm@schema{%
1442 \ifmtargexp{\hyxmp@DocumentID}{\hyxmp@def@DocumentID}{}%
1443 \ifmtargexp{\hyxmp@InstanceID}{\hyxmp@def@InstanceID}{}%
1444 \hyxmp@add@simple{xmpMM:DocumentID}{\hyxmp@DocumentID}%
1445 \hyxmp@add@simple{xmpMM:InstanceID}{\hyxmp@InstanceID}%
1446 \hyxmp@add@simple{xmpMM:VersionID}{\@pdfversionid}%
1447 \hyxmp@add@simple{xmpMM:RenditionClass}{\@pdfrendition}%
1448 }

```

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`.

```

1449 \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.
1450 \ifmtargexp{\@pdfcreationdate}{%
1451 \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@today@xmp}%
1452 }{%
1453 \hyxmp@add@simple{xmp:CreateDate}{%
1454 \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfcreationdate}}%
1455 }%

```

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

```
1456 \ifmtargexp{\@pdfmoddate}{%
1457   \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@today@xmp}%
1458 }{%
1459   \hyxmp@add@simple{xmp:ModifyDate}{%
1460     \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfmoddate}}%
1461 }%
```

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

```
1462 \ifmtargexp{\@pdfmetadatettime}{%
1463   \hyxmp@add@simple{xmp:MetadataDate}{\hyxmp@today@xmp}%
1464 }{%
1465   \hyxmp@add@simple{xmp:MetadataDate}{\@pdfmetadatettime}%
1466 }%
```

Define the creation tool and the base URL.

```
1467 \hyxmp@add@simple{xmp:CreatorTool}{\@pdfcreator}%
1468 \hyxmp@add@simple{xmp:BaseURL}{\@baseurl}%
1469 }
```

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.

```
1470 \gdef\hyxmp@photoshop@schema{%
1471   \edef\hyxmp@photoshop@data{\@pdfauthortitle\@pdfcaptionwriter}%
1472   \hyxmp@add@simple{photoshop:AuthorsPosition}{\@pdfauthortitle}%
1473   \hyxmp@add@simple{photoshop:CaptionWriter}{\@pdfcaptionwriter}%
1474 }
```

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`.

```
1475 \newcommand*{\hyxmp@pdfa@id@schema}{%
1476   \ifHy@pdfa
1477     \hyxmp@add@simple{pdfaid:part}{\@pdfapart}%
1478     \hyxmp@add@simple{pdfaid:conformance}{\@pdfaconformance}%
1479   \fi
1480 }
```

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

```
1481 \newcommand*{\hyxmp@pdfua@id@schema}{%
1482   \hyxmp@add@simple{pdfua:part}{\@pdfuapart}%
1483 }
```

`\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.

```
1484 \newcommand*{\hyxmp@pdfx@id@schema}{%
1485   \@tempcnta=0\hyxmp@pdfx@major\relax
1486   \ifnum\@tempcnta=0
1487   \else
1488     \ifnum\@tempcnta=1
1489       \hyxmp@add@simple{pdfx:GTS_PDFXVersion}{PDF/X-1:2001}%
1490       \hyxmp@add@simple{pdfx:GTS_PDFXConformance}{\@pdfxstandard}%
1491     \else
1492       \ifnum\@tempcnta<4
1493         \hyxmp@add@simple{pdfx:GTS_PDFXVersion}{\@pdfxstandard}%
1494       \else
1495         \hyxmp@add@simple{pdfx:GTS_PDFXVersion}{\@pdfxstandard}%
1496       \fi
1497     \fi
1498   \fi
1499 }
```

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.

```
1500 \begingroup
1501   \catcode'\&=12
1502   \catcode'\#=12
1503   \gdef\xmplinesep{&#xA;}
1504 \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.

```
1505 \newcommand*{\hyxmp@list@to@lines}[2]{%
1506   \@ifnotmtargexp{#2}{%
1507     \bgroup
1508     \hyxmp@add@to+xml{%
1509       \hyxmp@extra@indent_____<#1>%
1510     }%
1511   }
```

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


```

1511     \def\@elt@first##1{%
1512         \hyxmp@add@to@xml{##1}%
1513         \let\@elt=\@elt@rest
1514     }%

```

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

```

1515     \def\@elt@rest##1{%
1516         \hyxmp@add@to@xml{\xmplinesep##1}%
1517     }%

```

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

```

1518     \let\@elt=\@elt@first
1519     \hyxmp@xmlify{#2}%
1520     \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlied}%
1521     \hyxmp@list
1522     \hyxmp@add@to@xml{</#1>^^J}%
1523     \egroup
1524 }%
1525 }

```

`\hyxmp@iptc@schema` Add properties defined by the IPTC Photo Metadata schema [10] to the `\hyxmp@xml` macro. We currently support only the `Iptc4xmpCore:CreatorContactInfo` property, although this is a structure containing multiple fields.

```

1526 \gdef\hyxmp@iptc@schema{%

```

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

```

1527     \ifx\hyxmp@iptc@data\@empty
1528     \else
1529         \hyxmp@add@to@xml{%
1530         _____<Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">^^J%
1531         }%

```

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

```

1532     \bgroup
1533     \edef\hyxmp@extra@indent{\hyxmp@extra@indent\space\space}%
1534     \hyxmp@list@to@lines{Iptc4xmpCore:CIAdrExtadr}{\@pdfcontactaddress}%
1535     \hyxmp@add@simple{Iptc4xmpCore:CIAdrCity}{\@pdfcontactcity}%
1536     \hyxmp@add@simple{Iptc4xmpCore:CIAdrRegion}{\@pdfcontactregion}%
1537     \hyxmp@add@simple{Iptc4xmpCore:CIAdrPcode}{\@pdfcontactpostcode}%
1538     \hyxmp@add@simple{Iptc4xmpCore:CIAdrCtry}{\@pdfcontactcountry}%

```

`\xmplinesep` The IPTC standard states that sets of telephone numbers, email addresses, and URLs for the contact person or institution, “[m]ay have to be separated by a comma in the user interface” [10]. This is rather ambiguous: Does the comma

appear *only* in the user interface or also in the generated XML? Here we assume the latter interpretation and temporarily redefine `\xmplinesep` as a comma and use `\hyxmp@list@to@lines` to insert the data. Unlike `\hyxmp@add@simple`, this approach trims all spaces surrounding commas.

```

1539     \def\xmplinesep{,}%
1540     \hyxmp@list@to@lines{Iptc4xmpCore:CTelWork}{\@pdfcontactphone}%
1541     \hyxmp@list@to@lines{Iptc4xmpCore:CtEmailWork}{\@pdfcontactemail}%
1542     \hyxmp@list@to@lines{Iptc4xmpCore:CtUrlWork}{\@pdfcontacturl}%
1543     \egroup
1544     \hyxmp@add@to@xml{%
1545 -----</Iptc4xmpCore:CreatorContactInfo>^^J%
1546     }%
1547     \fi
1548 }

```

3.6.10 The PRISM Basic Metadata schema

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

```

1549 \newcommand*{\hyxmp@prism@schema}{%
1550   \ifx\hyxmp@prism@data\@empty
1551     \else
1552       \hyxmp@add@simple{prism:complianceProfile}{three}%
1553     \fi
1554     \hyxmp@add@simple@lang{prism:subtitle}{\@pdfsubtitle}%
1555     \hyxmp@add@simple@lang{prism:publicationName}{\@pdfpublication}%
1556     \hyxmp@add@simple{prism:aggregationType}{\@pdfpubtype}%
1557     \hyxmp@add@simple@lang{prism:bookEdition}{\@pdfbookedition}%
1558     \hyxmp@add@simple{prism:volume}{\@pdfvolumenum}%
1559     \hyxmp@add@simple{prism:number}{\@pdfissuenum}%
1560     \hyxmp@add@simple{prism:pageRange}{\@pdfpagerange}%
1561     \hyxmp@add@simple{prism:isbn}{\@pdfisbn}%
1562     \hyxmp@add@simple{prism:issn}{\@pdfissn}%
1563     \hyxmp@add@simple{prism:eIssn}{\@pdfeissn}%
1564     \hyxmp@add@simple{prism:doi}{\@pdfdoi}%
1565     \hyxmp@add@simple{prism:url}{\@pdfurl}%
1566     \hyxmp@add@simple{prism:byteCount}{\@pdfbytes}%
1567     \hyxmp@add@simple{prism:pageCount}{\@pdfnumpages}%
1568 }

```

3.6.11 The Journal Article Versions (JAV) schema

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

```

1569 \newcommand*{\hyxmp@jav@schema}{%
1570   \hyxmp@add@simple{jav:journal_article_version}{\@pdfpubstatus}%
1571 }

```

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.

```
1572 \newcommand*{\hyxmp@xmptpg@schema}{%
1573   \ifLuaTeX
1574     \luadirect{write_xmp_font_list(\the\hyxmp@cct)}%
1575   \fi
1576   \hyxmp@add@simple{xmpTPg:NPages}{\@pdfnumpages}%
1577 }
```

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

```
1578 \ifLuaTeX
1579   \newcatcodetable\hyxmp@cct
1580   \savecatcodetable\hyxmp@cct
1581 \fi
```

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

```
1582 \bgroup
1583   \catcode'\_ =11
1584   \gdef\hyxmp@prot@us{_%}
1585 \egroup
```

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

```
1586 \ifLuaTeX
1587   \begin{luacode*}
1588   function write_xmp_font_list (cct)
1589     local function show_field(name, ...)
1590       for i = 1, select("#", ...) do
1591         local val = select(i, ...)
1592         if val then
1593           local xml = string.gsub(val, "&", "&amp;")
1594           xml = string.gsub(xml, "<", "&lt;")
1595           xml = string.gsub(xml, ">", "&gt;")
1596           xml = string.gsub(xml, "_", "\\hyxmp@prot@us ")
1597           tex.print(cct, "_____<stFnt:" .. name .. ">" ..
1598             xml .. "</stFnt:" .. name .. ">^^J%")
1599         end
1600       end
1601     end
1602   end
1603   tex.print(cct, "\\hyxmp@add@to+xml{%")
1604   tex.print(cct, "_____<xmpTPg:Fonts>^^J%")
1605 }
```

```

1605 tex.print(cct, "_____<rdf:Bag>^^J%")
1606 for i, f in font.each() do
1607   tex.print(cct, "_____<rdf:li rdf:parseType=\"Resource\">^^J%")
1608   if f.filename then
1609     local fname = string.gsub(f.filename, "^harfloaded:(.*)", "%1")
1610     local info = fontloader.info(fname)
1611     if info then
1612       show_field("fontFace", info.fullname)
1613       show_field("fontFamily", info.familyname)
1614       show_field("fontName", info.fontname)
1615       show_field("versionString", info.version)
1616     end
1617     local baseName = string.gsub(f.filename, ".*[/\\](.*)", "%1")
1618     show_field("fontFileName", baseName)
1619   else
1620     show_field("fontName", f.psname, f.fullname, f.name)
1621   end
1622   if f.format and f.format ~= "unknown" then
1623     show_field("fontType", f.format)
1624   end
1625   tex.print(cct, "_____</rdf:li>^^J%")
1626 end
1627 tex.print(cct, "_____</rdf:Bag>^^J%")
1628 tex.print(cct, "_____</xmpTPg:Fonts>^^J%")
1629 tex.print(cct, "}")
1630 end
1631 \end{luacode*}
1632 \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.
1633 \newcommand*{\hyxmp@check@iptc@data}{%
1634 \edef\hyxmp@iptc@data{%
1635   \@pdfcontactaddress
1636   \@pdfcontactcity
1637   \@pdfcontactregion
1638   \@pdfcontactpostcode
1639   \@pdfcontactcountry
1640   \@pdfcontactphone
1641   \@pdfcontactemail
1642   \@pdfcontacturl

```

```

1643 }%
1644 }%

\hyxmp@check@prism@data Define \hyxmp@prism@data as the concatenation of all PRISM metadata supplied
\hyxmp@prism@data by the document.
1645 \newcommand*{\hyxmp@check@prism@data}{%
1646 \edef\hyxmp@prism@data{%
1647 \pdfbookedition
1648 \pdfbytes
1649 \pdfdoi
1650 \pdfeissn
1651 \pdfisbn
1652 \pdfissn
1653 \pdfissuenum
1654 \pdfnumpages
1655 \pdfpagerange
1656 \pdfpublication
1657 \pdfpubtype
1658 \pdfsubtitle
1659 \pdfurl
1660 \pdfvolumenum
1661 }%
1662 }%

\hyxmp@check@jav@data Define \hyxmp@jav@data as the concatenation of all JAV metadata supplied by the
\hyxmp@jav@data document.
1663 \newcommand*{\hyxmp@check@jav@data}{%
1664 \edef\hyxmp@jav@data{%
1665 \pdfpubstatus
1666 }%
1667 }

\hyxmp@begin@extension@decls Begin a block of XML tags that indicates we're declaring one or more extension
schemas.
1668 \newcommand*{\hyxmp@begin@extension@decls}{%
1669 \hyxmp@add@to@xml{%
1670 -----<pdfaExtension:schemas>^^J%
1671 -----<rdf:Bag>^^J%
1672 }%
1673 }

\hyxmp@end@extension@decls End the block of XML tags begun by \hyxmp@begin@extension@decls.
1674 \newcommand*{\hyxmp@end@extension@decls}{%
1675 \hyxmp@add@to@xml{%
1676 -----</rdf:Bag>^^J%
1677 -----</pdfaExtension:schemas>^^J%
1678 }%
1679 }

```

`\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.

```

1680 \newcommand*{\hyxmp@begin@ext@decl}[3]{%
1681   \hyxmp@add@to@xml{%
1682     <rdf:li rdf:parseType="Resource">^^J%
1683     <pdfaSchema:schema>#1</pdfaSchema:schema>^^J%
1684     <pdfaSchema:prefix>#2</pdfaSchema:prefix>^^J%
1685     <pdfaSchema:namespaceURI>#3</pdfaSchema:namespaceURI>^^J%
1686     <pdfaSchema:property>^^J%
1687     <rdf:Seq>^^J%
1688   }%
1689 }%
```

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

```

1690 \newcommand*{\hyxmp@end@ext@decl}{%
1691   \hyxmp@add@to@xml{%
1692     </rdf:Seq>^^J%
1693     </pdfaSchema:property>^^J%
1694     </rdf:li>^^J%
1695   }%
1696 }%
```

`\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.

```

1697 \newcommand{\hyxmp@declare@property}[4][Text]{%
1698   \hyxmp@add@to@xml{%
1699     <rdf:li rdf:parseType="Resource">^^J%
1700     <pdfaProperty:name>}%
1701   \xdef\hyxmp@xml{\hyxmp@xml#2}%
1702   \hyxmp@add@to@xml{</pdfaProperty:name>^^J%
1703     <pdfaProperty:valueType>#1</pdfaProperty:valueType>^^J%
1704     <pdfaProperty:category>#3</pdfaProperty:category>^^J%
1705     <pdfaProperty:description>#4</pdfaProperty:description>^^J%
1706     </rdf:li>^^J%
1707   }%
1708 }%
```

`\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.

```

1709 \newcommand{\hyxmp@declare@field}[3][Text]{%
1710   \hyxmp@add@to@xml{%
1711     <rdf:li rdf:parseType="Resource">^^J%
1712     <pdfaField:name>#2</pdfaField:name>^^J%
1713     <pdfaField:valueType>#1</pdfaField:valueType>^^J%
1714     <pdfaField:description>#3</pdfaField:description>^^J%
1715     </rdf:li>^^J%
1716   }%
```

1717 }

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

```
1718 \newcommand*{\hyxmp@pdf@extensions}{%
1719   \hyxmp@begin@ext@decl
1720     {Adobe PDF Schema}%
1721     {pdf}%
1722     {http://ns.adobe.com/pdf/1.3/}%
1723   \hyxmp@declare@property
1724     {Trapped}%
1725     {internal}%
1726     {Indication if the document has been modified to include trapping information}%
1727   \hyxmp@end@ext@decl
1728 }%
```

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

```
1729 \newcommand*{\hyxmp@mm@extensions}{%
1730   \hyxmp@begin@ext@decl
1731     {XMP Media Management Schema}%
1732     {xmpMM}%
1733     {http://ns.adobe.com/xap/1.0/mm/}%
1734   \hyxmp@declare@property
1735     [URI]
1736     {DocumentID}%
1737     {internal}%
1738     {UUID based identifier for all versions and renditions of a document}%
1739   \hyxmp@declare@property
1740     [URI]
1741     {InstanceID}%
1742     {internal}%
1743     {UUID based identifier for specific incarnation of a document}%
1744   \hyxmp@declare@property
1745     {VersionID}%
1746     {internal}%
1747     {Document version identifier}%
1748   \hyxmp@declare@property
1749     [RenditionClass]%
1750     {RenditionClass}%
1751     {internal}%
1752     {The manner in which a document is rendered}%
1753   \hyxmp@end@ext@decl
1754 }%
```

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

```
1755 \newcommand*{\hyxmp@pdfa@id@extensions}{%
1756   \hyxmp@begin@ext@decl
1757     {PDF/A Identification Schema}%
1758     {pdfaid}%
1759     {http://www.aiim.org/pdfa/ns/id/}%

```

```

1760 \hyxmp@declare@property
1761     [Integer]%
1762     {part}%
1763     {internal}%
1764     {Part of PDF/A standard}%
1765 \hyxmp@declare@property
1766     {conformance}%
1767     {internal}%
1768     {Conformance level of PDF/A standard}%
1769 \hyxmp@end@ext@decl
1770 }%

```

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

```

1771 \newcommand*{\hyxmp@pdfua@id@extensions}{%
1772     \hyxmp@begin@ext@decl
1773     {PDF/UA Universal Accessibility Schema}%
1774     {pdfuaid}%
1775     {http://www.aiim.org/pdfua/ns/id/}%
1776 \hyxmp@declare@property
1777     [Integer]%
1778     {part}%
1779     {internal}%
1780     {Part of ISO 14289 standard}%
1781 \hyxmp@end@ext@decl
1782 }%

```

\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.

```

1783 \newcommand*{\hyxmp@pdfx@id@extensions}{%
1784     \ifx\hyxmp@pdfx@major\empty
1785     \else
1786     \hyxmp@begin@ext@decl
1787         {Adobe Document Info PDF/X eXtension Schema}%
1788         {pdfx}%
1789         {http://ns.adobe.com/pdfx/1.3/}%
1790     \hyxmp@declare@property
1791         {GTS_PDFXVersion}%
1792         {internal}%
1793         {ID of PDF/X standard}%
1794     \hyxmp@declare@property
1795         {GTS_PDFXConformance}%
1796         {internal}%
1797         {Conformance level of PDF/X standard}%
1798     \hyxmp@end@ext@decl
1799 \fi

```

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

```

1800 \@tempcnta=0\hyxmp@pdfx@major\relax
1801 \ifnum\@tempcnta>3
1802     \hyxmp@begin@ext@decl

```



```

1803      {PDF/X ID Schema}%
1804      {pdfxid}%
1805      {http://www.npes.org/pdfx/ns/id/}%
1806      \hyxmp@declare@property
1807      {GTS_PDFXVersion}%
1808      {internal}%
1809      {ID of PDF/X standard}%
1810      \hyxmp@end@ext@decl
1811      \fi
1812 }%

```

`\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.

```

1813 \newcommand*{\hyxmp@iptc@extensions}{%
1814   \hyxmp@begin@ext@decl
1815     {IPTC Core Schema}%
1816     {Iptc4xmpCore}%
1817     {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}%
1818   \hyxmp@declare@property
1819     [ContactInfo]
1820     {CreatorContactInfo}
1821     {external}
1822     {Document creator's contact information}

```

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

```

1823   \hyxmp@add@to+xml{%
1824     _____</rdf:Seq>^^J%
1825     _____</pdfaSchema:property>^^J%
1826     _____<pdfaSchema:valueType>^^J%
1827     _____<rdf:Seq>^^J%
1828     _____<rdf:li rdf:parseType="Resource">^^J%
1829     _____<pdfaType:type>ContactInfo</pdfaType:type>^^J%
1830     _____<pdfaType:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaType:
1831     _____<pdfaType:prefix>Iptc4xmpCore</pdfaType:prefix>^^J%
1832     _____<pdfaType:description>%
1833         Basic set of information to get in contact with a person%
1834     _____</pdfaType:description>^^J%
1835     _____<pdfaType:field>^^J%
1836     _____<rdf:Seq>^^J%
1837   }%
1838   \hyxmp@declare@field
1839     {CiAdrCity}%
1840     {Contact information city}%
1841   \hyxmp@declare@field
1842     {CiAdrCtry}%
1843     {Contact information country}%
1844   \hyxmp@declare@field

```

```

1845         {CiAdrExtadr}%
1846         {Contact information address}%
1847 \hyxmp@declare@field
1848         {CiAdrPcode}%
1849         {Contact information local postal code}%
1850 \hyxmp@declare@field
1851         {CiAdrRegion}%
1852         {Contact information regional information such as state or province}%
1853 \hyxmp@declare@field
1854         {CiEmailWork}%
1855         {Contact information email address(es)}%
1856 \hyxmp@declare@field
1857         {CiTelWork}%
1858         {Contact information telephone number(s)}%
1859 \hyxmp@declare@field
1860         {CiUrlWork}%
1861         {Contact information Web URL(s)}%
1862 \hyxmp@add@to+xml{%
1863 -----</rdf:Seq>^^J%
1864 -----</pdfaType:field>^^J%
1865 -----</rdf:li>^^J%
1866 -----</rdf:Seq>^^J%
1867 -----</pdfaSchema:valueType>^^J%
1868 -----</rdf:li>^^J%
1869 }%
1870 }

```

\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.

```

1871 \newcommand*{\hyxmp@prism@extensions}{%
1872 \hyxmp@begin@ext@decl
1873     {PRISM Basic Metadata}%
1874     {prism}%
1875     {http://prismstandard.org/namespaces/basic/3.0/}%
1876 \hyxmp@declare@property
1877     {complianceProfile}%
1878     {internal}%
1879     {PRISM specification compliance profile to which this document adheres}%
1880 \hyxmp@declare@property
1881     {publicationName}%
1882     {external}%
1883     {Publication name}%
1884 \hyxmp@declare@property
1885     {aggregationType}%
1886     {external}%
1887     {Publication type}%
1888 \hyxmp@declare@property
1889     {bookEdition}%

```

```

1890         {external}%
1891         {Edition of the book in which the document was published}%
1892 \hyxmp@declare@property
1893         {volume}%
1894         {external}%
1895         {Publication volume number}%
1896 \hyxmp@declare@property
1897         {number}%
1898         {external}%
1899         {Publication issue number within a volume}%
1900 \hyxmp@declare@property
1901         {pageRange}%
1902         {external}%
1903         {Page range for the document within the print version of its publication}%
1904 \hyxmp@declare@property
1905         {issn}%
1906         {external}%
1907         {ISSN for the printed publication in which the document was published}%
1908 \hyxmp@declare@property
1909         {eIssn}%
1910         {external}%
1911         {ISSN for the electronic publication in which the document was published}%
1912 \hyxmp@declare@property
1913         {isbn}%
1914         {external}%
1915         {ISBN for the publication in which the document was published}%
1916 \hyxmp@declare@property
1917         {doi}%
1918         {external}%
1919         {Digital Object Identifier for the document}%
1920 \hyxmp@declare@property
1921         [URL]
1922         {url}%
1923         {external}%
1924         {URL at which the document can be found}%
1925 \hyxmp@declare@property
1926         [Integer]
1927         {byteCount}%
1928         {internal}%
1929         {Approximate file size in octets}%
1930 \hyxmp@declare@property
1931         [Integer]
1932         {pageCount}%
1933         {internal}%
1934         {Number of pages in the print version of the document}%
1935 \hyxmp@declare@property
1936         {subtitle}%
1937         {external}%
1938         {Document's subtitle}%
1939 \hyxmp@end@ext@decl

```

```

1940 }%

\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.
1941 \newcommand*{\hyxmp@jav@extensions}{%
1942   \hyxmp@begin@ext@decl
1943     {NISO/ALPSP Journal Article Versions}%
1944     {jav}%
1945     {http://www.niso.org/schemas/jav/1.0/}%
1946   \hyxmp@declare@property
1947     [Closed Choice of Text]%
1948     {journal_article_version}%
1949     {external}%
1950     {Article status, one of "A0", "SMUR", "AM", "P", "VoR", "CVoR", or "EVoR"}%
1951   \hyxmp@end@ext@decl
1952 }%

\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.
1953 \newcommand*{\hyxmp@declare@extensions}{%
1954   \hyxmp@begin@extension@decls
1955     Declare the Adobe PDF schema (always present).
1956   \hyxmp@pdf@extensions
1957     Declare the XMP Media Management extensions (always present).
1958   \hyxmp@mm@extensions
1959     Declare the PDF/A Identification extensions, but only when generating a PDF/A
1960     document.
1961   \ifHy@pdfa
1962     \hyxmp@pdfa@id@extensions
1963   \fi
1964     Conditionally declare the PDF/UA Universal Accessibility extensions.
1965   \ifx\@pdfuapart\@empty
1966   \else
1967     \hyxmp@pdfua@id@extensions
1968   \fi

\next  Conditionally declare the PDF/X extensions.
1969   \ifx\@pdfxversion\@empty
1970   \else
1971     \hyxmp@pdfx@id@extensions
1972   \fi

1973   Conditionally declare IPTC photo metadata extensions.
1974   \ifx\hyxmp@iptc@data\@empty

```

```

1969 \else
1970 \hyxmp@iptc@extensions
1971 \fi

    Conditionally declare PRISM basic metadata extensions.
1972 \ifx\hyxmp@prism@data\@empty
1973 \else
1974 \hyxmp@prism@extensions
1975 \fi

    Conditionally declare JAV metadata.
1976 \ifx\hyxmp@jav@data\@empty
1977 \else
1978 \hyxmp@jav@extensions
1979 \fi
1980 \hyxmp@end@extension@decls
1981 }

```

3.6.14 Combining schemata into an XMP packet

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

```

1982 \begingroup
1983 \ifhyxmp@unicodetex
1984 \lccode'\!="FEFF %
1985 \lowercase{%
1986 \gdef\hyxmp@bom{!}
1987 }%
1988 \else
1989 \catcode'\^^ef=12
1990 \catcode'\^^bb=12
1991 \catcode'\^^bf=12
1992 \gdef\hyxmp@bom{^^ef^^bb^^bf}%
1993 \fi
1994 \endgroup

```

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

```

1995 \def\hyxmp@construct@packet{%
1996 \gdef\hyxmp+xml{}%
1997 \hyxmp@add@to+xml{<?xpacket begin="\hyxmp@bom" %
1998 id="W5M0MpCehiHzreSzNTczkc9d"?>^^J%
1999 <x:xmpmeta xmlns:x="adobe:ns:meta/">^^J%
2000 __<rdf:RDF %
2001 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\hyxmp@hash">^^J%
2002 ____<rdf:Description rdf:about="^^J%

```

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

```

2003 _____xmlns:pdf="http://ns.adobe.com/pdf/1.3/"^^J%
2004 _____xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/"^^J%

```

```

2005 -----xmlns:dc="http://purl.org/dc/elements/1.1/"^^J%
2006 -----xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/"^^J%
2007 -----xmlns:xmp="http://ns.adobe.com/xap/1.0/"^^J%
2008 -----xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"^^J%
2009 -----xmlns:stEvt="http://ns.adobe.com/xap/1.0/sType/ResourceEvent\hyxmp@hash"^^J%
2010 -----xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/"^^J%
2011 -----xmlns:pdfuaid="http://www.aiim.org/pdfua/ns/id/"^^J%
2012 -----xmlns:pdfx="http://ns.adobe.com/pdfx/1.3/"^^J%
2013 -----xmlns:pdfxid="http://www.npes.org/pdfx/ns/id/"^^J%
2014 -----xmlns:prism="http://prismstandard.org/namespaces/basic/3.0/"^^J%
2015 -----xmlns:jav="http://www.niso.org/schemas/jav/1.0/"^^J%
2016 -----xmlns:xmpTPg="http://ns.adobe.com/xap/1.0/t/pg/"^^J%
2017 -----xmlns:stFnt="http://ns.adobe.com/xap/1.0/sType/Font\hyxmp@hash"^^J%
2018 -----xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"^^J%
2019 -----xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"^^J%
2020 -----xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema\hyxmp@hash"^^J%
2021 -----xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property\hyxmp@hash"^^J%
2022 -----xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type\hyxmp@hash"^^J%
2023 -----xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field\hyxmp@hash">^^J%
2024 }%

```

Declare non-standard schemata.

```

2025 \hyxmp@check@iptc@data
2026 \hyxmp@check@prism@data
2027 \hyxmp@check@jav@data
2028 \hyxmp@declare@extensions

```

Insert all the metadata we know how to insert.

```

2029 \hyxmp@pdf@schema
2030 \hyxmp@xmpRights@schema
2031 \hyxmp@dc@schema
2032 \hyxmp@photoshop@schema
2033 \hyxmp@xmp@basic@schema
2034 \hyxmp@pdfa@id@schema
2035 \hyxmp@pdfua@id@schema
2036 \hyxmp@pdfx@id@schema
2037 \hyxmp@mm@schema
2038 \hyxmp@iptc@schema
2039 \hyxmp@prism@schema
2040 \hyxmp@jav@schema
2041 \hyxmp@xmptpg@schema
2042 \hyxmp@add@to+xml{%
2043 ____</rdf:Description>^^J%
2044 __</rdf:RDF>^^J%
2045 </x:xmpmeta>^^J%
2046 \hyxmp@padding
2047 <?xpacket end="w"?>^^J%
2048 }%
2049 }

```

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.
2050 \newcommand*{\hyxmp@embed@packet}{%
2051   \hyxmp@construct@packet
2052   \def\hyxmp@driver{hpdfTeX}%
2053   \ifx\hyxmp@driver\Hy@driver
2054     \hyxmp@embed@packet@pdfTeX
2055   \else
2056     \def\hyxmp@driver{hLuaTeX}%
2057     \ifx\hyxmp@driver\Hy@driver
2058       \hyxmp@embed@packet@LuaTeX
2059     \else
2060       \def\hyxmp@driver{hdvipdfm}%
2061       \ifx\hyxmp@driver\Hy@driver
2062         \hyxmp@embed@packet@dvipdfm
2063       \else
2064         \def\hyxmp@driver{hXeTeX}%
2065         \ifx\hyxmp@driver\Hy@driver
2066           \hyxmp@embed@packet@XeTeX
2067         \else
2068           \@ifundefined{pdfmark}{%
2069             \PackageWarningNoLine{hyperxmp}{%
2070               Unrecognized hyperref driver ‘\Hy@driver’.\MessageBreak
2071               \hyxmp@jobname.tex’s XMP metadata will *not* be\MessageBreak
2072               embedded in the resulting file}%
2073             }{%
2074               \hyxmp@embed@packet@pdfmark
2075             }%
2076           \fi
2077         \fi
2078       \fi
2079     \fi
2080 }

```

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.

```
2081 \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.

```
2082 \newcommand*{\hyxmp@embed@packet@pdftex}{%
2083   \bgroup
2084   \ifluatex
2085   \else
2086     \pdfcompresslevel=0
2087   \fi
2088   \immediate\pdfobj \ifluatex uncompressed\fi stream attr {%
2089     /Type /Metadata
2090     /Subtype /XML
2091   }{\hyxmp@xml}%
2092   \pdfcatalog {/Metadata \the\pdflastobj\space 0 R}%
2093   \egroup
2094 }
```

3.7.2 Embedding using LuaTeX 0.85+

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

```
2095 \newcommand*{\hyxmp@embed@packet@luatex}{%
2096   \immediate\pdfextension obj uncompressed stream attr {%
2097     /Type /Metadata
2098     /Subtype /XML
2099   }{\hyxmp@xml}%
2100   \pdfextension catalog {/Metadata \the\numexpr\pdffeedback lastobj\relax\space 0 R}%
2101 }
```

3.7.3 Embedding using any pdfmark-based backend

`\hyxmp@embed@packet@pdfmark` Embed the XMP packet using `hyperref`’s `\pdfmark` command. I believe `\pdfmark` is used by the `dvipdf`, `dvipsone`, `dvips`, `dviwindo`, `nativepdf`, `pdfmark`, `ps2pdf`, `textures`, and `vtexpdfmark` options to `hyperref`, but I’ve tested only a few of those.

```
2102 \newcommand*{\hyxmp@embed@packet@pdfmark}{%
2103   \pdfmark{%
2104     pdfmark=/NamespacePush
2105   }%
2106   \pdfmark{%
2107     pdfmark=/OBJ,
2108     Raw={/_objdef \string\hyxmp@Metadata\string} /type /stream}%

```



```

2109 }%
2110 \pdfmark{%
2111   pdfmark=/PUT,
2112   Raw={\string{hyxmp@Metadata\string}
2113     2 dict begin
2114       /Type /Metadata def
2115       /Subtype /XML def
2116       currentdict
2117     end
2118   }%
2119 }%
2120 \pdfmark{%
2121   pdfmark=/PUT,
2122   Raw={\string{hyxmp@Metadata\string} (\hyxmp@xml)}%
2123 }%
2124 \pdfmark{%
2125   pdfmark=/Metadata,
2126   Raw={\string{Catalog\string} \string{hyxmp@Metadata\string}}%
2127 }%
2128 \pdfmark{%
2129   pdfmark=/NamespacePop
2130 }%
2131 }

```

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.

```

2132 \newcommand*{\hyxmp@embed@packet@dvipdfm}{%
2133   \hyxmp@string@len{\hyxmp@xml}%
2134   \special{pdf: object @hyxmp@Metadata
2135     <<
2136       /Type /Metadata
2137       /Subtype /XML
2138       /Length \the\@tempcnta
2139     >>
2140     stream~J\hyxmp@xml endstream%
2141   }%
2142   \special{pdf: docview
2143     <<
2144       /Metadata @hyxmp@Metadata
2145     >>
2146   }%
2147 }

```

`\hyxmp@string@len` Set `\@tempcnta` to the number of characters in a given string (`#1`). The approach is first to tally the number of space characters then to tally the number of non-space characters. While this is rather sloppy I haven't found a better way to achieve

the same effect, especially given that all of the characters in #1 have already been assigned their category codes.

```
2148 \newcommand*{\hyxmp@string@len}[1]{%
2149   \@tempcnta=0
2150   \expandafter\hyxmp@count@spaces#1 {} %
2151   \expandafter\hyxmp@count@non@spaces#1{}%
2152 }
```

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

```
2153 \def\hyxmp@count@spaces#1 {%
2154   \def\hyxmp@one@token{#1}%
2155   \ifx\hyxmp@one@token\empty
2156     \advance\@tempcnta by -1
2157   \else
2158     \advance\@tempcnta by 1
2159   \expandafter\hyxmp@count@spaces
2160   \fi
2161 }
```

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

```
2162 \newcommand*{\hyxmp@count@non@spaces}[1]{%
2163   \def\hyxmp@one@token{#1}%
2164   \ifx\hyxmp@one@token\empty
2165   \else
2166     \advance\@tempcnta by 1
2167   \expandafter\hyxmp@count@non@spaces
2168   \fi
2169 }
```

3.7.5 Embedding using X_qT_EX

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

```
2170 \newcommand*{\hyxmp@embed@packet@xetex}{%
2171   \special{pdf:stream @hyxmp@Metadata (\hyxmp@xml)
2172     <<
2173       /Type /Metadata
2174       /Subtype /XML
2175     >>
2176   }%
2177   \special{pdf:put @catalog
2178     <<
2179     /Metadata @hyxmp@Metadata
```

```

2180     >>
2181   }%
2182 }

```

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.

```

2183 \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.

```

2184 \catcode'\="=\hyxmp@dq@code

```

4 Help Wanted

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

A Sample XMP Packet

The following is an example of a complete XMP packet as may be produced by `hyperxmp`. This packet corresponds to the metadata included in the sample \LaTeX document presented on pages 10–11. For clarity, metadata values, either specified explicitly by the document or introduced automatically by `hyperxmp`, are colored blue.

```

<?xpacket begin="\357\273\277" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/">
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about=""
      xmlns:pdf="http://ns.adobe.com/pdf/1.3/"
      xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/"
      xmlns:xmp="http://ns.adobe.com/xap/1.0/"
      xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"
      xmlns:stEvt="http://ns.adobe.com/xap/1.0/sType/ResourceEvent#"
      xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/"
      xmlns:pdfuaid="http://www.aiim.org/pdfua/ns/id/"
      xmlns:pdfx="http://ns.adobe.com/pdfx/1.3/"
      xmlns:pdfxid="http://www.npes.org/pdfx/ns/id/"
      xmlns:prism="http://prismstandard.org/namespaces/basic/3.0/"

```

```

xmlns:jav="http://www.niso.org/schemas/jav/1.0/"
xmlns:xmpTPg="http://ns.adobe.com/xap/1.0/t/pg/"
xmlns:stFnt="http://ns.adobe.com/xap/1.0/sType/Font#"
xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"
xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"
xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema#"
xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property#"
xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type#"
xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field#"
<pdfaExtension:schemas>
  <rdf:Bag>

```

⋮
[over 200 lines of boilerplate definitions not shown]
 ⋮

```

</rdf:Bag>
</pdfaExtension:schemas>
<pdf:Keywords>
  energy quanta, Hertz effect, quantum physics
</pdf:Keywords>
<pdf:Producer>
  LuaHBTeX, Version 1.12.0 (TeX Live 2020)
</pdf:Producer>
<pdf:PDFVersion>1.5</pdf:PDFVersion>
<xmpRights:Marked>True</xmpRights:Marked>
<xmpRights:WebStatement>
  http://creativecommons.org/licenses/by-nc-nd/3.0/
</xmpRights:WebStatement>
<dc:format>application/pdf</dc:format>
<dc:title>
  <rdf:Alt>
    <rdf:li xml:lang="x-default">
      On a heuristic viewpoint concerning the production
      and transformation of light
    </rdf:li>
    <rdf:li xml:lang="en">
      On a heuristic viewpoint concerning the production
      and transformation of light
    </rdf:li>
    <rdf:li xml:lang="de">
      Über einen die Erzeugung und Verwandlung des Lichtes
      betreffenden heuristischen Gesichtspunkt
    </rdf:li>
  </rdf:Alt>
</dc:title>
<dc:description>
  <rdf:Alt>
    <rdf:li xml:lang="en">photoelectric effect</rdf:li>

```

```

    </rdf:Alt>
</dc:description>
<dc:rights>
  <rdf:Alt>
    <rdf:li xml:lang="en">Copyright (C) 1905, Albert Einstein</rdf:li>
  </rdf:Alt>
</dc:rights>
<dc:publisher>
  <rdf:Bag>
    <rdf:li>Wiley-VCH</rdf:li>
  </rdf:Bag>
</dc:publisher>
<dc:creator>
  <rdf:Seq>
    <rdf:li>Albert Einstein</rdf:li>
  </rdf:Seq>
</dc:creator>
<dc:subject>
  <rdf:Bag>
    <rdf:li>energy quanta</rdf:li>
    <rdf:li>Hertz effect</rdf:li>
    <rdf:li>quantum physics</rdf:li>
  </rdf:Bag>
</dc:subject>
<dc:date>
  <rdf:Seq>
    <rdf:li>1905-03-17</rdf:li>
  </rdf:Seq>
</dc:date>
<dc:language>
  <rdf:Bag>
    <rdf:li>en</rdf:li>
  </rdf:Bag>
</dc:language>
<dc:type>
  <rdf:Bag>
    <rdf:li>Text</rdf:li>
  </rdf:Bag>
</dc:type>
<dc:source>einstein.tex</dc:source>
<dc:identifier>info:lccn/50013519</dc:identifier>
<photoshop:AuthorsPosition>
  Technical Assistant, Level III
</photoshop:AuthorsPosition>
<photoshop:CaptionWriter>Scott Pakin</photoshop:CaptionWriter>
<xmp:CreateDate>2020-07-25T21:37:02-06:00</xmp:CreateDate>
<xmp:ModifyDate>2020-07-25T21:37:02-06:00</xmp:ModifyDate>
<xmp:MetadataDate>2020-07-25T21:37:02-06:00</xmp:MetadataDate>
<xmp:CreatorTool>LaTeX with hyperref package</xmp:CreatorTool>
<xmpMM:DocumentID>

```

```

        uuid:6d1ac9ec-4ff2-515a-954b-648eeb4853b0
</xmpMM:DocumentID>
<xmpMM:InstanceID>
    uuid:3e4c4182-b182-46c9-995f-754c41d13390
</xmpMM:InstanceID>
<xmpMM:VersionID>2.998e8</xmpMM:VersionID>
<xmpMM:RenditionClass>default</xmpMM:RenditionClass>
<Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">
    <Iptc4xmpCore:CiAdrExtadr>Kramgasse 49</Iptc4xmpCore:CiAdrExtadr>
    <Iptc4xmpCore:CiAdrCity>Bern</Iptc4xmpCore:CiAdrCity>
    <Iptc4xmpCore:CiAdrPcode>3011</Iptc4xmpCore:CiAdrPcode>
    <Iptc4xmpCore:CiAdrCtry>Switzerland</Iptc4xmpCore:CiAdrCtry>
    <Iptc4xmpCore:CiTelWork>031 312 00 91</Iptc4xmpCore:CiTelWork>
    <Iptc4xmpCore:CiEmailWork>aeinstein@ipi.ch</Iptc4xmpCore:CiEmailWork>
    <Iptc4xmpCore:CiUrlWork>
        http://einstein.biz/,
        https://www.facebook.com/AlbertEinstein
    </Iptc4xmpCore:CiUrlWork>
</Iptc4xmpCore:CreatorContactInfo>
<prism:complianceProfile>three</prism:complianceProfile>
<prism:subtitle xml:lang="en-US">
    Putting that bum Maxwell in his place
</prism:subtitle>
<prism:publicationName xml:lang="de">
    Annalen der Physik
</prism:publicationName>
<prism:aggregationType>journal</prism:aggregationType>
<prism:volume>322</prism:volume>
<prism:number>6</prism:number>
<prism:pageRange>132-148</prism:pageRange>
<prism:issn>0003-3804</prism:issn>
<prism:eIssn>1521-3889</prism:eIssn>
<prism:doi>10.1002/andp.19053220607</prism:doi>
<prism:url>
    http://www.physik.uni-augsburg.de/annalen/history/einstein-papers/190517132-148.pdf
</prism:url>
<prism:byteCount>41197</prism:byteCount>
<prism:pageCount>1</prism:pageCount>
<jav:journal_article_version>VoR</jav:journal_article_version>
<xmpTPg:Fonts>
    <rdf:Bag>
        <rdf:li rdf:parseType="Resource">
            <stFnt:fontFace>LMRoman10-Regular</stFnt:fontFace>
            <stFnt:fontFamily>LM Roman 10</stFnt:fontFamily>
            <stFnt:fontName>LMRoman10-Regular</stFnt:fontName>
            <stFnt:versionString>
                2.004;PS 2.004;hotconv 1.0.49;makeotf.lib2.0.14853
            </stFnt:versionString>
            <stFnt:fontFileName>lmroman10-regular.otf</stFnt:fontFileName>
            <stFnt:fontType>opentype</stFnt:fontType>

```

```

</rdf:li>
<rdf:li rdf:parseType="Resource">
  <stFnt:fontFace>LMRoman17-Regular</stFnt:fontFace>
  <stFnt:fontFamily>LM Roman 17</stFnt:fontFamily>
  <stFnt:fontName>LMRoman17-Regular</stFnt:fontName>
  <stFnt:versionString>
    2.004;PS 2.004;hotconv 1.0.49;makeotf.lib2.0.14853
  </stFnt:versionString>
  <stFnt:fontFileName>lmroman17-regular.otf</stFnt:fontFileName>
  <stFnt:fontType>opentype</stFnt:fontType>
</rdf:li>
<rdf:li rdf:parseType="Resource">
  <stFnt:fontFace>LMRoman12-Regular</stFnt:fontFace>
  <stFnt:fontFamily>LM Roman 12</stFnt:fontFamily>
  <stFnt:fontName>LMRoman12-Regular</stFnt:fontName>
  <stFnt:versionString>
    2.004;PS 2.004;hotconv 1.0.49;makeotf.lib2.0.14853
  </stFnt:versionString>
  <stFnt:fontFileName>lmroman12-regular.otf</stFnt:fontFileName>
  <stFnt:fontType>opentype</stFnt:fontType>
</rdf:li>
<rdf:li rdf:parseType="Resource">
  <stFnt:fontName>cmr12</stFnt:fontName>
</rdf:li>
<rdf:li rdf:parseType="Resource">
  <stFnt:fontName>cmr8</stFnt:fontName>
</rdf:li>
<rdf: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: Predefined XMP Properties in PDF/A-1*, March 20, 2008. Available from http://www.pdfa.org/wp-content/uploads/2011/08/tn0008_predefined_xmp_properties_in_pdfa-1_2008-03-20.pdf.
- [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-datetime, World Wide Web Consortium (W3C), September 15, 1997. Available from <http://www.w3.org/TR/NOTE-datetime>.

Change History

v1.0		X _Y TeX backend (xdvipdfmx) .. 1
General: Initial version	1	Added support for the
v1.1		Photoshop schema
\hyxmp@construct@packet:		Made the package compatible
Explicitly set the category		with <code>ngerman</code> . Thanks to
codes of characters $\langle EF \rangle$, $\langle BB \rangle$,		Tobias Mueller for the bug
and $\langle BF \rangle$ to “letter”. Thanks to		report.
Daniel Schömer for the bug		19
report	85	v1.3
v1.2		General: Introduced the
General: Added support for the		<code>pdfmetalang</code> package option,
		which enables an author to

	specify the language in which he wrote the document's metadata	34			
v1.4					
	\hyxmp@mm@schema: Renamed the xapMM namespace prefix to xmpMM	70		\hyxmp@photoshop@schema: Simplified using	
	\hyxmp@rdf@dc: Included metadata in the x-default language regardless of the specified metadata language	65		\hyxmp@add@simple	71
	\hyxmp@xmpRights@schema: Renamed the xapRights namespace prefix to xmpRights	69		\hyxmp@skiptorelax: Added by Heiko Oberdiek	54
v1.5				\hyxmp@skipzeros: Added by Heiko Oberdiek	53
	General: Made the XMP inclusion more robust. Thanks to Heiko Oberdiek for the bug report and suggested modifications.	19		\hyxmp@toxml: Added by Heiko Oberdiek	52
v2.0				Escaped parentheses written with pdfmarks to prevent dvips from line-wrapping the XMP packet	52
	\ProcessKeyvalOptions: Added this macro	30		\hyxmp@toxml@unicodetex: Added by Heiko Oberdiek	53
	\XMPTruncateList: Added this macro	43		\hyxmp@xetex@crap: Added by Heiko Oberdiek	53
	\hyxmp@ProcessKeyvalOptions: Added this macro	30		\hyxmp@xmlify: Completely rewritten by Heiko Oberdiek to better support Unicode-enabled T _E X programs	50
	\hyxmp@SpaceOther: Added by Heiko Oberdiek	54		\hyxmp@xmp@basic@schema: Added this macro	70
	\hyxmp@add@simple: Added this macro	56		\hyxmp@xmpRights@schema: Modified to include xmpRights:Marked only when pdfcopyright is specified and xmpRights:WebStatement only when pdflicenseurl is specified	69
	\hyxmp@add@to@xml: Updated also to replace commas	62		\hyxmp@zero: Added by Heiko Oberdiek	55
	\hyxmp@bom: Added by Heiko Oberdiek	85		\ifhyxmp@unicodetex: Added by Heiko Oberdiek	50
	\hyxmp@comma: Added this macro	43		\xmpcomma: Added this macro	43
	\hyxmp@construct@packet: Modified by Heiko Oberdiek to use an appropriate BOM representation via \hyxmp@bom	85		\xmpquote: Added this macro	43
	\hyxmp@crap@convert: Added by Heiko Oberdiek	54		General: Added support for the XMP Basic schema and miscellaneous other bits of metadata	1
	\hyxmp@crap@test: Added by Heiko Oberdiek	54		Heiko Oberdiek's major rewrite of the code to better support native-Unicode T _E X implementations (X _Ǝ T _E X and LuaT _E X)	1
	\hyxmp@dc@schema: Added support for dc:language and dc:source	68		New \AtBeginDocument code from Heiko Oberdiek to properly encode	
	\hyxmp@is@unicode: Added by Heiko Oberdiek	51		\@pdfmetalang	34
	\hyxmp@list@to@xml: Modified by Heiko Oberdiek to use the new Unicode-processing macros	67	v2.1	\hypersetup: Added this macro	30

\hyxmp@hypersetup: Added this macro	30	conform to the latest XMP specifications, a detail identified by Florian Breitwieser	68
\hyxmp@redefine@Hyp: Added this macro	29	\hyxmp@parse@time: Added this macro	45
General: Enabled hyperxmp and hyperref to be loaded in either order. This addresses a bug report by Yuri Donskoy	28	\hyxmp@parse@tz: Added this macro	45
v2.2		\hyxmp@parse@tz@char: Added this macro	45
\hyxmp@iptc@extensions: Added this macro to support PDF/A generation	81	\hyxmp@pdf@schema: Made \hyxmp@pdf@schema <i>always</i> include the Adobe PDF schema, even when empty. Florian Breitwieser noted that this is necessary for PDF/A-1b compliance	65
\hyxmp@iptc@schema: Added this macro	73	\hyxmp@pdf@to@xmp@date: Added this macro	44
\hyxmp@list@to@lines: Added this macro	72	\hyxmp@pdfa@id@schema: Added this macro	71
\xmpcomma: Changed the default from \relax to an ordinary comma	43	\hyxmp@today@xmp: Modified the code to parse the time and timezone from \pdfcreationdate, as proposed by Florian Breitwieser	48
\xmplinesep: Added this macro . .	72	\hyxmp@today@xmp@define: Added this macro	47
General: Added support for the IPTC Photo Metadata schema .	1	\hyxmp@xmp@to@pdf@date: Added this macro	45
v2.3		\xmptilde: Added this macro . . .	43
\hyxmp@iptc@extensions: Gave the lptc4xmpCore:CreatorContactInfo fields a unique pdfaType:prefix to better support conversion of the document to PDF/A	81	General: Added support for the PDF/A Identification schema, as requested by Florian Breitwieser	1
v2.3a		v2.5	
\hyxmp@detect@langs: Bug fix: Redefine \@pdflang as \@empty when hyperref has set it to \relax	41	\hyxmp@add@to@xml: Updated also to replace underscores and to modify only the text being added, not the already-modified text	62
v2.3b		\hyxmp@textunderscore: Added this macro	21
\XMPTruncateList: Made all definitions local to avoid spurious Too many unprocessed floats errors when running with memoir . .	43	\hyxmp@uscore: Added this macro	43
v2.4		General: Enabled “_” to work within email addresses, as requested by Leonid Sinev	1
\hyxmp@add@simple@var: Added this macro	56	v2.6	
\hyxmp@create@uuid: Modified this macro to produce a proper version 4 (random or pseudorandom) UUID	60	General: Added support for a new pdfdate key to explicitly specify	
\hyxmp@dc@schema: Made dc:language a Bag instead of an individual item so as to			

the document date (and optionally time)	1	Robert Schlicht, Leonid Sinev, and David Carlisle for bug reports and to Leonid Sinev for helping test the new hyperxmp code	1
v2.7		v3.1	
\hyxmp@auto@assign@data:		\hyxmp@embed@packet@luatex:	
Automatically use \title and \author if pdftitle and pdfauthor are left unspecified. Thanks to Maciej Radziejewski for the suggestion	36	Updated to use \pdfextension obj uncompressed as suggested by Hans Hagen	88
v2.8		\hyxmp@embed@packet@pdftex:	
\hyxmp@add@to+xml: Corrected inadvertent lowercasing of non-Latin characters when run under X _q L ^A T _E X or Lua ^A T _E X (bug reported by Leonid Sinev)	62	Leave the XMP packet—and only the XMP packet—uncompressed in both pdfT _E X and pre-0.85 LuaT _E X	88
v2.9		v3.2	
\hyxmp@iptc@schema: Use lptc4xmpCore instead of lptc4ContInfo as the contact-information metadata prefix. Leonid Sinev reports that Acrobat’s PDF/A validator seems to prefer lptc4xmpCore	73	\hyxmp@as@pdf@date: Added this macro	45
\hyxmp@pdfa@id@schema: Let the author specify the PDF/A part and conformance IDs, as requested by Leonid Sinev	71	\hyxmp@as@xmp@date: Added this macro	44
General: Force inclusion of dc:creator, dc:title, and dc:description—even if empty—when hyperref is loaded with the pdfa option (suggested by Leonid Sinev)	1	\hyxmp@today@xmp@define: Modified to include hours and minutes	47
Introduced the pdftype package option, which enables an author to specify the type of document being produced	1	\hyxmp@xmp@basic@schema: Honor hyperref’s pdfcreationdate and pdfmoddate options plus a new pdfmetadate option. Leonid Sinev requested this additional control and helped test the resulting hyperxmp code	70
v3.0		v3.3	
\hyxmp@embed@packet@luatex: Added this macro	88	\@pdfsource: Added this macro and the corresponding pdfsource option, at Niklas Beisert’s request	24
\hyxmp@today@xmp@define: Modified to accept the name of a macro to define	47	\XMPLangAlt: Added this macro based on a request—and some code—by Niklas Beisert to support metadata expressed in multiple languages	58
\hyxmp@xmp@basic@schema: Made the XMP xmp:CreateDate, xmp:ModifyDate, and xmp:MetadataDate match the PDF CreationDate	70	\hyxmp@rdf@dc: Bug fix: Output the metadata language as correct XML even when hyperref is loaded with the unicode option	65
General: Made the code compatible with LuaT _E X 0.85. Thanks to		General: Don’t overwrite an existing pdfmetalang with pdflang or x-default. This addresses a bug report by Niklas Beisert	34

v3.4		from here into	
	<code>\hyxmp@seed@string</code> : Correctly handle an author field of all spaces. Bug reported by Gaëtan Leurent	<code>\hyxmp@check@iptc@data</code> . . .	73
	General: Use <code>ifmtarg</code> to test for empty arguments, including non-empty but all spaces	Renamed this macro to <code>\hyxmp@iptc@schema</code> from <code>\hyxmp@photometa@schema</code> . .	73
		Rewrote this macro entirely to correct the use of fields within a structure	73
v3.5		<code>\hyxmp@mm@extensions</code> : Added this macro	79
	<code>\hyxmp@DocumentID</code> : Added the <code>pdfdocumentid</code> option, at Michael Osipov's request	<code>\hyxmp@mm@schema</code> : Include <code>xmpMM:VersionID</code> in the XMP packet	70
	<code>\hyxmp@InstanceID</code> : Added the <code>pdfinstanceid</code> option, at Michael Osipov's request	<code>\hyxmp@no@info@lists</code> : Added this macro	28
	<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	<code>\hyxmp@pdfa@id@extensions</code> : Added this macro	79
	<code>\hyxmp@seed@string</code> : Seed with the T _E X timestamp in addition to the document-specified timestamp	<code>\hyxmp@prism@extensions</code> : Added this macro	82
		<code>\hyxmp@prism@schema</code> : Added this macro	74
		General: Include all metadata within a single <code>rdf:Description</code> block	1
		v4.1	
		<code>\hyxmp@singleton@dc</code> : Added this macro	68
v4.0		General: Invoke <code>\hyxmp@no@info@lists</code> at the beginning of the document, for compatibility with both newer and older versions of <code>hyperref</code> .	33
	<code>\XMPTruncateList</code> : Deprecated this macro	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@add@simple@lang</code> : Added this macro		
	<code>\hyxmp@begin@ext@decl</code> : Added this macro	v5.0	
	<code>\hyxmp@declare@field</code> : Replaced <code>\hyxmp@declare@resource</code> with this macro	<code>\@pdfrendition</code> : Added the <code>pdfrendition</code> option	25
	<code>\hyxmp@declare@property</code> : Added this macro	<code>\@pdfxstandard</code> : Added this macro	23
	<code>\hyxmp@end@ext@decl</code> : Added this macro	<code>\hyxmp@add@simple</code> : Insert the tag name (#1) verbatim	56
	<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> .	<code>\hyxmp@check@standards</code> : Added this macro	32
		<code>\hyxmp@check@std</code> : Added this macro	23
	Rewrote to more closely honor the XMP specification	<code>\hyxmp@declare@property</code> : Insert the property name (#2) verbatim	78
	<code>\hyxmp@iptc@schema</code> : Moved the definition of <code>\hyxmp@iptc@data</code>		

<code>\hyxmp@define@pdfproducer:</code>	Robin Schwab for the bug
Added this macro 63	report 63
<code>\hyxmp@no@info@lists:</code> Renamed	<code>\hyxmp@timestamp:</code> Don't rely on
this macros from	<code>\jobname.aux</code> existing to query
<code>\hyxmp@suppress@pdf@metadata</code>	the current time under \LaTeX .
and rewrote it to replace, if	Instead, use <code>\jobname.log</code> .
possible, only Author and	Thanks to Ulrike Fischer for
Keywords 28	the bug report and for her
<code>\hyxmp@pdf@extensions:</code> Added	suggestion to use the log file. . 48
this macro 79	v5.2
<code>\hyxmp@pdf@schema:</code> Honor	<code>\hyxmp@add@simple@pfx:</code> Added
<code>pdftrapped</code> 65	this macro 57
<code>\hyxmp@pdfua@id@extensions:</code>	<code>\hyxmp@assign@major@minor:</code>
Added this macro 80	Added this macro. <code>hyperxmp</code>
<code>\hyxmp@pdfua@id@schema:</code> Added	now correctly specifies
this macro 72	<code>pdf:PDFVersion</code> when
<code>\hyxmp@pdfx@id@extensions:</code>	generating PDF 2.0+. Thanks
Added this macro 80	to Ulrike Fischer for alerting
<code>\hyxmp@pdfx@id@schema:</code> Added	me to PDF 2.0's availability in
this macro 72	the \TeX ecosystem and
<code>\hyxmp@today@pdf:</code> Added this	informing me how to activate it 64
macro 49	<code>\hyxmp@cond@dc@identifier:</code>
<code>\hyxmp@today@xmp:</code> Support	Added this macro 68
\LaTeX 's <code>\filemoddate</code> 48	<code>\next:</code> Define <code>\ifdraft</code> only
<code>\hyxmp@today@xmp@define:</code>	locally, at Niklas Beisert's
Modified to specify UTC 47	request 24
General: Added support for	General: Introduced the
PDF/UA standards, as requested	<code>pdfidentifier</code> package option,
by Robin Schwab 1	which enables an author to
Added support for PDF/X	specify a unique identifier for
standards, as requested by	the document 1
Robin Schwab 1	v5.3
Define a default producer 64	<code>\@if@def@and@nonempty:</code> Added
Don't set any document dates	this macro 21
(creation, modification, or	<code>\hyxmp@at@end:</code> Use
metadata) from <code>pdfdate</code> 1	<code>\AtEndDocument</code> in all \TeX
v5.1	back ends that provide it.
<code>\hyxmp@banner@to@producer:</code>	Thanks to Nelson Posse Lago
Prevent the category code of	for pointing out why <code>atenddvi</code> is
"@" from propagating past the	best avoided if possible 19
<code>\begin{document}</code> . Thanks to	<code>\hyxmp@auto@assign@data:</code>
Robert Schlicht for noticing this	Consider other author-provided
catcode "leak" and providing a	sources of metadata. Thanks to
correction 63	Robin Schwab for proposing
<code>\hyxmp@define@pdfproducer:</code>	that <code>hyperxmp</code> use the KOMA
Check for \LaTeX before	letter classes's metadata 36
checking for \pdfTeX to work	<code>\hyxmp@dc@schema:</code> Include all
around \luatex85 's confusing	languages used in the document
<code>iftex</code> by defining	in <code>dc:language</code> 68
<code>\pdfTeXversion</code> . Thanks to	

\hyxmp@detect@langs: Acquire the default language from the polyglossia package, if loaded. Thanks to Robin Schwab for bringing that package to my attention	41	modifying babel for hyperxmp's benefit	41
\hyxmp@parse@acmart: Added this macro	38	\hyxmp@jav@extensions: Added this macro	84
\hyxmp@set@koma@phones: Added this macro	34	\hyxmp@jav@schema: Added this macro	74
\hyxmp@use@first@valid: Added this macro	35	\hyxmp@mm@extensions: Corrected the type of xmpMM:RenditionClass. Thanks to Thorsten Wißmann for the bug report and patch	79
v5.4		\hyxmp@query@self: Added this macro	37
\hyxmp@dc@schema: Bug fix: Use \hyxmp@today@xmp as the date only if \pdfdatetime is undefined	68	\hyxmp@rdf@dc: List x-default alternatives before language-specific alternatives, as dictated by the XMP specification [5]	65
\hyxmp@detect@langs: Added support for babel	41	Rewrite the core part of this macro to divide it into four, cleanly defined cases	65
Refactored language detection into a separate command	41	\hyxmp@set@koma@phones: Support hyperlinks and other markup in frommobilephone and fromphone, as requested by Robin Schwab	34
\hyxmp@parse@acmart: Bug fix: Correct a missing “else” argument in two invocations of \@if@def@and@nonempty	39	\hyxmp@xmptpg@schema: Added this macro	75
General: Moved the automatic assignment of \pdflang and \pdfmetalang from \hyxmp@auto@assign@data to within a call to \hyxmp@at@end	34	General: Automatically assign pdfnumpages and pdfbytes under pdfL ^A T _E X and LuaL ^A T _E X	1
v5.5		Copy \title to pdftitle and \author to pdfauthor at the start of the document to improve consistency between XMP and PDF metadata	33
\hyxmp@auto@assign@data: Moved the language-detection and X _q L ^A T _E X date-detection code here from the \hyxmp@at@end block	36	Correctly handle source files with spaces in their name. Thanks to Peter Dybala for the bug report	19
Moved title and author autodetection to the \AtEndPreamble	36	Defer \AtEndPreamble execution until the end of the document. This enables hyperxmp itself to be loaded from \AtEndPreamble, as is done by doclicense v2.2.0. Thanks to Tommaso Pecorella for the bug report and help testing	1
Use Lua _T _E _X mechanisms, when available, to automatically compute the page count	36	Introduced the pdfpubstatus package option, which enables	
\hyxmp@detect@langs: Set the language(s) immediately instead of deferring them to \hyxmp@set@dc@lang	41		
Store the main language in \pdflang. Thanks to Javier Bezos for his help with the hyperxmp code and for			

an author to specify the document's publication status.	v5.6	General: Don't inadvertently replace underscores in filenames when writing font-related metadata	75
Thanks to Robin Schwab for pointing me to the Journal Article Versions recommendation [1]	1	Make <code>write_xmp_font_list</code> robust to fonts loaded using HarfBuzz. Thanks to John Lienhard for the bug report . .	75
Load <code>hyperref</code> automatically if the document does not do so explicitly, as requested by Robin Schwab	33	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
Move most of the <code>\AtEndPreamble</code> code to <code>\hyxmp@at@end</code>	34		

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>\#</code>	1220, 1502	<code>\@ifnextchar</code> .. 10, 1056 308, 521, 1534, 1635
<code>\&</code>	896, 928, 1501	<code>\@ifnotmtarg</code> <code>\@pdfcontactcity</code> <u>193</u> , 28, 75, 1050, 1059 309, 527, 1535, 1636
<code>\@acmBooktitle</code>	585	<code>\@ifnotmtargexp</code> <code>\@pdfcontactcountry</code> <u>27</u> , 368, <u>199</u> , 432, 1032, 1073, 310, 539, 1538, 1639
<code>\@acmConference</code> . . .	586	<code>\@pdfcontactemail</code> <u>203</u> , 311, 432, 1032, 1073, 453, 515, 1541, 1641
<code>\@acmDOI</code>	556, 557, 559	<code>\@pdfcontactphone</code> <u>201</u> , 1371, 1384, 1506 312, 457, 1540, 1640
<code>\@acmISBN</code>	566, 567, 570	<code>\@pdfcontactpostcode</code> <u>197</u> , 313, 545, 1537, 1638
<code>\@acmNumber</code>	600	<code>\@pdfcontactregion</code> <u>195</u> , 314, 533, 1536, 1637
<code>\@acmVolume</code>	597	<code>\@pdfcontacturl</code> <u>205</u> , 315, 460, 1542, 1642
<code>\@author</code>	391, <u>512</u>	<code>\@pdfcopyright</code> <u>62</u> , 316, 1393, 1428, 1434
<code>\@baseurl</code>	302, 1468	<code>\@pdfcreationdate</code> 317, 471, 472, 1450, 1454
<code>\@elt</code> <u>658</u> , <u>1355</u> , 1513, <u>1518</u>		
<code>\@elt@first</code>	<u>1511</u>	
<code>\@elt@rest</code>	1513, <u>1515</u>	
<code>\@if@def@and@nonempty</code>	<u>29</u> , 405, 406, 412, 441, 498, 555, 565, 1298	
<code>\@ifclassloaded</code> . . .	603	
<code>\@ifmtarg</code>	27, 1061	
<code>\@ifmtargexp</code>	<u>27</u> , 31, 352, 369, 420, 471, 797, 1287, 1345, 1395, 1412, 1442, 1443, 1450, 1456, 1462	
<code>\@pdfauthor</code> 218, <u>242</u> , 303, 389, 1185, 1193		
<code>\@pdfauthor</code>	<u>68</u> , 304, 1471, 1472	
<code>\@pdfbookedition</code>	<u>165</u> , 305, 1557, 1647	
<code>\@pdfbytes</code>	<u>155</u> , 306, 506, 1566, 1648	
<code>\@pdfcaptionwriter</code>	<u>70</u> , 307, 1471, 1473	
<code>\@pdfcontactaddress</code>	<u>185</u> , 471, 472, 1450, 1454	

<code>\@pdfcreator</code>	1467	<code>\@pdfsource</code>		Author	12, 13, 28, 102
<code>\@pdfdatetime</code> 125 , 1403, 1405		
	40 , 318, 1395, 1398	<code>\@pdfsubject</code>	334, 1392		
<code>\@pdfdoi</code>	175 ,	<code>\@pdfsubtitle</code>	181 ,		
	319, 558, 1408,		335, 463, 1554, 1658		
	1414, 1564, 1649	<code>\@pdftitle</code>			
<code>\@pdfissn</code> 336, 369, 385,		
	161 , 320, 1409,		1185, 1193, 1391		
	1415, 1563, 1650	<code>\@pdftrapped</code>	1281		
<code>\@pdfidentifier</code>		<code>\@pdftype</code>	64 , 1400		
	.. 179 , 321,	<code>\@pdfupart</code>	88 ,		
	1408, 1412, 1419		337, 366, 1482, 1960		
<code>\@pdfisbn</code>	163 ,	<code>\@pdfurl</code>			
	322, 572, 1409,		177 , 338, 1565, 1659		
	1417, 1561, 1651	<code>\@pdfversionid</code> 131 , 1446			
<code>\@pdfissn</code>		<code>\@pdfvolumenum</code>	169 ,		
	159 , 323, 1409,		339, 596, 1558, 1660		
	1416, 1562, 1652	<code>\@pdfxstandard</code>			
<code>\@pdfissuenum</code>	171 ,		. 106 , 340, 365,		
	324, 599, 1559, 1653		1490, 1493, 1495		
<code>\@pdfkeywords</code>		<code>\@pdfxversion</code>	1964		
	.. 224 , 263 , 325	<code>\@publishers</code>	467		
<code>\@pdflang</code>		<code>\@subtitle</code>	464		
	. 326, 442, 447,	<code>\@tempswafalse</code> 1287, 1345			
	450, 608, 610 , 625	<code>\@tempswatru</code> . 1287,			
<code>\@pdflicenseurl</code>			1289, 1345, 1347		
	66 , 327, 1424, 1438	<code>\@title</code>	387		
<code>\@pdfmetadatetime</code>		<code>\^</code> 646, 650, 833, 1213,			
	51 , 328, 1462, 1465		1214, 1989–1991		
<code>\@pdfmetalang</code> 72 , 446,		<code>_</code>	1212, 1214, 1583		
	448, 450, 1056,	<code>\~</code>	655, 966, 967		
	1303, 1319, 1332				
<code>\@pdfmoddate</code>					
	.. 329, 1456, 1460	<code>_</code>	932, 967, 1212		
<code>\@pdfnumpages</code>	157 ,				
	330, 501, 504,				
	1567, 1576, 1654				
<code>\@pdfpagerange</code>					
	173 , 331, 1560, 1655				
<code>\@pdfproducer</code>	1236 ,				
	1250 , 1257 , 1259				
<code>\@pdfpublication</code> 151 ,					
	332, 583, 1555, 1656				
<code>\@pdfpublisher</code>					
	167 , 466, 580, 1394				
<code>\@pdfpubstatus</code>					
	.. 183 , 1570, 1665				
<code>\@pdfpubtype</code>	153 ,				
	333, 593, 1556, 1657				
<code>\@pdfrendition</code> 143 , 1447					

<code>\define@key</code>	41, 52, 63, 65, 67, 69, 71, 73, 80, 85, 89, 108, 126, 128, 130, 132, 150, 152, 154, 156, 158, 160, 162, 164, 166, 168, 170, 172, 174, 176, 178, 180, 182, 184, 186, 194, 196, 198, 200, 202, 204, 206, 230, 242, 263, 1088	<code>\Hy@unicodefalse</code> 43, 54, 404	1080, 1199 , 1225, 1294, 1313, 1320, 1326, 1333, 1338, 1350, 1358, 1364, 1373, 1508, 1512, 1516, 1522, 1529, 1544, 1669, 1675, 1681, 1691, 1698, 1702, 1710, 1823, 1862, 1997, 2042
<code>\do</code>	... 1090, 1097, 1324	<code>hyperref</code> (package) 1, 4–6, 9, 10, 13, 16, 18, 20–23, 28–30, 32, 33, 41, 42, 63, 64, 87, 88, 99–101, 104	
<code>doclicense</code> (package)	. 103	<code>\hypersetup</code>	280 , 433, 1311	
<code>DOI</code> 2, 7, 26, 39	<code>hyperxmp</code> (package)	1, 2, 4–10, 13–22, 24, 27–29, 31, 33–37, 41–44, 50, 59, 63–65, 76, 87, 91, 99, 100, 102–104	<code>\hyxmp@address@val</code> 513 , 519 , 525 , 531 , 537 , 543
<code>draft</code> (option) 9	<code>\hyxmp@@is@unicode</code>	. 876	<code>\hyxmp@ae@toks</code> 15 , 286, 287, 378, 379, 2183
Dublin Core schema 2, 61, 65–69	<code>\hyxmp@acm@isbn</code>	... 565	<code>\hyxmp@alt@description</code> 1084 , 1094
<code>dvipdf</code> (option) 88	<code>\hyxmp@acm@publisher</code> 579	<code>\hyxmp@alt@rights</code> 1084 , 1095
<code>dvipdfm</code> 89	<code>\hyxmp@acm@pubtype</code>	. 588	<code>\hyxmp@alt@title</code> 1084 , 1093
<code>dvips</code> (option) 88	<code>\hyxmp@add@simple</code> 1031 , 1281, 1390, 1405, 1419, 1436, 1438, 1444–1447, 1451, 1453, 1457, 1459, 1463, 1465, 1467, 1468, 1472, 1473, 1477, 1478, 1482, 1489, 1490, 1493, 1495, 1535–1538, 1552, 1556, 1558–1567, 1570, 1576	<code>\hyxmp@and</code> 242 <code>\hyxmp@append@hex</code> 1133 , 1152–1154, 1158
<code>dvipsone</code> (option) 88	<code>\hyxmp@add@simple@lang</code> 1049 , 1554, 1555, 1557	<code>\hyxmp@append@hex@iii</code> 1151 , 1157, 1167, 1178
<code>dviwindo</code> (option)	... 88	<code>\hyxmp@add@simple@lang@i</code> 1052 , 1055	<code>\hyxmp@append@hex@iv</code> 1156 , 1162, 1163, 1165, 1180–1182
E		<code>\hyxmp@add@simple@lang@ii</code> 1056 , 1058	<code>\hyxmp@as@pdf@date</code> . 702 <code>\hyxmp@as@xmp@date</code> 46, 57, 674, 812, 1454, 1460
ε -TEX 63	<code>\hyxmp@add@simple@pfx</code> 1072 , 1385	<code>\hyxmp@assign@major@minor</code> 1263 , 1282
<code>\EdefEscapeHex</code>	855, 868	<code>\hyxmp@add@simple@var</code> 1040 , 1279, 1280, 1283	<code>\hyxmp@at@end</code> ... 3 , 396
<code>\EdefUnescapeHex</code>	. 872	<code>\Hy@driver</code>	2053, 2057, 2061, 2065, 2070	<code>\hyxmp@auto@assign@data</code> 397, 440
<code>\EdefUnescapeString</code>	842			<code>\hyxmp@banner@to@producer</code> .. 1239, 1242, 1250
<code>\email</code> 513			<code>\hyxmp@begin@ext@decl</code> 1680 , 1719, 1730, 1756, 1772, 1786, 1802, 1814, 1872, 1942
<code>\empty</code> 1784			
<code>\equal</code> 102			
<code>etoolbox</code> (package)	. 20			
ETX 43, 50			
F				
<code>\filemoddate</code> 810			
G				
<code>\getlocaleproperty</code>	. 612			
Ghostscript 12			
<code>gitver</code> (package) 7			
H				
<code>\hbox</code> 553			

\hyxmp@begin@extension@decl	\hyxmp@declare@field	1034, 1045, 1074, 1509, 1533
..... 1668, 1954 1709,	
\hyxmp@big@prime ..	1838, 1841, 1844,	\hyxmp@first@char .. 672
..... 1107,	1847, 1850,	\hyxmp@first@char@i
1110, 1120, 1130	1853, 1856, 1859 672, 675, 703
\hyxmp@big@prime@ii	\hyxmp@declare@property	\hyxmp@gobbletwo 742, 755
..... 1107, 1129	... 1697, 1723,	\hyxmp@hash ... 1219,
\hyxmp@bom .. 1982, 1997	1734, 1739, 1744,	2001, 2009,
\hyxmp@cct .. 1574, 1578	1748, 1760, 1765,	2017, 2020–2023
\hyxmp@check@iptc@data	1776, 1790, 1794,	\hyxmp@Hyp@pdfauthor
..... 1633, 2025	1806, 1818, 1876, 236
\hyxmp@check@jav@data	1880, 1884, 1888,	\hyxmp@Hyp@pdfkeywords
..... 1663, 2027	1892, 1896, 1900, 257
\hyxmp@check@prism@data	1904, 1908, 1912,	\hyxmp@hypersetup .. 280
..... 1645, 2026	1916, 1920, 1925,	\hyxmp@InstanceID ..
\hyxmp@check@standards	1930, 1935, 1946 129,
..... 350, 398	\hyxmp@def@DocumentID	1190, 1443, 1445
\hyxmp@check@std 1184, 1442	\hyxmp@iprefix 1076, 1077
.... 101, 113–121	\hyxmp@def@InstanceID	\hyxmp@iptc@data ..
\hyxmp@comma 1190, 1443	.. 1527, 1633, 1968
. 187, 243, 264, 645	\hyxmp@define@pdfproducer	\hyxmp@iptc@extensions
\hyxmp@commas@to@list 1236, 1260 1813, 1970
629, 665, 1356, 1520	\hyxmp@detect@langs	\hyxmp@iptc@schema ..
\hyxmp@commas@to@list@i 444, 605 1526, 2038
..... 631, 633	\hyxmp@DocumentID ..	\hyxmp@is@unicode ..
\hyxmp@concat@metadata 127, 844, 861, 876
..... 285, 299	1184, 1442, 1444	\hyxmp@jav@data ...
\hyxmp@cond@dc@identifier	\hyxmp@dq@code . 1, 2184 1663, 1976
.. 1382, 1414–1417	\hyxmp@driver 2050	\hyxmp@jav@extensions
\hyxmp@construct@packet	\hyxmp@embed@packet 1941, 1978
..... 1995, 2051 400, 2050	\hyxmp@jav@schema ..
\hyxmp@count@non@spaces	\hyxmp@embed@packet@dvipdfm 1569, 2040
..... 2151, 2162 2062, 2132	\hyxmp@jobname
\hyxmp@count@spaces	\hyxmp@embed@packet@luatex	12, 13, 125, 344,
..... 2150, 2153 2058, 2095	485, 493, 810,
\hyxmp@crap@convert	\hyxmp@embed@packet@pdfmark	1185, 1193, 2071
..... 958, 992 2074, 2102	\hyxmp@koma@phones ..
\hyxmp@crap@result ..	\hyxmp@embed@packet@pdftex 402, 458
..... 948, 984 2054, 2082	\hyxmp@LA@accept ..
\hyxmp@crap@test 955, 980	\hyxmp@embed@packet@xetex	.. 1087, 1093–1095
\hyxmp@create@uuid 2066, 2170	\hyxmp@lang@name .. 610
.. 1160, 1188, 1197	\hyxmp@end@ext@decl	\hyxmp@lang@tag ... 610
\hyxmp@cur@lang 1690,	\hyxmp@legal 1423
..... 1090, 1098	1727, 1753, 1769,	\hyxmp@list ... 1356,
\hyxmp@dc@lang . 442,	1781, 1798,	1362, 1520, 1521
604, 610, 626, 1407	1810, 1939, 1951	\hyxmp@list@to@lines
\hyxmp@dc@schema ..	\hyxmp@end@extension@decls 1505,
..... 1389, 2031 1674, 1980	1534, 1540–1542
\hyxmp@declare@extensions	\hyxmp@extra@indent	
..... 1953, 2028 1030,	

\hyxmp@list@to+xml .	45, 56, 63, 65, 67,	\hyxmp@remove@this .
..... 1344,	69, 71, 73, 82, 1254, 1257
1401, 1402, 1407	86, 91, 109, 126,	\hyxmp@rights . 1423,
\hyxmp@major@minor 1263	128, 130, 132,	1426, 1430, 1432
\hyxmp@mm@extensions	150, 152, 154,	\hyxmp@seed@rng ...
..... 1729, 1956	156, 158, 160,	.. 1109, 1186, 1195
\hyxmp@mm@schema ..	162, 164, 166,	\hyxmp@seed@rng@i ..
..... 1441, 2037	168, 170, 172, 1111, 1113
\hyxmp@modulo@a ...	174, 176, 178,	\hyxmp@seed@string .
... 1101, 1120,	180, 182, 184, 1184, 1190
1130, 1136, 1171	189, 194, 196,	\hyxmp@set@jobname 9, 14
\hyxmp@multi@langsfalse	198, 200, 202,	\hyxmp@set@jobname@dbl
..... 1285, 1301	204, 206, 407, 10, 12
\hyxmp@multi@langstrue	409, 413, 1076, 1089	\hyxmp@set@jobname@plain
..... 1285, 1299	\hyxmp@pdfua@id@extensions 10, 13
\hyxmp@new+xml 1215, 1216 1771, 1962	\hyxmp@set@koma@phones
\hyxmp@no@bad@parts	\hyxmp@pdfua@id@schema 402, 456
..... 74, 81, 90 1481, 2035	\hyxmp@set@pdfx@major
\hyxmp@no@info@lists	\hyxmp@pdfx@id@extensions 93, 123
.... 207, 231, 382 1783, 1966	\hyxmp@set@pdfx@major@i
\hyxmp@num 992	\hyxmp@pdfx@id@schema 93, 94
\hyxmp@one@token 1484, 2036	\hyxmp@set@pdfx@major@ii
..... 1109,	\hyxmp@pdfx@major 95, 98
1113, 2154,	.. 98, 107, 123,	\hyxmp@set@rand@num
2155, 2163, 2164	1485, 1784, 1800	.. 1126, 1134, 1169
\hyxmp@padding 1223, 2046	\hyxmp@photoshop@data	\hyxmp@singleton@dc
\hyxmp@parse@acmart 1470	... 1370, 1394,
.... 469, 510, 603	\hyxmp@photoshop@schema	1396, 1398, 1400
\hyxmp@parse@time 1470, 2032	\hyxmp@skiptorelax .
..... 683, 685	\hyxmp@prev@pdf@size 985, 991
\hyxmp@parse@tz 483, 493, 507	\hyxmp@skipzeros .. 943
.... 692, 695, 699	\hyxmp@prism@data ..	\hyxmp@Space@other ..
\hyxmp@parse@tz@char	.. 1550, 1645, 1972 952, 965
..... 687, 689	\hyxmp@prism@extensions	\hyxmp@standards .. 363
\hyxmp@pdf@extensions 1871, 1974	\hyxmp@string@len ..
..... 1718, 1955	\hyxmp@prism@schema 2133, 2148
\hyxmp@pdf@schema 1549, 2039	\hyxmp@strip@isbn@date
..... 1278, 2029	\hyxmp@ProcessKeyvalOptions 565
\hyxmp@pdf@to@xmp@date 275	\hyxmp@sublist
. 676, 681, 804, 807	\hyxmp@prot@us ... 1582	. 634, 635, 638, 639
\hyxmp@pdfa@id@extensions	\hyxmp@query@self ..	\hyxmp@suppress@pdf@info
..... 1755, 1958 476, 496 208
\hyxmp@pdfa@id@schema	\hyxmp@rand@num ...	\hyxmp@temp@list .. 658
..... 1475, 2034	... 1126, 1135,	\hyxmp@temp@str ... 658
\hyxmp@pdfauthor ..	1170, 1187, 1196	\hyxmp@text
... 233, 242, 1401	\hyxmp@rdf@dc 840, 918, 948, 992
\hyxmp@pdfkeywords .	.. 1286, 1391-1393	\hyxmp@textunderscore
... 233, 263, 1402	\hyxmp@redefine@Hyp 34
\hyxmp@pdfstringdef 235, 277, 282	\hyxmp@timestamp .. 809
..... 34,		\hyxmp@today@pdf 472, 819

<code>\hyxmp@today@xmp</code> ..	1334, 1356, 1376,	<code>\ifLuaTeX</code> .. 24, 478,
..... 797, <u>802</u> ,	1383, 1413, 1520	482, 497, 1238,
820, 1193, 1396,	<code>\hyxmp@xmllify</code>	1573, 1578, 1586
1451, 1457, 1463 <u>840</u> , 1033,	<code>ifluatex</code> (package) ... 88
<code>\hyxmp@today@xmp@define</code>	1043, 1051, 1060,	<code>\ifluatex</code> .. 2084, 2088
... <u>768</u> , 817, 1191	1077, 1079, 1292,	<code>ifmtarg</code> (package) 20, 101
<code>\hyxmp@toxml</code> .. 870, <u>893</u>	1319, 1325, 1332,	<code>\ifPDFTeX</code> ... 492, 1241
<code>\hyxmp@toxml@unicodetex</code>	1355, 1372, 1519	<code>\IfSubStr</code>
..... 858, <u>918</u>	<code>\hyxmp@xmp@basic@schema</code>	. 556, 557, 566, 567
<code>\hyxmp@trimb</code> .. 826, <u>829</u> 1449, 2033	<code>iftex</code> (package) .. 20, 102
<code>\hyxmp@trimc</code> .. 829, <u>830</u>	<code>\hyxmp@xmp@to@pdf@date</code>	<code>ifthen</code> (package) 20
<code>\hyxmp@trimspaces</code> 706, <u>709</u> , 820	<code>\ifthenelse</code>
..... 638, <u>822</u>	<code>\hyxmp@xmp@to@pdf@date@i</code>	<code>\ifXeTeX</code> 848, 1244
<code>\hyxmp@try</code> 710, <u>712</u>	<code>Info</code> 12–14, 28, 33, 36, 63
..... <u>948</u>	<code>\hyxmp@xmp@to@pdf@date@iii</code>	<code>intcalc</code> (package) 20
<code>\hyxmp@try@today</code> 796, 715, <u>718</u>	<code>\intcalcDiv</code>
803, 806, 809, 816	<code>\hyxmp@xmp@to@pdf@date@iii</code>	... 997, 1004, 1011
<code>\hyxmp@unicodetexfalse</code> 721, <u>724</u>	<code>\intcalcMod</code>
..... <u>832</u>	<code>\hyxmp@xmp@to@pdf@date@iv</code>	.. 999, 1006, 1013
<code>\hyxmp@unicodetextrue</code> 727, <u>730</u>	<code>IPTC</code> 14, 27, 61, 73, 76,
..... <u>832</u>	<code>\hyxmp@xmp@to@pdf@date@v</code>	81, 84, 99, 109, 113
<code>\hyxmp@uscore</code> .. 36, <u>649</u> 733, <u>736</u>	<code>IPTC Photo Metadata</code>
<code>\hyxmp@use@first@valid</code>	<code>\hyxmp@xmp@to@pdf@date@vi</code>	schema . 61, 72–74
. 385, 389, <u>419</u> , 739, <u>743</u>	<code>lptc4xmpCore:Contact-</code>
453, 457, 460,	<code>\hyxmp@xmp@to@pdf@date@vii</code>	<code>Info</code>
463, 466, 506, 746, 749, <u>759</u>	<code>lptc4xmpCore:Creator-</code>
515, 521, 527,	<code>\hyxmp@xmp@to@pdf@date@viii</code>	<code>ContactInfo</code> ...
533, 539, 545, 762, <u>765</u> 2, 3, 73, 99
558, 572, 580,	<code>\hyxmp@xmpRights@schema</code>	<code>ISBN</code>
583, 593, 596, 599 1422, 2030	2, 7, 26, 40
<code>\hyxmp@use@first@valid@i</code>	<code>\hyxmp@xmptpg@schema</code>	<code>ISO</code> 6, 7, 16, 22, 57
..... 421, <u>425</u> 1572, 2041	<code>ISSN</code>
<code>\hyxmp@value</code> <u>1089</u> , <u>1293</u>	<code>\hyxmp@zero</code>	2, 7, 25
<code>\hyxmp@warn@if@no@metadata</code>	... 1001, 1008,	J
..... <u>299</u> , 399	1015, 1021, <u>1026</u>	<code>JAV</code>
<code>\hyxmp@x@default</code> ..		77, 84, 85
.... 448, <u>1235</u> ,	I	<code>jav:journal_article_ver-</code>
1303, 1314, 1321	<code>IETF</code>	<code>sion</code>
<code>\hyxmp@xetex@crap</code> ..	6	3
..... 849, <u>948</u>	<code>\if@ACM@journal</code> ... 588	<code>\jobname</code>
<code>\hyxmp@xml</code> 1035, 1037,	<code>\if@tempwa</code> . 1291, 1349	14
1075, 1081, 1216,	<code>ifdraft</code> (package) 18, 24, 104	<code>Journal Article Versions</code>
<u>1223</u> , 1701, <u>1995</u> ,	<code>\ifdraft</code> <u>133</u> , 143	schema ... 61, 74
2091, 2099, 2122,	<code>\iffalse</code> ... 1286, 1344	K
2133, 2140, 2171	<code>\ifHy@pdfa</code>	<code>keeppdfinfo</code> (option) 14, 28
<code>\hyxmp@xmllified</code> ...	351, 1391, 1392,	<code>Keywords</code> 12, 13, 28, 65, 102
.... <u>840</u> , 1036,	1401, 1476, 1957	<code>Koma</code> (class) 16, 34, 102
1045, 1052, 1063,	<code>\ifhyxmp@multi@langs</code>	<code>\KV@Hyp@pdfauthor</code> .. <u>242</u>
1067, 1078, 1080,	.. <u>1285</u> , 1304, 1318	<code>\KV@Hyp@pdfkeywords</code> <u>263</u>
1293, 1322, 1327,	<code>\ifhyxmp@unicodetex</code>	<code>kvoptions</code> (package) 20, 30
	<u>832</u> , 843, 1202, 1983	L

`\LocaleForEach` 611
`Lua` 75
`lua`code (package) . . . 20
`\luadirect` 484, 501, 1574
`LuaLTeX` 10, 11, 14, 15,
 37, 48, 64, 100, 103
`LuaTeX` . . 20, 37, 38,
 50, 53, 75, 87, 88,
 91, 98, 100, 102, 103
`luatex85` (package) . . . 102
`\luatexbanner` 1239

M

`\makeatletter` 1254
`memoir` (package) . . . 99
`Metadata` 12, 87, 90
`\month` 770, 771, 773

N

`NAK` 21, 43, 50
`nativepdf` (option) . . . 88
`\newcatcodetable` . 1579
`\newif` 832, 1285
`\newtoks` 15
`\next` . . . [44](#), [55](#), 103,
 [110](#), [133](#), 148,
 [208](#), 426, 428,
 434, 438, [633](#),
 811, 814, [1113](#), [1964](#)
`ngerman` (package) 19, 97
`NISO` 8, 74
`\number` 995,
 997, 999, 1004,
 1006, 1011, 1013
`\numexpr` 2100

O

`options`
 `baseurl`
 5, 7, 16, 20, 26, 70
 `draft` 9
 `dvipdf` 88
 `dvips` 88
 `dvipsone` 88
 `dviwindo` 88
 `keeppdfinfo` . . . 14, 28
 `nativepdf` 88
 `pdfa` 9, 22, 23, 32, 100
 `pdfaconformance` .
 5, 9, 71

`pdfapart` . 5, 9, 22, 71
`pdfauthor` . . 5, 6,
 13, 15, 16, 20, 28,
 29, 33, 68, 100, 103
`pdfauthortitle` . 5, 6, 16
`pdfbookedition` . . 5, 8
`pdfbytes` 5, 10, 38, 103
`pdfcaptionwriter` . 5, 6
`pdfcontactaddress` .
 5, 6, 14
`pdfcontactcity` . . 5, 6
`pdfcontactcountry` 5, 6
`pdfcontactemail` . 5, 6
`pdfcontactphone` . 5, 6
`pdfcontactpostcode` .
 5, 6
`pdfcontactregion` . 5, 6
`pdfcontacturl` . 5, 6, 16
`pdfcopyright`
 . . . 5, 6, 68, 69, 98
`pdfcreationdate` . .
 5, 9, 36, 100
`pdfdate` 5, 8, 9, 16,
 21, 61, 68, 99, 102
`pdfdocumentid` 5, 7, 101
`pdfdoi` 5, 7
`pdffeissn` 5, 7
`pdfidentifier` 5, 7, 68, 102
`pdfinstanceid` 5, 7, 101
`pdfisbn` 5, 7
`pdfissn` 5, 7
`pdfissuenum` 5, 8
`pdfkeywords` . . . 5,
 13, 15, 20, 28, 29, 68
`pdflang` . . 5–7, 16,
 20, 41, 57, 68, 100
`pdflicenseurl`
 . . . 5, 6, 16, 69, 98
`pdfmark` 88
`pdfmetadate`
 5, 9, 21, 100
`pdfmetalang` 5, 6, 16,
 57, 66, 67, 97, 100
`pdfmoddate` . 5, 9, 100
`pdfnumpages` 5, 10, 103
`pdfpagerange`
 5, 8, 17, 18
`pdfproducer` . 5, 20, 64
`pdfpublication` . . . 5–8
`pdfpublisher` 5, 8

`pdfpubstatus` 5, 8, 103
`pdfpubtype` 5, 8
`pdfrendition` . 5, 9, 101
`pdfsource` . . 5, 10, 100
`pdfsubject` 5, 13, 20, 68
`pdfsubtitle` 5, 6
`pdftitle` 5, 6, 13, 16,
 20, 33, 68, 100, 103
`pdftrapped` 5, 9, 20, 102
`pdftype` . 6, 9, 68, 100
`pdfupart` . 6, 9, 22, 72
`pdfurl` 6, 7
`pdfversionid` . . 6, 7, 70
`pdfvolumenum` . . 6, 8
`pdfxstandard`
 . . . 6, 9, 23, 24, 72
`ps2pdf` 88
`textures` 88
`unicode` 16, 100
`vtexpdfmark` 88

P

`\PackageError` 1305
`packages`
 `atenddv`i 19, 102
 `babel` 16,
 18, 31, 35, 41, 103
 `doclicense` 103
 `etoolbox` 20
 `gitver` 7
 `hyperref` . . 1, 4–6,
 9, 10, 13, 16, 18,
 20–23, 28–30, 32,
 33, 41, 42, 63, 64,
 87, 88, 99–101, 104
 `hyperxmp` 1,
 2, 4–10, 13–22,
 24, 27–29, 31,
 33–37, 41–44, 50,
 59, 63–65, 76, 87,
 91, 99, 100, 102–104
 `ifdraft` . . . 18, 24, 104
 `ifluatex` 88
 `ifmtarg` 20, 101
 `iftex` 20, 102
 `ifthen` 20
 `intcalc` 20
 `kvoptions` 20, 30
 `luacode` 20
 `luatex85` 102

memoir	99	pdf:PDFVersion	3, 64, 102	pdfdate (option)	5, 8, 9, 16,
ngerman	19, 97	pdf:Producer	3, 63–65	21, 61, 68, 99, 102	
pdfescape	20	pdf:trapped	3	PDFDocEncoding	28, 50, 51
pdfx	4, 5	\PDF@FinishDoc	209, 217, 223	pdfdocumentid (option)	5, 7, 101
polyglossia	16, 18, 31,	pdfa (option)	9, 22, 23, 32, 100	pdfdoi (option)	5, 7
34, 35, 41, 42, 103		pdfa conformance (op-	5, 9, 71	pdfdoisn (option)	5, 7
stringenc	20	tion)	5, 9, 71	pdfescape (package)	20
texdate	17	pdfaid:conformance	3	\pdfextension	2096, 2100
totpages	17, 18, 37, 38	pdfaid:part	3	\pdffeedback	807, 2100
xmpincl	4	pdfapart (option)	5, 9, 22, 71	\pdffilesize	493
\PackageWarning	76, 111, 659	pdfaType:prefix	99	pdfidentifier (option)	5, 7, 68, 102
\PackageWarningNoLine	210,	pdfauthor (option)	5, 6,	pdfinstanceid (option)	5, 7, 101
343, 353, 370, 2069		13, 15, 16, 20, 28,	29, 33, 68, 100, 103	pdfisbn (option)	5, 7
\patchcmd	216, 222	pdfauthor title (option)	5, 6, 16	pdfissn (option)	5, 7
PDF 1–5, 8, 9, 11–15, 17,		pdfbookedition (option)	5, 8	pdfissuenumber (option)	5, 8
19, 21, 28, 32–34,		pdfbytes (option)	5, 10, 38, 103	pdfkeywords (option)	13, 15, 20, 28, 29, 68
36, 42, 44, 45, 49,		pdfcaptionwriter (op-	5, 6	pdflang (option)	5–7, 16,
50, 52, 61, 63, 64,		tion)	5, 6	20, 41, 57, 68, 100	
75, 79, 84, 85, 87,		\pdfcatalog	2092	\pdflastobj	2092
88, 90, 99, 100,		\pdfcompresslevel	2086	pdfL ^A T _E X	4, 10, 11,
102, 103, 105, 113		pdfcontactaddress (op-	5, 6, 14	14, 37, 48, 64, 103	
Author	12, 13, 28, 102	tion)	5, 6, 14	pdflicenseurl (option)	5, 6, 16, 69, 98
CreationDate	36, 100	pdfcontactcity (option)	5, 6	\pdfmajorversion	1271
Info	12–14,	pdfcontactcountry (op-	5, 6	pdfmark (option)	88
28, 33, 36, 63		tion)	5, 6	\pdfmark	2103,
Keywords	12, 13, 28, 65, 102	pdfcontactemail (op-	5, 6	2106, 2110,	
Metadata	12, 87, 90	tion)	5, 6	2120, 2124, 2128	
Producer	65	pdfcontactphone (op-	5, 6	pdfmetadate (option)	5, 9, 21, 100
Subject	12, 13	tion)	5, 6	pdfmetalang (option)	5, 6, 16,
Title	12, 13	pdfcontactpostcode (op-	5, 6	57, 66, 67, 97, 100	
PDF/A	3, 9, 13, 14, 22,	tion)	5, 6	\pdfminorversion	1267
23, 28, 32, 63, 64,		pdfcontactregion (op-	5, 6	pdfmoddate (option)	5, 9, 100
71, 76, 79, 81, 82,		tion)	5, 6	pdfnumpages (option)	5, 10, 103
84, 99, 100, 111, 113		pdfcontacturl (option)	5, 6, 16	\pdfobj	2088
PDF/A Identification		pdfcopyright (option)	5, 6, 68, 69, 98	pdfpagerange (option)	5, 8, 17, 18
schema	61, 71–72	pdfcreationdate (option)	5, 9, 36, 100	pdfproducer (option)	5, 20, 64
PDF/UA	3, 9, 23, 32, 72,	\pdfcreationdate	804	pdfpublication (option)	5–8
80, 84, 102, 111, 113					
PDF/UA Identification					
schema	61, 71–72				
PDF/X	3,				
9, 23, 24, 32, 72,					
80, 84, 102, 111, 113					
PDF/X Identification					
schema	61, 71–72				
pdf:Keywords	3, 13, 64, 65				

pdfpublisher (option)	5, 8	prism:bookEdition	2	photoshop:Author-	
pdfpubstatus (option)		prism:byteCount	2	sPosition	3, 71
.....	5, 8, 103	prism:doi	2	photoshop:Caption-	
pdfpubtype (option)	5, 8	prism:elssn	2	Writer	3, 71
pdfrendition (option)		prism:isbn	2	prism:aggregation-	
.....	5, 9, 101	prism:issn	2	Type	3
pdfsource (option)	5, 10, 100	prism:number	2	prism:bookEdition	2
\pdfstringdef	37, 504	prism:pageCount	3	prism:byteCount	2
pdfsubject (option)		prism:pageRange	3	prism:doi	2
.....	5, 13, 20, 68	prism:publicationName	3	prism:elssn	2
pdfsubtitle (option)	5, 6	prism:subtitle	3	prism:isbn	2
pdfTeX	52,	prism:url	3	prism:issn	2
87, 88, 91, 100, 102		prism:volume	4	prism:number	2
\pdfTeXbanner	1242	\ProcessKeyvalOptions		prism:pageCount	3
pdftitle (option)		275	prism:pageRange	3
...	5, 6, 13, 16,	Producer	65	prism:publication-	
20, 33, 68, 100, 103		properties, XMP		Name	3
pdftrapped (option)		dc:creator	2, 13, 68, 100	prism:subtitle	3
.....	5, 9, 20, 102	dc:date	2, 68	prism:url	3
pdftype (option)		dc:description		prism:volume	4
.....	6, 9, 68, 100		3, 13, 57, 68, 100	xmp:BaseURL	2
pdfuaid:part	3	dc:format	2	xmp:CreateDate	
pdfuapart (option)		dc:identifier	3, 68	2, 36, 100
.....	6, 9, 22, 72	dc:language	3, 16,	xmp:CreatorTool	3
pdfurl (option)	6, 7	31, 68, 98, 99, 102		xmp:MetadataDate	
\pdfvariable	1275	dc:publisher	3	2, 100
pdfversionid (option)		dc:rights	2, 17, 57, 68	xmp:ModifyDate	2, 100
.....	6, 7, 70	dc:source	2, 68, 98	xmpMM:Documen-	
pdfvolumenum (option)		dc:subject	3, 68	tID	3, 58, 70, 84
.....	6, 8	dc:title	3, 13, 57, 68, 100	xmpMM:InstancelD	
pdfx (package)	4, 5	dc:type	3, 68	4, 58, 70, 84
pdfxid:GTS_PDFXVer-		lptc4xmpCore:Con-		xmpMM:Rendition-	
sion	3	tactInfo	73, 81	Class	3, 103
pdfxstandard (option)		lptc4xmpCore:Cre-		xmpMM:VersionID	
...	6, 9, 23, 24, 72	atorContact-		4, 70, 101
Photoshop schema	61, 71	Info	2, 3, 73, 99	xmpRights:Marked	
photoshop:AuthorsPosi-		jav:journal_arti-		2, 69, 98
tion	3, 71	cle_version	3	xmpRights:WebState-	
photoshop:Caption-		pdf:Keywords		ment	3, 69, 98
Writer	3, 71	3, 13, 64, 65	ps2pdf (option)	88
PI	61	pdf:PDFVersion			
polyglossia (package)		3, 64, 102	Q	
.....	16, 18, 31,	pdf:Producer	3, 63–65	\Q	822, 831
34, 35, 41, 42, 103		pdf:trapped	3		
\postcode	543	pdfaid:conformance	3	R	
PRISM	8, 61, 74,	pdfaid:part	3	RDF	67
77, 82, 85, 112, 113		pdfaType:prefix	99	rdf:Description	101
PRISM Basic Metadata		pdfuaid:part	3	rdf:li	2
schema	61, 74	pdfxid:GTS_PDFXVer-		rdf:Seq	2
prism:aggregationType	3	sion	3	\ref	504

renewcommand 276	\StringEncodingConvert 62, 64, 65, 67, 68,
\RequirePackage 845,	72, 74, 77, 85, 100
. . . 4, 16–23, 25,	851, 862, 865, 960
138, 381, 480, 2081	XMP 1, 2, 4–6, 8,
	9, 12–21, 24, 28,
	33, 34, 36, 38–45,
	48, 49, 52, 55–58,
	61–67, 69, 70, 75,
	76, 79, 84, 87–91,
	98–101, 103, 113
	properties
	see properties, XMP
	XMP Basic schema 61, 70–71
	XMP Media Management
	schema . . . 61, 70
	XMP Paged-Text
	schema . 61, 75–76
	XMP Rights Manage-
	ment schema .
 61, 69–70
	xmp:BaseURL 2
	xmp:CreateDate 2, 36, 100
	xmp:CreatorTool 3
	xmp:MetadataDate . 2, 100
	xmp:ModifyDate . . 2, 100
	\xmpcomma 187,
	190, 242 , 263 , 644
	xmpincl (package) 4
	\XMLLangAlt . 1096 , 1306
	\xmplinesep
	. . 1500 , 1516, 1539
	xmpMM:DocumentID .
 3, 58, 70, 84
	xmpMM:InstanceID . .
 4, 58, 70, 84
	xmpMM:RenditionClass
 3, 103
	xmpMM:VersionID . . .
 4, 70, 101
	\xmpquote 188,
	191, 242 , 263 , 653
	xmpRights:Marked 2, 69, 98
	xmpRights:WebState-
	ment . . 3, 69, 98
	\xmptilde 654
	\XMPTruncateList . . 658
	\xpg@bcp@loaded . . . 626
	Y
	\year 769
S	T
\savecatcodetable . 1580	TeX 17, 19, 20, 47–50,
\scantokens . 1251, 1254	52, 53, 58, 59, 61,
schemata	63, 70, 90, 91, 102
Adobe PDF 61, 63–65	texdate (package) . . . 17
Dublin Core	Text 78
. 2, 61, 65–69	\textunderscore . . .
IPTC Photo Meta- 35, 36, 38
data . . . 61, 72–74	textures (option) 88
Journal Article Ver-	\time 780, 788
sions 61, 74	Title 12, 13
PDF/A Identifica-	totpages (package) . .
tion . . . 61, 71–72 17, 18, 37, 38
PDF/UA Identifica-	
tion . . . 61, 71–72	U
PDF/X Identifica-	\undefined 430
tion . . . 61, 71–72	Unicode 16, 20, 50–54,
Photoshop 61, 71	67, 73, 85, 91, 98
PRISM Basic Meta-	unicode (option) . 16, 100
data 61, 74	URI 7
XMP Basic . 61, 70–71	URL . . . 2, 3, 6, 7, 16,
XMP Media Manage-	22, 26, 28, 69–71, 73
ment 61, 70	UTF-16BE 51
XMP Paged-Text .	UTF-32BE 51
. 61, 75–76	UTF-8 51
XMP Rights Manage-	UUID 3, 4, 6,
ment . . 61, 69–70	7, 24, 58, 60, 61, 99
\scr@fromemail@var . 454	V
\scr@frommobilephone@var	\vfuzz 830
. 407, 409	vtexpdfmark (option) . 88
\scr@fromname@var . 390	
\scr@fromphone@var .	X
. 407, 413	\x 948
\scr@fromurl@var . . 461	x-default
\scr@subject@var . . 386	6, 16, 36, 57, 63,
scrItr2 (class) 16	66, 67, 98, 100, 103
\SE>pdfdoc03 838	xdvipdfmx . . . 15, 36, 90
\SE>pdfdoc15 839	X_qLaTeX 11, 15,
\setbox 553	36, 48, 64, 100, 102
\setkeys 1099	X_qTeX . . . 50, 53, 90,
\special 2134,	91, 97, 98, 102, 103
2142, 2171, 2177	\XeTeXrevision . . . 1245
\state 531	\XeTeXversion . . . 1245
\streetaddress 519	XML 1, 2, 14, 42,
stringenc (package) . . 20	50–53, 55–57, 61,