

The hyperxmp package*

Scott Pakin
scott+hyxmp@pakin.org

March 17, 2024

Abstract

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

1 Introduction

Adobe Systems, Inc. has been promoting XMP [5]—eXtensible Metadata Platform—as a standard way to include metadata within a document. The idea behind XMP is that it is an XML-based description of various document attributes and is embedded as uncompressed, unencoded text within the document it describes. By storing the metadata this way it is independent of the document's file format. That is, regardless of whether a document is in PDF, JPEG, HTML, or any other format, it is trivial for a program (or human) to locate, extract, and—using any standard XML parser—process the embedded XMP metadata.

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

This is too abstract! Give me an example. Consider a \LaTeX document with three authors—Jack Napier, Edward Nigma, and Harvey Dent—named in the \LaTeX source in the usual way: “`\author{Jack Napier \and Edward Nigma \and Harvey Dent}`”. With `hyperxmp`, the generated PDF file will contain, among other information, the following stanza of XMP code embedded within it:

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

*This document corresponds to `hyperxmp` v5.13, dated 2024/03/17.

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

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

1.1 Supported metadata

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

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

- page count (prism:pageCount)
- page range(s) (prism:pageRange)
- PDF version (pdf:PDFVersion)
- PDF-generating tool (pdf:Producer and xmp:CreatorTool)
- PDF/A version and conformance level (pdfaid:part and pdfaid:conformance)
- PDF/UA version (pdfuaid:part)
- PDF/X standard compliance (pdfxid:GTS_PDFXVersion)
- position/title of primary author (photoshop:AuthorsPosition)
- publication name of parent publication (prism:publicationName)
- publisher of the document (dc:publisher)
- rendition variation of the document (xmpMM:RenditionClass)
- summary (dc:description)
- subtitle (prism:subtitle)
- telephone number(s) of primary author
(lptc4xmpCore:CreatorContactInfo.CiTelWork)
- title (dc:title)
- trapping of colors (pdf:trapped)
- type of document (dc:type)
- type of parent publication (prism:aggregationType)
- unique identifier for the document (dc:identifier)
- URL of the document (prism:url)
- URL(s) of the primary author (lptc4xmpCore:CreatorContactInfo.CiUrlWork)
- UUID for the document (xmpMM:DocumentID)
- UUID for the document instance (xmpMM:InstanceID)
- version identifier for the document (xmpMM:VersionID)
- volume number of parent publication (prism:volume)

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

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

Figure 1: Comparison of pdfx and hyperxmp

1.2 Comparisons with similar packages

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

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

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

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

Figure 1 adapts an example appearing in the `pdfx` manual to `hyperxmp`. The two are comparable line-by-line in terms of how one specifies the title, author, document language, keywords, and publisher. However, `hyperxmp` implicitly writes a wealth of additional metadata into the XMP packet such as the document date, creation date, creator tool, file format, PDF version, and unique document and instance IDs. In fact, if a document omits all of the code shown in Figure 1(b), it will still store the `\title` and `\author` data in the XMP packet.

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

2 Usage

hyperxmp works by postprocessing some of the package options honored by hyperref. To use hyperxmp, merely put a `\usepackage{hyperxmp}` in your document's preamble. That line can appear anywhere *after* the `\usepackage{hyperref}` but *before* hyperref's PDF options are specified with `\hypersetup`. hyperxmp will construct its XMP data using the following hyperref options:

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

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

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

2.1 Option descriptions

`pdftitle` The document title is specified as normal for hyperref with `pdftitle`, but see Note 7 on page 15 for instructions on how to specify a title in multiple languages. If `pdftitle` is not specified it will inherit its value from the document's `\title`. hyperxmp introduces a complementary `pdfsubtitle` option:

```
pdftitle={Frankenstein},
```

`pdfsubtitle={The Modern Prometheus},`

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

`pdfauthor` `hyperref`'s `pdfauthor` option specifies the document's author(s). See Note 4 on page 14 for a discussion of the correct syntax. If `pdfauthor` is not specified it will inherit its value from the document's `\author`. `pdfauthor` indicates the primary author's position or title. `pdfcaptionwriter` specifies the name of the person who added the metadata to the document.

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

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

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

XMP can include a universally unique identifier (UUID) for each document and for each instance of a given document. By default, `hyperxmp` assigns a version 4 (i.e., pseudorandom) UUID [12] for each of these. However, a document can alternatively specify a particular document identifier using `pdfdocumentid` and (not normally recommended) a particular instance identifier using `pdfinstanceid`. These should be of the form `uuid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`, where "x" is a lowercase hexadecimal number. For example, `uuid:53ab7f19-a48c-5177-8bb2-403ad907f632` is a valid argument to `pdfdocumentid` (or `pdfinstanceid`). See Leach, Mealling, and Salz's UUID specification document for details on how to produce the various forms of UUIDs [12]. A more freeform mechanism than `pdfinstanceid` for versioning documents is available via `pdfversionid`. The version specified by `pdfversionid` can be incremented as 1, 2, 3, ...; identified with a hierarchical numbering scheme (e.g., this document is versioned 5.13 to match the package version); or labeled using any other approach. One possibility is to use a revision number or commit hash from the version-control software maintaining the document. For example, the `\gitVer` macro from the `gitver` package is an expandable (see Note 8 on page 16) version of the current `Git` hash that can suitably be passed to `pdfversionid`. If not specified, `pdfversionid` defaults to 1.

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

<code>pdfissn</code>	specifies the ISBN. <code>pdfissn</code> refers to the ISSN of the <i>print</i> version of the document while <code>pdfeissn</code> refers to the ISSN of the <i>electronic</i> version of the document.
<code>pdfdoi</code>	specifies the DOI and should include only the DOI name without any URL prefix. For example, specify <code>pdfdoi={10.1145/3149526.3149532}</code> , <i>not</i> <code>pdfdoi={https://doi.org/10.1145/3149526.3149532}</code> .
<code>pdfurl</code>	points to the complete URL for the document. In contrast, <code>baseurl</code> points one level up and is used to resolve relative URLs.
<code>baseurl</code>	
<code>pdfidentifier</code>	<code>pdfidentifier</code> provides an alternative mechanism to uniquely identify a document. Its advantage relative to <code>pdfisbn</code> , <code>pdfissn</code> , <code>pdfdoi</code> , etc. is its flexibility; any of a wide variety of identification types can be used. ¹ <code>pdfidentifier</code> 's disadvantage is that it allows only a single identifier per document. For example, a document could use <code>pdfidentifier=urn:iso:std:32000:ed-1:v1:en</code> to identify itself as version 1 of English-language ISO standard 32000-1, but then this same document could not also use <code>pdfidentifier</code> to identify itself by DOI (<code>info:doi/...</code>), ISBN (<code>urn:ISSN:...</code>), etc. (It can still use the options described in the previous paragraph, though.) If <code>pdfidentifier</code> is not specified explicitly, <code>hyperxmp</code> will use the first non-empty value out of the DOI, electronic ISSN, print ISSN, and ISBN or skip the identifier entirely if all of those are empty.
<code>pdfpublication</code>	Already-published documents can further be identified by the publication in which they appear. <code>pdfpublication</code> specifies the title of the journal, magazine, or other parent document. The title language is assumed to be the same as the document language (<code>pdflang</code>) but can be overridden by preceding the text with a bracketed ISO 639-1 two-letter language code and an optional ISO 3166-1 two-letter region code. For example, <code>pdfpublication={[fr]Charlie Hedbo}</code> indicates a French-language title. Were the language or pronunciation differences significant, <code>fr-FR</code> would indicate specifically the French spoken in France, as opposed to that spoken in, say, Canada (<code>fr-CA</code>) or Belgium (<code>fr-BE</code>). The publisher itself can be named using <code>pdfpublisher</code> .
<code>pdfpublisher</code>	
<code>pdfpubtype</code>	<code>pdfpubtype</code> indicates the type of publication in which the document was published. This should be one of the PRISM aggregation types [9] such as <code>book</code> , <code>journal</code> , <code>magazine</code> , <code>manual</code> , <code>report</code> , or <code>whitepaper</code> .
<code>pdfvolumenum</code>	For publications in journals, magazines, and similar periodicals, a document can specify the volume number with <code>pdfvolumenum</code> and the issue number within the volume with <code>pdfissuenum</code> .
<code>pdfissuenum</code>	<code>pdfpagerange</code> indicates the page numbers at which the document appears within the publication. The intention is that this be a comma-separated list of dash-separated ranges, as in <code>pdfpagerange={1,4-5}</code> . See Note 9 on page 16 for advice on how to assign <code>pdfpagerange</code> semi-automatically. A journal article's publication status can be indicated with <code>pdfpubstatus</code> . This option expects to take one of the values listed in Table 1. See the NISO/ALPSP Journal Article Versions recommendation [1] for an explanation of each of those values and when to use them.
<code>pdfpagerange</code>	
<code>pdfpubstatus</code>	
<code>pdfbookedition</code>	For books, <code>pdfbookedition</code> names the edition of the book. This is specified as text, not a number. As with <code>pdfpublication</code> (above), <code>pdfbookedition</code> accepts a bracketed language code, as in <code>pdfbookedition={[en]Second edition}</code> .
<code>pdfdate</code>	XMP metadata can include a number of dates (in fact, timestamps, as they include both date and time components). <code>pdfdate</code> specifies the document date. It is analogous to the L ^A T _E X <code>\date</code> command, and, like <code>\date</code> , defaults to the date

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

Table 1: Valid arguments for `pdfpubstatus`

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

the document was built. It must be specified in either XMP format [5] or PDF format [4]. XMP dates are written in the form `YYYY-MM-DDThh:mm:ss+TT:tt`.² A W3C recommendation [15] discusses this format in more detail, but as an example, 14 hours, 15 minutes, 9 seconds past midnight U.S. Mountain Daylight Time (UTC-6) on the 23rd day of September in the year 2014 should be written as `2014-09-23T14:15:09-06:00`. This can be truncated (with loss of information) to `2014-09-23T14:15:09`, `2014-09-23T14:15`, `2014-09-23`, `2014-09`, or `2014` but no other subsets. PDF dates are written in the form `D:YYYYMMDDhhmmss+TT’tt’`. The same date in the preceding example would be written as `D:20140923141509-06’00’` in PDF format.

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

`pdftype` describes the type of document being produced. This refers to “the nature or genre of the resource” [5] such as `poem`, `novel` or `working paper`, as opposed to the file format (always `application/pdf` when generated by `hyperxmp`). Although `pdftype` can be assigned an arbitrary piece of text, the XMP specification recommends selecting types from a “controlled vocabulary” such as the DCMI Type Vocabulary [6]. The DCMI Type Vocabulary currently consists of only `Collection`, `Dataset`, `Event`, `Image`, `InteractiveResource`, `MovingImage`, `PhysicalObject`, `Service`, `Software`, `Sound`, `StillImage`, and `Text`. `pdftype` defaults to `Text`, which refers to “books, letters, dissertations, poems, newspapers, articles, archives of mailing lists,” [6] and other forms of text—all things L^AT_EX is commonly used to typeset.

Sometimes a base document is rendered in different forms. `pdfrendition` indicates the particular rendition the current document instance represents. The value should come from the following controlled vocabulary [5]: `default`, `draft`, `low-res`, `proof`, `screen`, and `thumbnail`. `hyperxmp`’s default value is `default`, which indicates the master document, unless the `draft` option is passed to `\documentclass`, in which case `hyperxmp` defaults to `draft`.

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

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

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

`pdfapart` and `pdfaconformance`, are used in conjunction with `hyperref`'s `pdfa` option to claim a particular PDF/A standard by which the document abides. They default to `pdfapart=1` and `pdfaconformance=B`, indicating the PDF/A-1b standard. These can be changed (with caution) to assert that the document abides by a different standard (e.g., PDF/A-2u). A document that conforms to the PDF/UA standard can use `pdfuapart` to indicate the PDF/UA conformance level. For example, `pdfuapart=1` asserts that the document respects PDF/UA-1. `pdfxstandard` indicates the particular PDF/X standard by which the document abides. Unlike `pdfapart` and `pdfaconformance`, which accept a number and a letter, respectively, `pdfxstandard` expects a textual identification of a standard name. The following are the acceptable PDF/X standard names as of at the time of this writing.

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

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

Rarely needed options

`pdfsource` `pdfsource` overrides the name of the \LaTeX source file. It defaults to `\jobname.tex` but can be replaced by any other string. If `pdfsource` is given an empty argument, no document source will be specified at all.

The number of pages in the published, print version of the document can be expressed with `pdfnumpages`. This is computed automatically when the document is built using either `pdf \LaTeX` or `Lua \LaTeX` .

`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 `pdf \LaTeX` and `Lua \LaTeX` , 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 \LaTeX document that provides values for most of the metadata options that `hyperxmp` recognizes:

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

```

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

\hypersetup{%
  pdftitle={%
    On a heuristic viewpoint concerning the production and
    transformation of light},
  pdfsubtitle={[en-US]Putting that bum Maxwell in his place},
  pdfauthor={Albert Einstein},
  pdfauthortitle={\xmpquote{Technical Assistant\xmpcomma\ Level III}},
  pdfdate={1905-03-17},
  pdfcopyright={Copyright (C) 1905, Albert Einstein},
  pdfsubject={photoelectric effect},
  pdfkeywords={energy quanta, Hertz effect, quantum physics},
  pdflicenseurl={http://creativecommons.org/licenses/by-nc-nd/3.0/},
  pdfcaptionwriter={Scott Pakin},
  pdfcontactaddress={Kramgasse 49},
  pdfcontactcity={Bern},
  pdfcontactpostcode={3011},
  pdfcontactcountry={Switzerland},
  pdfcontactphone={031 312 00 91},
  pdfcontactemail={aeinstein@ipi.ch},
  pdfcontacturl={%
    http://einstein.biz/,
    https://www.facebook.com/AlbertEinstein
  },
  pdfdocumentid={uuid:6d1ac9ec-4ff2-515a-954b-648eeb4853b0},
  pdfversionid={2.998e8},
  pdfpublication={[de]Annalen der Physik},
  pdfpublisher={Wiley-VCH},
  pdfpubtype={journal},
  pdfvolumenum={322},
  pdfissuenum={6},
  pdfpagerange={132-148},
  pdfissn={0003-3804},
  pdfeissn={1521-3889},
  pdfpubstatus={VoR},
  pdflang={en},
  pdfmetalang={en},
  pdfurl={http://www.physik.uni-augsburg.de/annalen/history/einstein-
papers/1905_17_132-148.pdf},
  pdfdoi={10.1002/andp.19053220607},
  pdfidentfier={info:lccn/50013519}
}
\XMLLangAlt{de}{pdftitle={Über einen die Erzeugung und Verwandlung des
  Lichtes betreffenden heuristischen Gesichtspunkt}}

\begin{document}
\maketitle
A profound formal difference exists between the theoretical
concepts that physicists have formed about gases and other

```

```
ponderable bodies, and Maxwell's theory of electromagnetic
processes in so-called empty space\dots
\end{document}
```

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

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

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

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

2.3 Usage notes

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

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

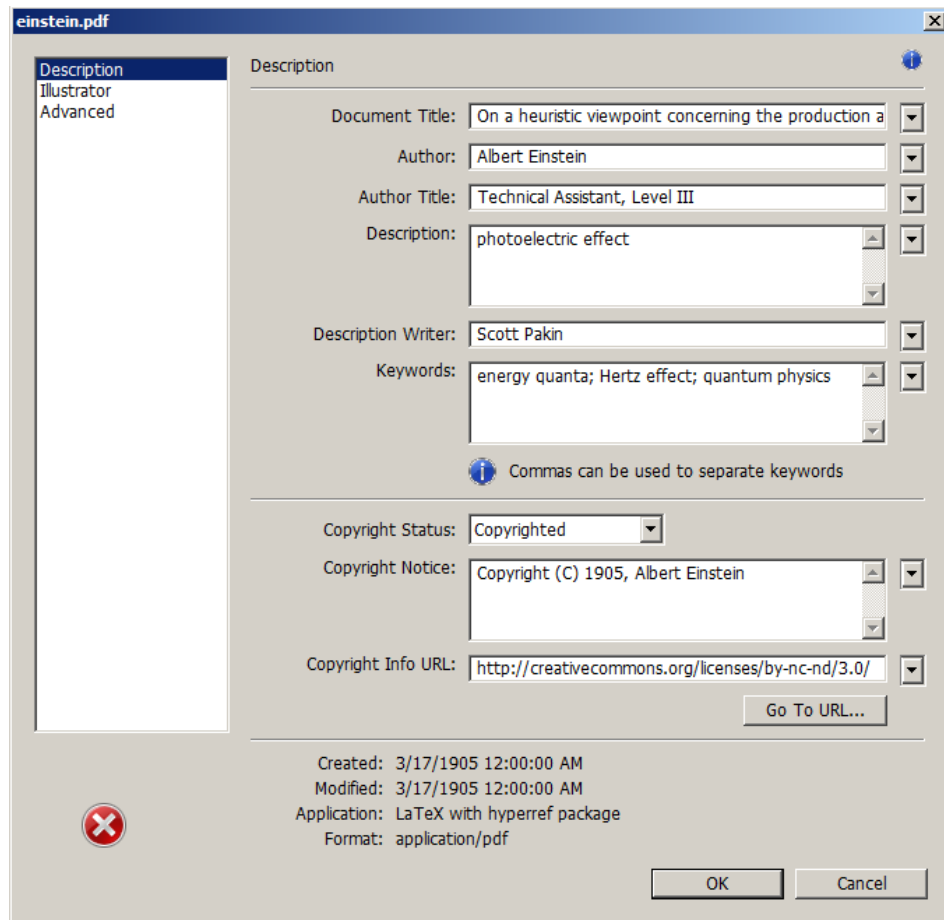


Figure 2: XMP metadata as it appears in Adobe Acrobat

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

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

Note 2: Acrobat multiline-field bug The IPTC Photo Metadata schema states that “the [contact] address is a multiline field” [10]. `hyperxmp` converts commas in `pdfcontactaddress`’s argument to line breaks in the generated XML. Unfortunately, A

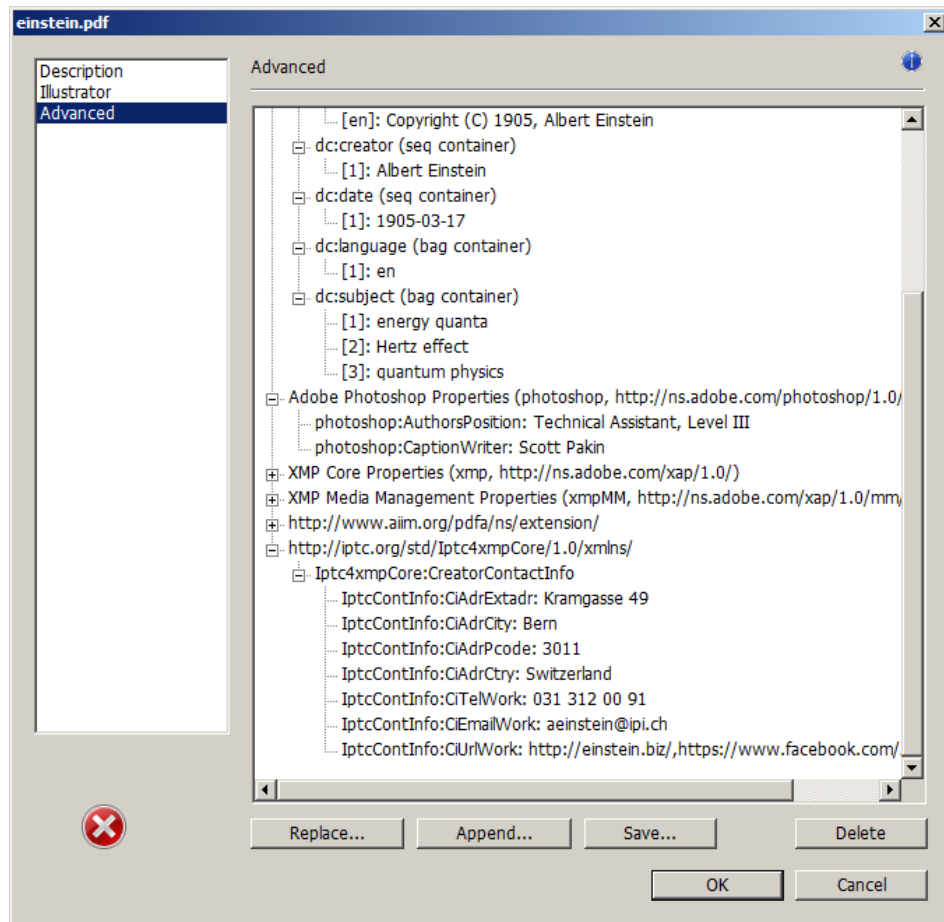


Figure 3: Additional XMP metadata as it appears in Adobe Acrobat

bug in Adobe Acrobat—at least in Adobe Acrobat DC (2019) and earlier versions—causes that PDF reader to discard line breaks in the contact address. Interestingly, Adobe Illustrator CS5 correctly displays the contact address. If you find Adobe Acrobat’s behavior bothersome, you can redefine the `\xmplinesep` macro as a string to use as an address-line separator. For example, the following replaces all commas appearing in `pdfcontactaddress`’s argument with semicolons:

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

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

1. When run with versions of `LuaATEX` earlier than 0.85, `hyperxmp` leaves all PDF objects uncompressed. This is due to `LuaATEX` treating object compression

as a global parameter, unlike pdfL^AT_EX, which treats it as a local parameter. Hence, when hyperxmp requests that the XMP packet be left uncompressed, LuaL^AT_EX in fact leaves *all* PDF streams uncompressed. Beginning with version 3.0, hyperxmp includes a workaround that correctly leaves only the XMP metadata uncompressed, but this workaround is implemented only for LuaL^AT_EX v0.85 onwards.

2. X_YL^AT_EX (or, more precisely, the xdvipdfmx back end) exhibits the opposite problem. It compresses *all* PDF objects, including the ones containing XMP metadata. While Adobe Acrobat can still detect and utilize the XMP metadata, non-PDF-aware applications are unlikely to see the metadata. Three options to consider are to (1) use a different program (e.g., LuaL^AT_EX), (2) pass the `--output-driver="xdvipdfmx -z0"` option to X_YL^AT_EX to instruct xdvipdfmx to turn off all compression (which will of course make the PDF file substantially larger), or (3) postprocess the generated PDF file by loading it into the commercial version of Adobe Acrobat and re-saving it with the Save As... menu option.

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

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

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

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

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

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

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

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

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

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

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

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

Note 6: Automatically specified metadata `hyperxmp` attempts to identify certain metadata automatically. The hope is that in many cases, an author can simply include `\usepackage{hyperxmp}` in a document’s preamble and benefit from a modicum of XMP metadata with no additional effort.

Currently, `pdftitle` defaults to the document’s title as specified by `\title{...}`. `pdfauthor` defaults to the document’s author(s) as specified by `\author{...}`. `pdfdate` defaults to the current date and time. `pdfmetalang` defaults to the same value as `pdflang` if non-empty, “x-default” otherwise. `hyperxmp` recognizes some class-specific metadata as well, such as that provided via the Koma letter classes (e.g., `scrlettr2`) and the ACM article class (`acmart`).

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

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

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

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

```
\hypersetup{%
  pdfmetalang={en},
  pdftitle={English title}
}
\XMPLangAlt{de}{pdftitle={Deutscher Titel}}
```

```

\XMPLangAlt{fr}{pdftitle={Titre fran\c{c}ais}}
\XMPLangAlt{it}{pdftitle={Titolo italiano}}
\XMPLangAlt{rm}{pdftitle={Titel rumantsch}}

```

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

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

```

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

```

Rather, it generates a `dc:rights` tag of the form “Copyright © =2=0=by-1by=02024, Scott Pakin”. The garbage in that line corresponds to the remnants of the `\printfdate` code after expanding all of the \TeX primitives and certain other control sequences it uses to the empty string. For example, “`\global\advance\texd@yr by-1`” expands to “`by-1`”.

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

Note 9: Semi-automatic page ranges Although `pdfpagerange` is intended to refer to pages in the final, published version of a document, it would be convenient for them to be generated automatically when producing a standalone PDF file that is not intended to be incorporated into a book, journal, or other publication (or if it is known that the pages will not be renumbered for publication). One approach is to use the `totpages` package help generate `pdfpagerange`. For documents numbered from 1 to n , a simple

```

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

```

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


```

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

```

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

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

```

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

```

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

Starting with `hyperxmp` v5.9, the `hyperxmp` distribution includes a Perl script called `hyperxmp-add-bytecount` that edits a PDF file in place, adding or replacing the `prism:byteCount` property with one that specifies the final file size.³ Run the script as “`hyperxmp-add-bytecount <filename.pdf>`”.

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

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

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

```

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

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

```

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

3 Implementation

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

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

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

3.1 Initial preparation

```

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

```

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

```

7 \edef\hyxmp@dq@code{\the\catcode'\}
8 \catcode'\=12

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

\hyxmp@jobname 23 \def\hyxmp@set@jobname@plain#1\relax{\xdef\hyxmp@jobname{#1}}

```

Define `\hyxmp@jobname` as a sanitized version of `\jobname`. The problem with using `\jobname` directly is that it surrounds the filename with double quotes if it contains a space character. For example, a source file named `my-file.tex` results in a `\jobname` of “`my-file`”, but a source file named `my file.tex` results in a `\jobname` of “`"my file"`”. Trying to access “`my file`”.log (as is done on page 47) will fail because the filename does not in fact contain literal double quotes.

```

24 \expandafter\hyxmp@set@jobname\jobname\relax

```

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

```

25 \newtoks{\hyxmp@aep@toks}

```

3.2 Integration with `hyperref`

An important design decision underlying `hyperxmp` is that the package should integrate seamlessly with `hyperref`. To that end, `hyperxmp` takes XMP metadata

from `hyperref`'s `baseurl`, `pdfauthor`, `pdfkeywords`, `pdflang`, `pdfproducer`, `pdfsubject`, `pdftrapped`, and `pdftitle` options. It also introduces a number of new options, which are listed on page 5. For consistency with `hyperref`'s document-metadata naming conventions (which are in turn based on L^AT_EX's document-metadata naming conventions), we do not prefix metadata-related macro names with our package-specific `\hyxmp@` prefix. That is, we use names like `\pdfcopyright` instead of `\hyxmp@pdfcopyright`.

We load a bunch of helper packages: `kvoptions` for package-option processing, `pdfescape` and `stringenc` for re-encoding Unicode strings, `intcalc` for performing integer calculations (division and modulo), `iftex` for determining which T_EX engine is being used, `ifmtarg` for testing if a macro argument is empty or all spaces, `etoolbox` for dynamically patching existing commands (specifically, `hyperref`'s `\PDF@FinishDoc`), and `ifthen` for convenient string comparisons.

```

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

```

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

```

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

```

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

```

37 \def\ifmtargexp#1{\expandafter\ifmtarg\expandafter{#1}}
38 \def\ifnotmtargexp#1{\expandafter\ifnotmtarg\expandafter{#1}}

```

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

```

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

```

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

```

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

```

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

80 \def\@pdfcaptionwriter{}
81 \define@key{Hyp}{pdfcaptionwriter}{\hyxmp@pdfstringdef\@pdfcaptionwriter{#1}}

```

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

```

82 \def\@pdfmetalang{}
83 \define@key{Hyp}{pdfmetalang}{\hyxmp@pdfstringdef\@pdfmetalang{#1}}

```

`\hyxmp@no@bad@parts` Complain about a bad pdfapart or pdfuapart if given trailing non-digits after a part number.

```

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

```

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

```

89 \newcount\@hyxmp@count

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

```

\hyxmp@set@pdfx@major@ii This is the second helper macro for \hyxmp@set@pdfx@major. It copies the PDF/X
\hyxmp@pdfx@major major version from \@hyxmp@count to \@hyxmp@pdfx@major and discards the rest
of the PDF/X standard string.
109 \def\hyxmp@set@pdfx@major@ii#1!{%
110 \edef\hyxmp@pdfx@major{\the\hyxmp@count}%
111 }

\hyxmp@check@std Compare a user-provided string to a fixed string. (Assumption: Both are names of
PDF/X standard versions.) If they match, undefine \next, which we assume was
previously defined to issue an “unrecognized standard” warning message.
112 \newcommand*\hyxmp@check@std[2]{%
113 \ifthenelse{\equal{#1}{#2}}%
114 {\global\let\next=\relax}%
115 {}%
116 }%

\@pdfxstandard Prepare to store the PDF/X standard.
117 \def\@pdfxstandard{}
118 \def\hyxmp@pdfx@major{}
119 \define@key{Hyp}{pdfxstandard}{%
120 \hyxmp@pdfstringdef\@pdfxstandard{#1}%

\next Issue a warning message if the PDF/X standard named by the user does not appear
in a list of known PDF/X standards. This is to caution the user that hyperxmp
generates standard-specific XMP metadata and it can only guess at the correct
format for new standard versions. (See the comments on page 69 above the
definition of \hyxmp@pdfx@id@schema, for example.)
121 \gdef\next{%
122 \PackageWarning{hyperxmp}{Unrecognized PDF/X standard ‘#1’}%
123 }%
124 \hyxmp@check@std{#1}{PDF/X-1a:2001}%
125 \hyxmp@check@std{#1}{PDF/X-1a:2003}%
126 \hyxmp@check@std{#1}{PDF/X-3:2002}%
127 \hyxmp@check@std{#1}{PDF/X-3:2003}%
128 \hyxmp@check@std{#1}{PDF/X-4}%
129 \hyxmp@check@std{#1}{PDF/X-4p}%
130 \hyxmp@check@std{#1}{PDF/X-5g}%
131 \hyxmp@check@std{#1}{PDF/X-5n}%
132 \hyxmp@check@std{#1}{PDF/X-5pg}%
133 \next

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

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

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

```

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

```

140 \def\hyxmp@InstanceID{}
141 \define@key{Hyp}{pdfinstanceid}{\hyxmp@pdfstringdef\hyxmp@InstanceID{#1}}

```

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

```

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

```

`\ifdraft` Use the `ifdraft` package to determine if this is a draft or final document. The `\next` challenge here is that we want to use `ifdraft` if it’s already loaded, load it if not, and not break any incompatible, author-defined `\ifdraft` macros that may occur either before or after the `\usepackage{hyperxmp}`. Our solution works as follows:

- If `ifdraft` is already loaded, `\next` is defined as a no-op.
- If `ifdraft` is not (yet) loaded, the code backs up then undefines `\ifdraft`, which may be author-defined. It then loads `ifdraft` and defines `\next` to “unload” the package by setting the package’s author-visible commands to `\relax`.
- Below, after `\ifdraft` is used to define `\@pdfrendition`, `\next` is invoked to unload a `hyperxmp`-loaded `ifdraft`. Finally, `\ifdraft` is restored to its original definition.

```

144 \let\hyxmp@orig@ifdraft=\ifdraft
145 \@ifpackageloaded{ifdraft}{%
146   \let\next=\relax
147 }{%
148   \let\ifdraft=\relax
149   \RequirePackage{ifdraft}%
150   \def\next{%
151     \expandafter\let\csname ver@ifdraft.sty\endcsname=\relax
152     \let\ifdraft=\relax
153     \let\ifoptiondraft=\relax
154     \let\ifoptionfinal=\relax
155   }%
156 }%

```

`\@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`.

```

157 \ifdraft{%
158   \def\@pdfrendition{draft}%
159 }{%
160   \def\@pdfrendition{default}%
161 }
162 \next
163 \let\ifdraft=\hyxmp@orig@ifdraft
164 \define@key{Hyp}{pdfrendition}{\hyxmp@pdfstringdef\@pdfrendition{#1}}

```

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

```

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

```


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

```

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

```

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

```

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

```

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

```

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

```

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

```

173 \def\@pdfissn{}
174 \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.

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

185 \def\@pdfissuenum{}
186 \define@key{Hyp}{pdfissuenum}{\hyxmp@pdfstringdef\@pdfissuenum{#1}}

```

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

```

187 \def\@pdfpagerange{}
188 \define@key{Hyp}{pdfpagerange}{\hyxmp@pdfstringdef\@pdfpagerange{#1}}

```

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

```

189 \def\@pdfdoi{}
190 \define@key{Hyp}{pdfdoi}{\hyxmp@pdfstringdef\@pdfdoi{#1}}

```

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

```
191 \def\@pdfurl{}
192 \define@key{Hyp}{pdfurl}{\hyxmp@pdfstringdef\@pdfurl{#1}}
```

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

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

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

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

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

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

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

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

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

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

```
199 \def\@pdfcontactaddress{}
200 \define@key{Hyp}{pdfcontactaddress}{%
201   \let\xmpcomma=\hyxmp@comma
202   \def\xmpquote##1{##1}%
203   \hyxmp@pdfstringdef\@pdfcontactaddress{#1}%
204   \def\xmpcomma{,}%
205   \let\xmpquote=\relax
206 }
```

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

```
207 \def\@pdfcontactcity{}
208 \define@key{Hyp}{pdfcontactcity}{\hyxmp@pdfstringdef\@pdfcontactcity{#1}}
```

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

```
209 \def\@pdfcontactregion{}
210 \define@key{Hyp}{pdfcontactregion}{\hyxmp@pdfstringdef\@pdfcontactregion{#1}}
```

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

```
211 \def\@pdfcontactpostcode{}
212 \define@key{Hyp}{pdfcontactpostcode}{\hyxmp@pdfstringdef\@pdfcontactpostcode{#1}}
```

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

```

213 \def\@pdfcontactcountry{}
214 \define@key{Hyp}{pdfcontactcountry}{\hyxmp@pdfstringdef\@pdfcontactcountry{#1}}

```

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

```

215 \def\@pdfcontactphone{}
216 \define@key{Hyp}{pdfcontactphone}{\hyxmp@pdfstringdef\@pdfcontactphone{#1}}

```

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

```

217 \def\@pdfcontactemail{}
218 \define@key{Hyp}{pdfcontactemail}{\hyxmp@pdfstringdef\@pdfcontactemail{#1}}

```

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

```

219 \def\@pdfcontacturl{}
220 \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`.

```

221 \def\hyxmp@no@info@lists{}

```

`\hyxmp@suppress@pdf@info` If `\patchcmd` fails for any reason—most likely, a modification to the `hyperref` package—our fallback is to prevent `hyperref` from writing *any* data to the PDF Info dictionary.

```

222 \def\hyxmp@suppress@pdf@info{%
223   \global\let\PDF@FinishDoc=\@empty
224   \PackageWarningNoLine{hyperxmp}{%
225     Suppressing the _entire_ PDF Info dictionary.\MessageBreak
226     Please notify the hyperxmp maintainer%
227   }%
228 }%
229 \let\next=\relax
230 \patchcmd
231   {\PDF@FinishDoc}%
232   {/Author(\@pdfauthor)}%
233   {}%
234   {}%
235   {\let\next=\hyxmp@suppress@pdf@info}%
236 \patchcmd
237   {\PDF@FinishDoc}%
238   {/Keywords(\@pdfkeywords)}%
239   {}%
240   {}%
241   {\let\next=\hyxmp@suppress@pdf@info}%
242 \next
243 }

244 \define@key{Hyp}{keeppdfinfo}[true]{%
245   \gdef\hyxmp@no@info@lists{}%
246 }

```

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`.

New in v5.12 `hyperref` must be loaded *before* `hyperxmp`. This is because recent changes in `hyperref` and the L^AT_EX kernel prevent `hyperxmp` from hooking into `hyperref`'s internals if `hyperref` is loaded first.

```
247 \@ifpackageloaded{hyperref}{%
248 }{%
249   \PackageError{hyperxmp}%
250     {hyperref must be loaded before hyperxmp}%
251     {Recent versions of hyperref and the LaTeX kernel inhibit\MessageBreak
252       hyperxmp's ability to hook into hyperref's internals unless\MessageBreak
253       hyperref is loaded first.}
254 }
```

```
\hyxmp@pdfauthor Prepare to store the name of the author and a list of keywords.
\hyxmp@pdfkeywords 255 \def\hyxmp@pdfauthor{}
256 \def\hyxmp@pdfkeywords{}
```

```
\hyxmp@redefine@Hyp If not already redefined, redefine hyperref's pdfauthor and pdfkeywords options to
properly handle \xmpcomma and \xmpquote.
257 \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.
258 \@ifundefined{KV@Hyp@pdfauthor}{}{%
259   \@ifundefined{hyxmp@Hyp@pdfauthor}{%
260     \expandafter\let\expandafter\hyxmp@Hyp@pdfauthor
261     \csname KV@Hyp@pdfauthor\endcsname
262   }{}%
263 }
```

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

```

264 \define@key{Hyp}{pdfauthor}{%
265   \let\xmpcomma=\hyxmp@comma
266   \def\xmpquote####1{####1}%
267   \let\hyxmp@and=\and
268   \def\and{,}%
269   \hyxmp@Hyp@pdfauthor{##1}%
270   \global\let\hyxmp@pdfauthor=\@pdfauthor
271   \def\and{and\space}%
272   \def\xmpcomma{,%}
273   \def\xmpquote####1{"####1"%
274   \hyxmp@Hyp@pdfauthor{##1}%
275   \def\xmpcomma{,%}
276   \let\xmpquote=\relax
277   \let\and=\hyxmp@and
278 }%
```

`\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`.

```

279 \@ifundefined{KV@Hyp@pdfkeywords}{-}{%
280   \@ifundefined{hyxmp@Hyp@pdfkeywords}{-%
281     \expandafter\let\expandafter\hyxmp@Hyp@pdfkeywords
282     \csname KV@Hyp@pdfkeywords\endcsname
283   }-}{%
284 }%
```

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

```

285 \define@key{Hyp}{pdfkeywords}{-%
286   \let\xmpcomma=\hyxmp@comma
287   \def\xmpquote####1{####1}%
288   \hyxmp@Hyp@pdfkeywords{##1}%
289   \global\let\hyxmp@pdfkeywords=\@pdfkeywords
290   \def\xmpcomma{,%}
291   \def\xmpquote####1{"####1"%
292   \hyxmp@Hyp@pdfkeywords{##1}%
293   \def\xmpcomma{,%}
```

```

294   \let\xmpquote=\relax
295   }%
296 }

\hyxmp@ProcessKeyvalOptions Redefine kvoptions's \ProcessOptions command to invoke \hyxmp@redefine@Hyp
\ProcessKeyvalOptions before performing its normal option processing.
297 \let\hyxmp@ProcessKeyvalOptions=\ProcessKeyvalOptions
298 \renewcommand*\ProcessKeyvalOptions}{%
299   \global\let\ProcessKeyvalOptions=\hyxmp@ProcessKeyvalOptions
300   \hyxmp@redefine@Hyp
301   \hyxmp@ProcessKeyvalOptions
302 }

\hyxmp@hypersetup Redefine hyperref's \hypersetup command to invoke \hyxmp@redefine@Hyp before
\hypersetup performing its normal option processing.
303 \let\hyxmp@hypersetup=\hypersetup
304 \def\hypersetup{%
305   \hyxmp@redefine@Hyp
306   \hyxmp@hypersetup
307 }

\hyxmp@concat@metadata Assume that if the document loaded either babel or polyglossia it will eventually
\hyxmp@aep@toks define one or more languages that hyperxmp can list within a dc:language element.
As explained in Section 3.1, we defer the invocation of \AtEndPreamble to the end
of the file.
308 \edef\hyxmp@concat@metadata{}
309 \expandafter\hyxmp@aep@toks\expandafter=\expandafter{%
310   \the\hyxmp@aep@toks
311   \AtEndPreamble{%
312     \@ifpackageloaded{babel}{%
313       \edef\hyxmp@concat@metadata{babel}%
314     }{%
315       \@ifpackageloaded{polyglossia}{%
316         \edef\hyxmp@concat@metadata{polyglossia}%
317       }{%
318         }%
319     }%
320 }%
321 }

\hyxmp@warn@if@no@metadata Issue a warning message if the author failed to specify any metadata at all. This
\hyxmp@concat@metadata excludes metadata that is included automatically such as the current timestamp.
Note that we don't consider \@pdfmetlang 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.
322 \newcommand*\hyxmp@warn@if@no@metadata}{%
323   \edef\hyxmp@concat@metadata{%
324     \hyxmp@concat@metadata
325     \@baseurl
326     \@pdfauthor
327     \@pdfauthortitle
328     \@pdfbookedition
329     \@pdfbytes

```

```

330 \pdfcaptionwriter
331 \pdfcontactaddress
332 \pdfcontactcity
333 \pdfcontactcountry
334 \pdfcontactemail
335 \pdfcontactphone
336 \pdfcontactpostcode
337 \pdfcontactregion
338 \pdfcontacturl
339 \pdfcopyright
340 \pdfcreationdate
341 \pdfdatetime
342 \pdfdoi
343 \pdfeissn
344 \pdfidentifier
345 \pdfisbn
346 \pdfissn
347 \pdfissuenum
348 \pdfkeywords
349 \pdflang
350 \pdflicenseurl
351 \pdfmetadatetitle
352 \pdfmoddate
353 \pdfnumpages
354 \pdfpagerange
355 \pdfpublication
356 \pdfpubtype
357 \pdfsubject
358 \pdfsubtitle
359 \pdftitle
360 \pdfuapart
361 \pdfurl
362 \pdfvolumenum
363 \pdfxstandard
364 }%
365 \ifx\hyxmp@concat@metadata\@empty
366 \PackageWarningNoLine{hyperxmp}{%
367 \hyxmp@jobname.tex did not specify any metadata to\MessageBreak
368 include in the XMP packet.\space\space Please see the\MessageBreak
369 hyperxmp documentation for instructions on how to\MessageBreak
370 provide metadata values to hyperxmp}%
371 \fi
372 }

```

`\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.

```
373 \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.

```

374 \ifHy@pdfa
375 \ifmtargexp{\pdfapart}{%
376 \PackageWarningNoLine{hyperxmp}{%

```

```

377         'pdfa' was passed to hyperref, but 'pdfapart' was\MessageBreak
378         not specified.\space\space Setting pdfapart to '1' and\MessageBreak
379         pdfaconformance to 'B'%
380     }%
381     \gdef\@pdfapart{1}%
382     \gdef\@pdfaconformance{B}%
383 }%
384 {}%
385 \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).

```

386 \edef\hyxmp@standards{%
387     \@pdfapart
388     \@pdfxstandard
389     \@pdfuapart
390 }%

```

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

```

391 \@ifnotmtargexp{\hyxmp@standards}{%
392     \@ifmtargexp{\@pdftitle}{%
393         \PackageWarningNoLine{hyperxmp}{%
394             Missing pdftitle (required for PDF standards\MessageBreak
395             compliance)%
396         }%
397     }%
398 }%
399 }%
400 }

```

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

```

401 \expandafter\hyxmp@aep@toks\expandafter=\expandafter{%
402     \the\hyxmp@aep@toks
403     \AtEndPreamble{%
404         \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.

```

405     \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.

```

406   \begingroup
407     \let\@latex@warning@no@line=\gobble
408     \hyxmp@use@first@valid{pdftitle}{\@pdftitle}{%
409       \scr@subject@var,%
410       \@title
411     }%
412     \hyxmp@use@first@valid{pdfauthor}{\@pdfauthor}{%
413       \scr@fromname@var,%
414       \@author
415     }%
416   \endgroup
417 }%
418 }

```

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.

```
419 \hyxmp@at@end{%
```

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

```
420   \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.

```
421   \hyxmp@check@standards
```

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

```

422   \hyxmp@warn@if@no@metadata
423   \hyxmp@embed@packet
424 }

```

3.3 Advanced metadata detection

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

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

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

```

425 \newcommand*{\hyxmp@set@koma@phones}{%
426   \begingroup
427     \Hy@unicodedefalse
428     \@if@def@and@nonempty{scr@frommobilephone@var}{%
429       \@if@def@and@nonempty{scr@fromphone@var}{%
430         \hyxmp@pdfstringdef\hyxmp@koma@phones{\scr@frommobilephone@var,\scr@fromphone@var}%
431       }{%
432         \hyxmp@pdfstringdef\hyxmp@koma@phones{\scr@frommobilephone@var}%
433       }%
434     }{%
435       \@if@def@and@nonempty{scr@fromphone@var}{%
436         \hyxmp@pdfstringdef\hyxmp@koma@phones{\scr@fromphone@var}%
437       }{%
438         }%
439     }%
440 \endgroup
441 }

```

`\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.

```

442 \newcommand*{\hyxmp@use@first@valid}[3]{%
443   \ifmtargexp{#2}{%
444     \hyxmp@use@first@valid@i{#1}#3,!,%
445   }%
446 }%
447 }

```

`\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`.

```

448 \def\hyxmp@use@first@valid@i#1#2,{%
449   \def\next{\hyxmp@use@first@valid@i{#1}}%
450   \ifx#2!%
451     \let\next=\relax
452   \else
453     \ifx#2\undefined
454     \else
455       \@ifnotmtargexp{#2}{%
456         \hypersetup{#1={#2}}%
457       \def\next##1!,{}}%
458     }%
459   \fi
460 \fi
461 \next
462 }

```

`\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).

```

463 \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.

```

464 \if@def@and@nonempty{@pdflang}{%
465   \let\hyxmp@dc@lang=\@pdflang
466 }{%
467   \hyxmp@detect@langs
468 }%

```

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

```

469 \ifx\@pdfmetalang\@empty
470   \ifx\@pdflang\@empty
471     \let\@pdfmetalang=\hyxmp@x@default
472   \else
473     \edef\@pdfmetalang{\@pdflang}%
474   \fi
475 \fi

```

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

```

476 \hyxmp@use@first@valid{pdfcontactemail}{\@pdfcontactemail}{%
477   \scr@fromemail@var
478 }%
479 \hyxmp@set@koma@phones
480 \hyxmp@use@first@valid{pdfcontactphone}{\@pdfcontactphone}{%
481   \hyxmp@koma@phones
482 }%
483 \hyxmp@use@first@valid{pdfcontacturl}{\@pdfcontacturl}{%
484   \scr@fromurl@var
485 }%
486 \hyxmp@use@first@valid{pdfsubtitle}{\@pdfsubtitle}{%
487   \@subtitle
488 }%
489 \hyxmp@use@first@valid{pdfpublisher}{\@pdfpublisher}{%
490   \@publishers
491 }%

```

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.

```

492 \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 `XqLaTeX`. 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.

```

493 \@ifundefined{XeTeXversion}{}{%
494   \@ifmtargexp{\@pdfcreationdate}{%
495     \let\@pdfcreationdate=\hyxmp@today@pdf

```

```

496     }%
497     {}%
498 }%

```

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

```

499 \hyxmp@query@self
500 }

```

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

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

```

501 \ifLuaTeX

```

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

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

```

502 \begin{luacode*}
503 function get_pdf_size (bname)

```

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

```

504     local odir = ""
505     for _, opt in ipairs(arg) do
506         local m = string.match(opt, "%-output%-d.-=(.*)")
507         if m then
508             odir = m
509         end
510     end

```

Set the path separator to either `/`" or `\`", depending on the operating system.

```

511     local sep = "/"
512     if os.type == "windows" or os.type == "msdos" then
513         sep = "\\\"
514     end

```

Concatenate the output directory, path separator, base name, and `.pdf` extension. Do not insert a path separator if either (1) no output directory was specified,

(2) the output directory already ends with the path separator, or (3) the output directory ends in a colon (and is therefore a relative directory) on Windows/DOS. As a few examples,

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

```

515 local dlast = string.sub(odir, -1)
516 if odir == "" or dlast == sep or (dlast == ":" and sep == "\\") then
517     sep = ""
518 end
519 local fname = odir .. sep .. bname .. ".pdf"

```

Query the file size and return it.

```

520 local nbytes = lfs.attributes(fname, "size")
521 return nbytes
522 end
523 \end{luacode*}

```

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

```

524 \xdef\hyxmp@prev@pdf@size{%
525     \luadirect{
526 nbytes = get_pdf_size("\hyxmp@jobname")
527 if nbytes then
528     tex.write(nbytes)
529 end
530 }%
531 }%
532 \fi

```

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

```

533 \newcommand*{\hyxmp@query@self}{%

```

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

```

534 \if@def@and@nonempty{@pdfnumpages}{%
535 }{%
536     \xdef\@pdfnumpages{\thetotalpages}%
537 }%

```

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

```

538 \hyxmp@use@first@valid{pdfbytes}{\@pdfbytes}{%
539     \hyxmp@prev@pdf@size
540 }%
541 }

```

```

\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.
542 \newcommand*{\hyxmp@parse@acmart}{%
543   \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.
544   \let\@author=\@gobble

\email Within \addresses, \email is defined to accept two arguments, the second of
\hyxmp@address@val which is the author's email address.
545   \def\email##1##2{%
546     \def\hyxmp@address@val{##2}%
547     \hyxmp@use@first@valid{pdfcontactemail}{\@pdfcontactemail}{%
548       \hyxmp@address@val
549     }%
550   }%

\streetaddress \streetaddress wraps the author's street address.
\hyxmp@address@val 551   \def\streetaddress##1{%
552     \def\hyxmp@address@val{##1}%
553     \hyxmp@use@first@valid{pdfcontactaddress}{\@pdfcontactaddress}{%
554       \hyxmp@address@val
555     }%
556   }%

\city \city wraps the author's city name.
\hyxmp@address@val 557   \def\city##1{%
558     \def\hyxmp@address@val{##1}%
559     \hyxmp@use@first@valid{pdfcontactcity}{\@pdfcontactcity}{%
560       \hyxmp@address@val
561     }%
562   }%

\state \state wraps the author's state or region name.
\hyxmp@address@val 563   \def\state##1{%
564     \def\hyxmp@address@val{##1}%
565     \hyxmp@use@first@valid{pdfcontactregion}{\@pdfcontactregion}{%
566       \hyxmp@address@val
567     }%
568   }%

\country \country wraps the author's country name.
\hyxmp@address@val 569   \def\country##1{%
570     \def\hyxmp@address@val{##1}%
571     \hyxmp@use@first@valid{pdfcontactcountry}{\@pdfcontactcountry}{%
572       \hyxmp@address@val
573     }%
574   }%

```

```

\postcode \postcode wraps the author's postal code.
\hyxmp@address@val 575 \def\postcode##1{%
                    576 \def\hyxmp@address@val{##1}%
                    577 \hyxmp@use@first@valid{pdfcontactpostcode}{\@pdfcontactpostcode}{%
                    578 \hyxmp@address@val
                    579 }%
                    580 }%

```

`\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.

```

581 \def\affiliation##1##2{%
582 ##2%
583 \let\affiliation=\@gobbletwo
584 }%

```

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.

```

585 \setbox0=\hbox{\addresses}%
586 \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.

```

587 \@if@def@and@nonempty{@acmDOI}{%
588 \IfSubStr{\@acmDOI}{10.1145/1122445.1122456}{-}{%
589 \IfSubStr{\@acmDOI}{10.1145/nnnnnnn.nnnnnnn}{-}{%
590 \hyxmp@use@first@valid{pdfdoi}{\@pdfdoi}{%
591 \@acmDOI
592 }%
593 }%
594 }%
595 }%
596 }%

```

`\hyxmp@strip@isbn@date` Papers appearing in conference proceedings specify the proceedings' ISBN. As 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.

```

597 \@if@def@and@nonempty{@acmISBN}{%
598 \IfSubStr{\@acmISBN}{XXXX}{-}{%
599 \IfSubStr{\@acmISBN}{xxxx}{-}{%
600 \def\hyxmp@strip@isbn@date##1/##2!{##1}%
601 \edef\hyxmp@acm@isbn{%
602 \expandafter\hyxmp@strip@isbn@date\@acmISBN/!%
603 }%

```

```

604     \hyxmp@use@first@valid{pdfisbn}{\@pdfisbn}{%
605     \hyxmp@acm@isbn
606     }%
607   }%
608 }%
609 }%
610 {}%

```

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

```

611 \def\hyxmp@acm@publisher{Association for Computing Machinery}%
612 \hyxmp@use@first@valid{pdfpublisher}{\@pdfpublisher}{%
613   \hyxmp@acm@publisher
614 }%

```

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

```

615 \hyxmp@use@first@valid{pdfpublication}{\@pdfpublication}{%
616   \@journalName,%
617   \@acmBooktitle,%
618   \@acmConference
619 }%

```

`\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).

```

620 \if@ACM@journal
621   \def\hyxmp@acm@pubtype{journal}%
622 \else
623   \def\hyxmp@acm@pubtype{book}%
624 \fi
625 \hyxmp@use@first@valid{pdfpubtype}{\@pdfpubtype}{%
626   \hyxmp@acm@pubtype
627 }%

```

Journal articles have a volume and issue number.

```

628 \hyxmp@use@first@valid{pdfvolumenum}{\@pdfvolumenum}{%
629   \@acmVolume
630 }%
631 \hyxmp@use@first@valid{pdfissuenum}{\@pdfissuenum}{%
632   \@acmNumber
633 }%
634 }

```

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

```

635 \@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.

```

636 \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.

```

637 \newcommand*{\hyxmp@detect@langs}{%
638   \@ifundefined{mainbcp47id}{%
639     \@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.

```
640     \let\pdflang=\@empty
641   }{%
```

`\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.

```
\pdflang 642     \BabelEnsureInfo
643     \LocaleForEach{%
644         \getlocaleproperty\hyxmp@lang@tag{##1}{identification/tag.bcp47}%
645         \ifx\hyxmp@dc@lang\@empty
646             \xdef\hyxmp@dc@lang{\hyxmp@lang@tag}%
647         \else
648             \xdef\hyxmp@dc@lang{\hyxmp@dc@lang,\hyxmp@lang@tag}%
649         \fi
650     \def\hyxmp@lang@name{##1}%
651     \ifx\hyxmp@lang@name\bb1@main@language
652         \edef\pdflang{\hyxmp@lang@tag}%
653     \fi
654   }%
655 }%
656 }{%
```

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

```
657     \xdef\pdflang{\csname mainbcp47id\endcsname}%
658     \edef\hyxmp@dc@lang{\xpg@bcp@loaded}%
659   }%
660 }
```

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 `LATEX` 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 `LATEX` `\@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.)

```
661 \newcommand*\hyxmp@commas@to@list}[2]{%
662   \gdef#1{%
663     \expandafter\hyxmp@commas@to@list@\expandafter#1#2,,%
664 }
```

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

```
\next 665 \def\hyxmp@commas@to@list@i#1#2,{%
666   \gdef\hyxmp@sublist{#2}%
667   \ifx\hyxmp@sublist\@empty
668     \let\next=\relax
669   \else
670     \hyxmp@trimspaces\hyxmp@sublist
671     \@cons{#1}{\hyxmp@sublist}%
672     \def\next{\hyxmp@commas@to@list@i{#1}}%
673   \fi
674   \next
675 }
```

`\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.

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

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

```
677 \bgroup
678   \catcode'^^C=11
679   \gdef\hyxmp@comma{^^C}
680 \egroup
```

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

```
681 \bgroup
682   \catcode'^^U=11
683   \gdef\hyxmp@uscore{^^U}
684 \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.

```
685 \let\xmpquote=\relax
```

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

```
686 \bgroup
687 \catcode'\~ =12%
688 \gdef\xmptilde{~}%
689 \egroup
```

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

```
\hyxmp@temp@str
\hyxmp@temp@list
\@elt
690 \newcommand{\XMPTruncateList}[1]{%
691   \PackageWarning{hyperxmp}{%
692     \noexpand\XMPTruncateList has been deprecated since\MessageBreak
693     hyperxmp 4.0 and may be removed in future\MessageBreak
694     versions of the package. \noexpand\XMPTruncateList\MessageBreak
695     was found}%
696   \edef\hyxmp@temp@str{\csname hyxmp@#1\endcsname}%
697   \hyxmp@commas@to@list{\hyxmp@temp@list}{\hyxmp@temp@str}%
698   \def\@elt##1{%
699     \expandafter\gdef\csname @#1\endcsname{##1}%
700     \let\@elt=\@gobble
701   }
702   \hyxmp@temp@list
703 }
```

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:20240317214217-06’00’) [4], while XMP timestamps are of the form “YYYY-MM-DDThh:mm:ss+TT’tt” (e.g., 2024-03-17T21:42:17-06:00) [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 704 \def\hyxmp@first@char#1{\hyxmp@first@char@i#1\relax}
705 \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.

```
706 \def\hyxmp@as@xmp@date#1{%
707   \expandafter\ifnum\expandafter'\hyxmp@first@char@i#1\relax='D
708   \hyxmp@pdf@to@xmp@date{#1}%
709   \else
710     #1%
711   \fi
712 }
```

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

```

713 \def\hyxmp@pdf@to@xmp@date#1:#2#3#4#5#6#7#8#9{%
714 #2#3#4#5-#6#7-#8#9%
715 \hyxmp@parse@time
716 }

```

`\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`.

```

717 \def\hyxmp@parse@time#1#2#3#4#5#6{%
718 T#1#2:#3#4:#5#6%
719 \hyxmp@parse@tz@char
720 }

```

`\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.

```

721 \def\hyxmp@parse@tz@char#1{%
722 #1%
723 \ifx#1-%
724 \expandafter\hyxmp@parse@tz
725 \else
726 \ifx#1+%
727 \expandafter\hyxmp@parse@tz
728 \fi
729 \fi
730 }

```

`\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.

```

731 \def\hyxmp@parse@tz#1'#2' {%
732 #1:#2%
733 }

```

`\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.

```

734 \def\hyxmp@as@pdf@date#1{%
735 \expandafter\ifx\hyxmp@first@char@i#1\relax D%
736 #1%
737 \else
738 \hyxmp@xmp@to@pdf@date{#1}%
739 \fi
740 }

```

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

```

741 \def\hyxmp@xmp@to@pdf@date#1{%
742   D:\hyxmp@xmp@to@pdf@date@i#1\relax\relax
743 }

```

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

```

744 \def\hyxmp@xmp@to@pdf@date@i#1#2#3#4#5#6{%
745   #1#2#3#4%
746   \ifx#5-%
747     \expandafter\hyxmp@xmp@to@pdf@date@ii\expandafter#6%
748   \fi
749 }

```

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

```

750 \def\hyxmp@xmp@to@pdf@date@ii#1#2#3#4{%
751   #1#2%
752   \ifx#3-%
753     \expandafter\hyxmp@xmp@to@pdf@date@iii\expandafter#4%
754   \fi
755 }

```

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

```

756 \def\hyxmp@xmp@to@pdf@date@iii#1#2#3#4{%
757   #1#2%
758   \ifx#3T%
759     \expandafter\hyxmp@xmp@to@pdf@date@iv\expandafter#4%
760   \fi
761 }

```

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

```

762 \def\hyxmp@xmp@to@pdf@date@iv#1#2#3#4{%
763   #1#2%
764   \ifx#3:%
765     \expandafter\hyxmp@xmp@to@pdf@date@v\expandafter#4%
766   \fi
767 }

```

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

```

768 \def\hyxmp@xmp@to@pdf@date@v#1#2#3#4{%
769   #1#2%
770   \ifx#3:%
771     \expandafter\hyxmp@xmp@to@pdf@date@vi\expandafter#4%
772   \fi
773 }

```

\hyxmp@gobbletwo This is exactly the same as L^AT_EX 2_ε's \@gobbletwo but needs to be a different literal for \hyxmp@xmp@to@pdf@date@vii's pattern-matching to work.

```

774 \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).

```

775 \def\hyxmp@xmp@to@pdf@date@vii#1#2#3#4{%
776   #1#2%
777   \ifx#3+%
778     +\expandafter\hyxmp@xmp@to@pdf@date@vii
779   \fi
780   \ifx#3-%
781     -\expandafter\hyxmp@xmp@to@pdf@date@vii
782   \fi
783   \ifx#3Z%
784     Z%
785   \fi
786   \ifx#3\relax
787     \expandafter\hyxmp@gobbletwo
788   \fi
789   \@gobbletwo #4%
790 }

```

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

```

791 \def\hyxmp@xmp@to@pdf@date@vii#1\@gobbletwo#2#3#4#5{%
792   #2#3%
793   \ifx#4:%
794     \expandafter\hyxmp@xmp@to@pdf@date@viii\expandafter#5%
795   \fi
796 }

```

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

```

797 \def\hyxmp@xmp@to@pdf@date@viii#1#2#3#4{%
798   '#1#2'%
799 }

```

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

```
800 \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.

```

801   \xdef#1{\the\year}%
802   \ifnum\month<10
803     \xdef#1{#1-0\the\month}%
804   \else
805     \xdef#1{#1-\the\month}%
806   \fi
807   \ifnum\day<10
808     \xdef#1{#1-0\the\day}%
809   \else
810     \xdef#1{#1-\the\day}%
811   \fi

```

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

```

812   \@hyxmp@count=\time
813   \divide\@hyxmp@count by 60

```

```

814 \ifnum\@hyxmp@count<10
815   \xdef#1{#1T0\the\@hyxmp@count}%
816 \else
817   \xdef#1{#1T\the\@hyxmp@count}%
818 \fi
819 \multiply\@hyxmp@count by -60
820 \advance\@hyxmp@count by \time
821 \ifnum\@hyxmp@count<10
822   \xdef#1{#1:0\the\@hyxmp@count}%
823 \else
824   \xdef#1{#1:\the\@hyxmp@count}%
825 \fi
826 \xdef#1{#1Z}%
827 }

```

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

```

828 \def\hyxmp@try@today#1#2{%
829   \@ifmtargexp{\hyxmp@today@xmp}{%
830     \@ifundefined{#1}{#2}%
831   }%
832 }%
833 }

```

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

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

Case 1: `\pdfcreationdate` is defined (pdfL^AT_EX and pre-0.85 LuaL^AT_EX).

```

835 \hyxmp@try@today{\pdfcreationdate}{%
836   \edef\hyxmp@today@xmp{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
837 }

```

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

```

838 \hyxmp@try@today{\pdffeedback}{%
839   \edef\hyxmp@today@xmp{\expandafter\hyxmp@pdf@to@xmp@date\pdffeedback creationdate}%
840 }

```

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

```

841 \hyxmp@try@today{\filemoddate}{%
842   \edef\hyxmp@today@xmp{\filemoddate{\hyxmp@jobname.log}}%
843   \edef\next{%
844     \edef\noexpand\hyxmp@today@xmp{\noexpand\hyxmp@as@xmp@date{\hyxmp@today@xmp}}%
845   }%
846   \next
847 }%

```

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

```

848 \hyxmp@try@today{\year}{%
849   \hyxmp@today@xmp@define\hyxmp@today@xmp
850 }

```

`\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`.

```
851 \expandafter\edef\expandafter\hyxmp@today@pdf\expandafter{%
852 \expandafter\hyxmp@xmp@to@pdf@date\expandafter{\hyxmp@today@xmp}%
853 }
```

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.

```
854 \catcode'\Q=3
```

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

```
855 \newcommand{\hyxmp@trimspaces}[1]{%
```

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

```
856 \begingroup
```

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

```
857 \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.

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

Transfer the trimmed text back into #1.

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

```
860 }
```

`\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.

```
861 \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.

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

```
863 \catcode'\Q=11
```


3.4.4 Converting text to XML

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

```
\ifhyxmp@unicodetex XYTeX and LuaTeX natively support Unicode. We define the conditional
\hyxmp@unicodetextrue \ifhyxmp@unicodetex to check for these so we can properly handle encoding
\hyxmp@unicodetexfalse 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.
```

```
864 \newif\ifhyxmp@unicodetex
865 \ifnum64='^^^^0040\relax
866 \hyxmp@unicodetextrue
867 \else
868 \hyxmp@unicodetexfalse
869 \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.

```
870 \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.

```
871 \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 &.

```
\hyxmp@xmlified
\hyxmp@text
872 \newcommand*{\hyxmp@xmlify}[1]{%
873 \gdef\hyxmp@xmlified{}
```

Escaped PDF string → PDFDocEncoding/Unicode

```
874 \EdefUnescapeString\hyxmp@text{#1}%
875 \ifhyxmp@unicodetex
```

PDFDocEncoding/Unicode → UTF-32BE

```
876 \hyxmp@is@unicode\hyxmp@text{%
877 \StringEncodingConvert
878 \hyxmp@text\hyxmp@text{utf16be}{utf32be}%
879 }{%
880 \ifXeTeX
881 \hyxmp@xetex@crap
882 \else
883 \StringEncodingConvert
884 \hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
885 \fi
886 }%
```

```

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

UTF-32BE → XML in ASCII
888   \edef\hyxmp@text{%
889     \expandafter
890   }\expandafter\hyxmp@toxml@unicodetex\hyxmp@text
891   \relax\relax\relax\relax\relax\relax\relax\relax
892   \else

PDFDocEncoding/Unicode → UTF-8
893   \hyxmp@is@unicode\hyxmp@text{%
894     \StringEncodingConvert
895     \hyxmp@text\hyxmp@text{utf16be}{utf8}%
896   }{%
897     \StringEncodingConvert
898     \hyxmp@text\hyxmp@text{pdfdoc}{utf8}%
899   }%

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

UTF-8 as hex string → XML in UTF-8 as hex string
901   \edef\hyxmp@text{%
902     \expandafter\hyxmp@toxml\hyxmp@text\@empty\@empty
903   }%

XML in UTF-8 as hex string → XML in UTF-8
904   \EdefUnescapeHex\hyxmp@text{\hyxmp@text}%
905   \fi
906   \global\let\hyxmp@xmlified\hyxmp@text
907 }

```

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

```

908 \begingroup
909   \lccode'\<=254 %
910   \lccode'\>=255 %
911   \catcode254=12 %
912   \catcode255=12 %
913 \lowercase{\endgroup
914   \def\hyxmp@is@unicode#1{%
915     \expandafter\hyxmp@@is@unicode#1<>\@nil
916   }%
917   \def\hyxmp@@is@unicode#1<>#2\@nil{%
918     \ifx\#1\%
919       \expandafter\@firstoftwo
920     \else
921       \expandafter\@secondoftwo
922     \fi
923   }%
924 }

```

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

```

925 \def\hyxmp@toxml#1#2{%
926   \ifx#1\@empty
927   \else
928     \ifnum"#1#2='\& %
929       26616D703B% &amp;
930     \else\ifnum"#1#2='\< %
931       266C743B% &lt;t;
932     \else\ifnum"#1#2='\> %
933       2667743B% &gt;t;
934     \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.

```

935   \@ifundefined{pdfmark}{%
936     #1#2%
937   }{%
938   \ifnum"#1#2='\( %
939     5C28% \ (
940   \else\ifnum"#1#2='\) %
941     5C29% \ )
942   \else
943     #1#2%
944   \fi\fi
945   }%
946   \fi\fi\fi
947   \expandafter\hyxmp@toxml
948   \fi
949 }

```

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

```

950 \def\hyxmp@toxml@unicodetex#1#2#3#4#5#6#7#8{%
951   \ifx#1\relax
952   \else
953     \ifnum"#1#2#3#4#5#6#7#8>127 %
954       \uccode'\*"#1#2#3#4#5#6#7#8\relax
955       \uppercase{%
956         \edef\hyxmp@text{\hyxmp@text *}%
957       }%
958     \else\ifnum"#7#8='\< %
959       \edef\hyxmp@text{\hyxmp@text &lt;t;}%
960     \else\ifnum"#7#8='\& %
961       \edef\hyxmp@text{\hyxmp@text &amp;}%
962     \else\ifnum"#7#8='\> %

```

```

963     \edef\hyxmp@text{\hyxmp@text &gt;}%
964     \else\ifnum"#7#8='\ %
965     \edef\hyxmp@text{\hyxmp@text\space}%
966     \else
967     \uccode'\*"#7#8\relax
968     \uppercase{%
969     \edef\hyxmp@text{\hyxmp@text *}%
970     }%
971     \fi\fi\fi\fi\fi
972     \expandafter\hyxmp@toxml@unicodetex
973     \fi
974 }

```

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

```

975 \def\hyxmp@skipzeros#1{%
976   \ifx#10%
977     \expandafter\hyxmp@skipzeros
978   \fi
979 }

```

`\x` In the case of $X_{\text{H}}\text{TEX}$, the strings defined by `\pdfstringdef` can contain big characters. In this case, the string is treated as Unicode.

```

\hyxmp@xetex@crap \hyxmp@try 980 \begingroup
\hyxmp@crap@result 981 \def\x#1{\endgroup
\hyxmp@text 982 \def\hyxmp@xetex@crap{%
983   \edef\hyxmp@try{%
984     \expandafter\hyxmp@SpaceOther\hyxmp@text#1\@nil
985   }%
986   \let\hyxmp@crap@result=N%
987   \expandafter\hyxmp@crap@test\hyxmp@try\relax
988   \ifx\hyxmp@crap@result Y%
989     \let\hyxmp@text\@empty
990     \expandafter\hyxmp@crap@convert\hyxmp@try\relax
991   \else
992     \StringEncodingConvert\hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
993   \fi
994 }%
995 }
996 \x{ }

```

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

```

997 \begingroup
998 \catcode'\~ =12 %
999 \lccode'\~ ='\ %
1000 \lowercase{\endgroup
1001 \def\hyxmp@SpaceOther#1 #2\@nil{%
1002   #1%
1003   \ifx\relax#2\relax
1004     \expandafter\@gobble
1005   \else
1006     ~%
1007     \expandafter\@firstofone
1008   \fi
1009   {\hyxmp@SpaceOther#2\@nil}%

```

```

1010 }%
1011 }

```

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

```

1012 \def\hyxmp@crap@test#1{%
1013   \ifx#1\relax
1014   \else
1015     \ifnum'#1>127 %
1016       \let\hyxmp@crap@result=Y%
1017       \expandafter\expandafter\expandafter\hyxmp@skiptorelax
1018     \else
1019       \expandafter\expandafter\expandafter\hyxmp@crap@test
1020   \fi
1021 \fi
1022 }

```

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

```

1023 \def\hyxmp@skiptorelax#1\relax{}

```

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

```

\hyxmp@num 1024 \def\hyxmp@crap@convert#1{%
\hyxmp@text 1025   \ifx#1\relax
1026   \else
1027     \edef\hyxmp@num{\number'#1}%
1028     \ifnum\hyxmp@num>"FFFFFF %
1029       \lccode'\!=\intcalcDiv{\hyxmp@num}{\number"1000000}\relax
1030       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
1031       \edef\hyxmp@num{\intcalcMod{\hyxmp@num}{\number"1000000}}%
1032     \else
1033       \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
1034     \fi
1035     \ifnum\hyxmp@num>"FFFF %
1036       \lccode'\!=\intcalcDiv{\hyxmp@num}{\number"10000}\relax
1037       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
1038       \edef\hyxmp@num{\intcalcMod{\hyxmp@num}{\number"10000}}%
1039     \else
1040       \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
1041     \fi
1042     \ifnum\hyxmp@num>"FF %
1043       \lccode'\!=\intcalcDiv{\hyxmp@num}{\number"100}\relax
1044       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
1045       \edef\hyxmp@num{\intcalcMod{\hyxmp@num}{\number"100}}%
1046     \else
1047       \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
1048     \fi
1049     \ifnum\hyxmp@num>0 %
1050       \lccode'\!=\hyxmp@num\relax
1051       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
1052     \else
1053       \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
1054     \fi
1055     \expandafter\hyxmp@crap@convert
1056 \fi
1057 }

```

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

```
1058 \begingroup
1059 \catcode0=12 %
1060 \gdef\hyxmp@zero{^^00}%
1061 \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.

```
1062 \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.

```
1063 \newcommand*{\hyxmp@add@simple}[2]{%
1064 \ifnotmtargexp{#2}{%
1065 \hyxmp@xmlify{#2}%
1066 \hyxmp@add@to@xml{\hyxmp@extra@indent_____<}%
1067 \xdef\hyxmp@xml{\hyxmp@xml#1}%
1068 \hyxmp@add@to@xml{>\hyxmp@xmlied</}%
1069 \xdef\hyxmp@xml{\hyxmp@xml#1>^^J}%
1070 }%
1071 }
```

`\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.

```
1072 \newcommand*{\hyxmp@add@simple@var}[2]{%
1073 \expandafter\ifx\csname#2\endcsname\relax
1074 \else
1075 \hyxmp@xmlify{\csname#2\endcsname}%
1076 \hyxmp@add@to@xml{%
1077 \hyxmp@extra@indent_____<#1>\hyxmp@xmlied</#1>^^J%
1078 }%
1079 \fi
1080 }
```

`\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.

```
1081 \newcommand*{\hyxmp@add@simple@lang}[2]{%
```

```

1082 \ifnotmtarg{#2}{%
1083   \hyxmp@xmlify{#2}%
1084   \expandafter\hyxmp@add@simple@lang@i\hyxmp@xmlified\relax{#1}%
1085 }%
1086 }

```

`\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.

```

1087 \newcommand*{\hyxmp@add@simple@lang@i}{%
1088   \ifnextchar[\hyxmp@add@simple@lang@ii{\hyxmp@add@simple@lang@ii[\@pdfmetalang]}%
1089 }

```

`\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.

```

1090 \def\hyxmp@add@simple@lang@ii[#1]#2\relax#3{%
1091   \ifnotmtarg{#2}{%
1092     \hyxmp@xmlify{#2}%
1093     \@ifmtarg{#1}{%
1094       \hyxmp@add@to+xml{%
1095         <#3>\hyxmp@xmlified</#3>^^J%
1096       }%
1097     }{%
1098       \hyxmp@add@to+xml{%
1099         <#3 xml:lang="#1">\hyxmp@xmlified</#3>^^J%
1100       }%
1101     }%
1102   }%
1103 }

```

`\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.

```

1104 \newcommand*{\hyxmp@add@simple@pfx}[3]{%
1105   \ifnotmtargexp{#3}{%
1106     \hyxmp@add@to+xml{\hyxmp@extra@indent_____<}%
1107     \xdef\hyxmp+xml{\hyxmp+xml#1}%
1108     \hyxmp@pdfstringdef\hyxmp@iprefix{#2}%
1109     \hyxmp@xmlify{\hyxmp@iprefix}%
1110     \hyxmp@add@to+xml{>\hyxmp@xmlified}%
1111     \hyxmp@xmlify{#3}%
1112     \hyxmp@add@to+xml{\hyxmp@xmlified<}%
1113     \xdef\hyxmp+xml{\hyxmp+xml#1>^^J}%
1114   }%
1115 }

```

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}`”.

```
1116 \def\hyxmp@alt@title{}
1117 \def\hyxmp@alt@description{}
1118 \def\hyxmp@alt@rights{}
```

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

```
1119 \newcommand{\hyxmp@LA@accept}[2]{%
1120 \define@key{hyxmp@LA}{#1}{%
```

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

```
1121 \hyxmp@pdfstringdef\hyxmp@value{##1}%
1122 \xdef#2{#2\noexpand\do {\hyxmp@cur@lang} {\hyxmp@value}}%
1123 }
1124 }
```

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

```
1125 \hyxmp@LA@accept{pdftitle}{\hyxmp@alt@title}
1126 \hyxmp@LA@accept{pdfsubject}{\hyxmp@alt@description}
1127 \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.

```
1128 \newcommand{\XMPLangAlt}[2]{%
1129 \let\do=\relax
```

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

```
1130 \edef\hyxmp@cur@lang{#1}%
1131 \setkeys{hyxmp@LA}{#2}%
1132 }
```

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

```

1133 \def\hyxmp@modulo@a#1{%
1134   \@tempcntb=\@hyxmp@count
1135   \divide\@tempcntb by #1
1136   \multiply\@tempcntb by #1
1137   \advance\@hyxmp@count by -\@tempcntb
1138 }

```

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

```

\hyxmp@big@prime@ii 1139 \def\hyxmp@big@prime{536870923}
1140 \def\hyxmp@big@prime@ii{536870027}

```

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

```

\hyxmp@one@token 1141 \def\hyxmp@seed@rng#1{%
1142   \@hyxmp@count=\hyxmp@big@prime
1143   \futurelet\hyxmp@one@token\hyxmp@seed@rng@i#1\@empty
1144 }

```

`\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 $\@hyxmp@count \leftarrow 3 \cdot \@hyxmp@count + c \pmod{\hyxmp@big@prime}$.

```

\next 1145 \def\hyxmp@seed@rng@i{%
1146   \ifx\hyxmp@one@token\@empty
1147     \let\next=\relax
1148   \else
1149     \def\next##1{%
1150       \multiply\@hyxmp@count by 3
1151       \advance\@hyxmp@count by '##1
1152       \hyxmp@modulo@a{\hyxmp@big@prime}%
1153       \futurelet\hyxmp@one@token\hyxmp@seed@rng@i
1154     }%
1155   \fi
1156   \next
1157 }

```

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

```

1158 \def\hyxmp@set@rand@num{%
1159   \@hyxmp@count=\hyxmp@rand@num
1160   \multiply\@hyxmp@count by 3
1161   \advance\@hyxmp@count by \hyxmp@big@prime@ii
1162   \hyxmp@modulo@a{\hyxmp@big@prime}%
1163   \xdef\hyxmp@rand@num{the\@hyxmp@count}%
1164 }

```

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

```

1165 \def\hyxmp@append@hex#1{%
1166   \hyxmp@set@rand@num
1167   \@hyxmp@count=\hyxmp@rand@num
1168   \hyxmp@modulo@a{16}%

```

```

1169 \ifnum \@hyxmp@count < 10
1170   \xdef #1{#1\the \@hyxmp@count}%
1171 \else

```

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

```

1172   \advance \@hyxmp@count by -10
1173   \ifcase \@hyxmp@count
1174     \xdef #1{#1a}%
1175     \or \xdef #1{#1b}%
1176     \or \xdef #1{#1c}%
1177     \or \xdef #1{#1d}%
1178     \or \xdef #1{#1e}%
1179     \or \xdef #1{#1f}%
1180   \fi
1181 \fi
1182 }

```

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

```

1183 \def \hyxmp@append@hex@iii#1{%
1184   \hyxmp@append@hex#1%
1185   \hyxmp@append@hex#1%
1186   \hyxmp@append@hex#1%
1187 }

```

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

```

1188 \def \hyxmp@append@hex@iv#1{%
1189   \hyxmp@append@hex@iii#1%
1190   \hyxmp@append@hex#1%
1191 }

```

`\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 `\@hyxmp@count` and `\@tempcntb`.

```

1192 \def \hyxmp@create@uuid#1{%
1193   \def #1{uuid:}%
1194   \hyxmp@append@hex@iv#1%
1195   \hyxmp@append@hex@iv#1%
1196   \g@addto@macro#1{-}%
1197   \hyxmp@append@hex@iv#1%
1198   \g@addto@macro#1{-4}%
1199   \hyxmp@append@hex@iii#1%
1200   \g@addto@macro#1{-}%

```

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

```

1201   \hyxmp@set@rand@num
1202   \@hyxmp@count = \hyxmp@rand@num
1203   \hyxmp@modulo@a{4}%
1204   \ifcase \@hyxmp@count
1205     \g@addto@macro#1{8}%
1206     \or \g@addto@macro#1{9}%
1207     \or \g@addto@macro#1{a}%
1208     \or \g@addto@macro#1{b}%

```

```

1209 \fi
1210 \hyxmp@append@hex@iii#1%
1211 \g@addto@macro#1{-}%
1212 \hyxmp@append@hex@iv#1%
1213 \hyxmp@append@hex@iv#1%
1214 \hyxmp@append@hex@iv#1%
1215 }

```

`\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.

```

1216 \newcommand*{\hyxmp@def@DocumentID}{%
1217 \edef\hyxmp@seed@string{\hyxmp@jobname:\@pdftitle:\@pdfauthor:}%
1218 \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
1219 \edef\hyxmp@rand@num{\the\@hyxmp@count}%
1220 \hyxmp@create@uuid\hyxmp@DocumentID
1221 }

```

`\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).

```

1222 \newcommand*{\hyxmp@def@InstanceID}{%
1223 \hyxmp@today@xmp@define{\hyxmp@seed@string}%
1224 \edef\hyxmp@seed@string{%
1225 \hyxmp@jobname:\@pdftitle:\@pdfauthor:\hyxmp@today@xmp:\hyxmp@seed@string
1226 }%
1227 \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
1228 \edef\hyxmp@rand@num{\the\@hyxmp@count}%
1229 \hyxmp@create@uuid\hyxmp@InstanceID
1230 }

```

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.

```

1231 \newcommand*{\hyxmp@add@to@xml}[1]{%
1232   \bgroup
1233     \@hyxmp@count=0
1234     \ifhyxmp@unicodetex
1235       \@tempcntb=65536%
1236     \else
1237       \@tempcntb=256%
1238     \fi
1239     \loop
1240       \lccode\@hyxmp@count=\@hyxmp@count
1241       \advance\@hyxmp@count by 1
1242       \ifnum\@hyxmp@count<\@tempcntb
1243     \repeat
1244     \lccode'\_='\ \relax
1245     \lccode'\^C='\,\relax
1246     \lccode'\^U='\_\relax
1247     \lowercase{\xdef\hyxmp@new@xml{#1}}%
1248     \xdef\hyxmp@xml{\hyxmp@xml\hyxmp@new@xml}%
1249   \egroup
1250 }
```

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

```

1251 \bgroup
1252 \catcode'\#=11
1253 \gdef\hyxmp@hash{#}
1254 \egroup
```

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

```

1255 \bgroup
1256 \xdef\hyxmp@xml{}%
1257 \hyxmp@add@to@xml{%
1258 -----^^J%
1259 }
1260 \xdef\hyxmp@padding{\hyxmp@xml}%
1261 \egroup
1262 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
1263 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
1264 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
1265 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
1266 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
```

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

```

1267 \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.
1268 \newcommand*{\hyxmp@define@pdfproducer}{%
1269   \gdef\@pdfproducer{TeX}
1270   \ifLuaTeX
1271     \expandafter\hyxmp@banner@to@producer\expandafter{\luatexbanner}%
1272   \else
1273     \ifPDFTeX
1274       \expandafter\hyxmp@banner@to@producer\expandafter{\pdftexbanner}%
1275     \else
1276       \ifXeTeX
1277         \edef\@pdfproducer{XeTeX version \the\XeTeXversion\XeTeXrevision}%
1278       \fi
1279     \fi
1280   \fi
1281 }

\@pdfproducer Define \@pdfproducer as the TEX engine's banner string (e.g., "This is
\hyxmp@banner@to@producer LuaHBTeX, Version 1.17.0 (TeX Live 2023)"), removing the initial "This is"
if possible (specifically, when  $\varepsilon$ -TEX's \scantokens primitive is available).
1282 \def\hyxmp@banner@to@producer#1{%
1283   \ifx\scantokens\relax
1284     \gdef\@pdfproducer{#1}%
1285   \else
1286     {\scantokens{\makeatletter\hyxmp@remove@this#1\relax}}%
1287   \fi
1288 }

\@pdfproducer Define \@pdfproducer as a given banner string with the initial "This is" stripped
\hyxmp@remove@this off the beginning.
1289 \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.
1290 \AtBeginDocument{%
1291   \ifx\@pdfproducer\relax
1292     \hyxmp@define@pdfproducer
1293   \fi
1294 }

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

```

`\hyxmp@major@minor`

```
1295 \newcommand*{\hyxmp@assign@major@minor}{%
1296   \@ifundefined{pdfvariable}{%
1297     \@ifundefined{pdfminorversion}{%
Case 1: Neither \pdfvariable nor \pdfminorversion is defined (XeLaTeX and
regular LATEX).
```

```
1298   }{%
```

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

```
1299     \xdef\hyxmp@major@minor{\the\pdfminorversion}%
1300     \@ifundefined{pdfmajorversion}{%
```

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

```
1301       \xdef\hyxmp@major@minor{1.\hyxmp@major@minor}%
1302     }{%
```

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

```
1303       \xdef\hyxmp@major@minor{\the\pdfmajorversion.\hyxmp@major@minor}%
1304     }%
1305   }%
1306 }{%
```

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

```
1307     \xdef\hyxmp@major@minor{\the\pdfvariable majorversion.\the\pdfvariable minorversion}%
1308   }%
1309 }
```

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

```
1310 \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.

```
1311   \hyxmp@add@simple@var{pdf:Producer}{@pdfproducer}%
1312   \hyxmp@add@simple@var{pdf:Keywords}{@pdfkeywords}%
1313   \hyxmp@add@simple{pdf:Trapped}{\@pdftrapped}%
1314   \hyxmp@assign@major@minor
1315   \hyxmp@add@simple@var{pdf:PDFVersion}{hyxmp@major@minor}%
1316 }
```

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.

```
1317 \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.

```
1318 \newcommand*{\hyxmp@rdf@dc}[3][\iffalse]{%
```

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

```
1319 \@ifmtargexp{#3}{\@tempswafalse}{\@tempswatru}%
```

```
1320 #1
```

```
1321 \@tempswatru
```

```
1322 \fi
```

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

```
1323 \if@tempswa
```

```
1324 \hyxmp@xmlify{#3}%
```

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

```
1325 \let\hyxmp@value=\hyxmp@xmlified
```

```
1326 \hyxmp@add@to@xml{%
```

```
1327 -----<dc:#2>^^J%
```

```
1328 -----<rdf:Alt>^^J%
```

```
1329 }%
```

Record whether property #2 has definitions in multiple languages.

```
1330 \@if@def@and@nonempty{\hyxmp@alt@#2}{%
```

```
1331 \hyxmp@multi@langstrue
```

```
1332 }{%
```

```
1333 \hyxmp@multi@langfalse
```

```
1334 }%
```

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

```
1335 \ifx\@pdfmetalang\hyxmp@x@default
```

```
1336 \ifhyxmp@multi@langs
```

Case 1: No `pdfmetalang` but `\XMLLangAlt`. 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.

```
1337 \PackageError{hyperxmp}%
```

```
1338 {\string\XMLLangAlt\space was used without specifying
```

```
1339 pdfmetalang\MessageBreak
```

```
1340 or pdflang}%
```

```
1341 {Be sure to assign a language code to the pdfmetalang key and/or a
```

```
1342 document\MessageBreak
```

```
1343 language to the pdflang key (e.g., \string\hypersetup{pdfmetalang={en}}).}%
```

```
1344 \else
```

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

```

1345     \hyxmp@add@to@xml{%
1346 -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@value</rdf:li>^^J%
1347     }%
1348     \fi
1349     \else
1350     \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.

```

1351     \hyxmp@xmlify{\@pdfmetalang}%
1352     \hyxmp@add@to@xml{%
1353 -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@value</rdf:li>^^J%
1354 -----<rdf:li xml:lang="\hyxmp@xmlified">\hyxmp@value</rdf:li>^^J%
1355     }%
1356     \def\do##1##2{
1357         \hyxmp@xmlify{##2}%
1358         \hyxmp@add@to@xml{%
1359 -----<rdf:li xml:lang="##1">\hyxmp@xmlified</rdf:li>^^J%
1360         }%
1361     }%
1362     \csname hyxmp@alt@#2\endcsname
1363     \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.

```

1364     \hyxmp@xmlify{\@pdfmetalang}%
1365     \hyxmp@add@to@xml{%
1366 -----<rdf:li xml:lang="\hyxmp@xmlified">\hyxmp@value</rdf:li>^^J%
1367     }%
1368     \fi
1369     \fi

```

Complete this XMP element.

```

1370     \hyxmp@add@to@xml{%
1371 -----</rdf:Alt>^^J%
1372 -----</dc:#2>^^J%
1373     }%
1374     \fi
1375 }%

```

`\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.

```

1376 \newcommand*{\hyxmp@list@to@xml}[4][\iffalse]{%

```

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

```

1377 \@ifmtargexp{#4}{\@tempwafalse}{\@tempwattrue}%
1378 #1
1379     \@tempwattrue
1380     \fi

```

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


```

1381 \if@tempwa
1382 \hyxmp@add@to@xml{%
1383 -----<dc:#2>^^J%
1384 -----<rdf:#3>^^J%
1385 }%
1386 \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`.

```

1387 \hyxmp@xmlify{#4}%
1388 \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
1389 \def\@elt##1{%
1390 \hyxmp@add@to@xml{%
1391 -----<rdf:li>##1</rdf:li>^^J%
1392 }%
1393 }%
1394 \hyxmp@list
1395 \egroup
1396 \hyxmp@add@to@xml{%
1397 -----</rdf:#3>^^J%
1398 -----</dc:#2>^^J%
1399 }%
1400 \fi
1401 }

```

`\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.

```

1402 \newcommand{\hyxmp@singleton@dc}[3][Bag]{%
1403 \@ifnotmtargexp{#3}{%
1404 \hyxmp@xmlify{#3}%
1405 \hyxmp@add@to@xml{%
1406 -----<dc:#2>^^J%
1407 -----<rdf:#1>^^J%
1408 -----<rdf:li>\hyxmp@xmlified</rdf:li>^^J%
1409 -----</rdf:#1>^^J%
1410 -----</dc:#2>^^J%
1411 }%
1412 }
1413 }

```

`\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).

```

1414 \newcommand*{\hyxmp@cond@dc@identifier}[2]{%
1415 \ifx\hyxmp@xmlified\@empty
1416 \@ifnotmtargexp{#2}{%
1417 \hyxmp@add@simple@pfx{dc:identifier}{#1}{#2}%
1418 }%
1419 \fi
1420 }

```

`\hyxmp@dc@schema` Add properties defined by the Dublin Core schema to the `\hyxmp@xml` macro. Specifically, we add entries for the `dc:title` property if the author specified a

pdftitle, the dc:description property if the author specified a pdfsubject, the dc:rights property if the author specified a pdfcopyright, the dc:creator property if the author specified a pdfauthor, the dc:subject property if the author specified pdfkeywords, the dc:language property if the author specified pdflang, the dc:type property if the author specified pdftype, and the dc:identifier if the author specified pdfidentifier or if we can derive it from other options. We also specify the dc:source property using the base name of the source file with .tex appended and the dc:date property using the date the document was run through L^AT_EX—unless the author specified pdfdate, in which case we use that.

```

1421 \newcommand*{\hyxmp@dc@schema}{%
1422   \hyxmp@add@simple{dc:format}{application/pdf}%
1423   \hyxmp@rdf@dc[\ifHy@pdfa]{title}{\@pdftitle}%
1424   \hyxmp@rdf@dc[\ifHy@pdfa]{description}{\@pdfsubject}%
1425   \hyxmp@rdf@dc{rights}{\@pdfcopyright}%
1426   \hyxmp@singleton@dc{publisher}{\@pdfpublisher}%
1427   \@ifmtargexp{\@pdfdatetime}{%
1428     \hyxmp@singleton@dc[Seq]{date}{\hyxmp@today@xmp}%
1429   }{%
1430     \hyxmp@singleton@dc[Seq]{date}{\@pdfdatetime}%
1431   }%
1432   \hyxmp@singleton@dc{type}{\@pdftype}%
1433   \hyxmp@list@to@xml[\ifHy@pdfa]{creator}{Seq}{\hyxmp@pdfauthor}%
1434   \hyxmp@list@to@xml{subject}{Bag}{\hyxmp@pdfkeywords}%
1435   \ifx\@pdfsource\@empty
1436     \else
1437       \hyxmp@add@simple{dc:source}{\@pdfsource}%
1438     \fi
1439   \hyxmp@list@to@xml{language}{Bag}{\hyxmp@dc@lang}%
1440 % If |\@pdfidentifier| is empty, try setting it to each of |\@pdfdoi|,
1441 % |\@pdfeissn|, |\@pdfissn|, and |\@pdfisbn|, in turn, with proper
1442 % syntactic adjustments.
1443 %   \begin{macrocode}
1444   \@ifmtargexp{\@pdfidentifier}{%
1445     \let\hyxmp@xmlified=\@empty
1446     \hyxmp@cond@dc@identifier{info:doi/}{\@pdfdoi}%
1447     \hyxmp@cond@dc@identifier{urn:ISSN:}{\@pdfeissn}%
1448     \hyxmp@cond@dc@identifier{urn:ISSN:}{\@pdfissn}%
1449     \hyxmp@cond@dc@identifier{urn:ISBN:}{\@pdfisbn}%
1450   }{%
1451     \hyxmp@add@simple{dc:identifier}{\@pdfidentifier}%
1452   }%
1453 }

```

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.

```

1454 \newcommand*{\hyxmp@xmpRights@schema}{%

```

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

```
1455 \let\hyxmp@rights=\@empty
1456 \ifx\@pdflicenseurl\@empty
1457 \else
1458   \def\hyxmp@rights{YES}%
1459 \fi
1460 \ifx\@pdfcopyright\@empty
1461 \else
1462   \def\hyxmp@rights{YES}%
1463 \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.

```
1464 \ifx\hyxmp@rights\@empty
1465 \else
1466   \ifx\@pdfcopyright\@empty
1467   \else
1468     \hyxmp@add@simple{xmpRights:Marked}{True}%
1469   \fi
1470   \hyxmp@add@simple{xmpRights:WebStatement}{\@pdflicenseurl}%
1471 \fi
1472 }
```

3.6.5 The XMP Media Management schema

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

```
1473 \expandafter\hyxmp@aep@toks\expandafter=\expandafter{%
1474 \the\hyxmp@aep@toks
1475 \AtEndPreamble{%
1476   \@ifmtargexp{\hyxmp@DocumentID}{\hyxmp@def@DocumentID}{}%
1477   \@ifmtargexp{\hyxmp@InstanceID}{\hyxmp@def@InstanceID}{}%
1478 }%
1479 }
```

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

```
1480 \gdef\hyxmp@mm@schema{%
1481   \hyxmp@add@simple{xmpMM:DocumentID}{\hyxmp@DocumentID}%
1482   \hyxmp@add@simple{xmpMM:InstanceID}{\hyxmp@InstanceID}%
1483   \hyxmp@add@simple{xmpMM:VersionID}{\@pdfversionid}%
1484   \hyxmp@add@simple{xmpMM:RenditionClass}{\@pdfrendition}%
1485 }
```

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`.

```
1486 \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`.

```
1487 \ifmtargexp{\@pdfcreationdate}{%
1488   \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@today@xmp}%
1489 }{%
1490   \hyxmp@add@simple{xmp:CreateDate}{%
1491     \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfcreationdate}}%
1492 }%
```

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

```
1493 \ifmtargexp{\@pdfmoddate}{%
1494   \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@today@xmp}%
1495 }{%
1496   \hyxmp@add@simple{xmp:ModifyDate}{%
1497     \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfmoddate}}%
1498 }%
```

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

```
1499 \ifmtargexp{\@pdfmetadatetime}{%
1500   \hyxmp@add@simple{xmp:MetadataDate}{\hyxmp@today@xmp}%
1501 }{%
1502   \hyxmp@add@simple{xmp:MetadataDate}{\@pdfmetadatetime}%
1503 }%
```

Define the creation tool and the base URL.

```
1504 \hyxmp@add@simple{xmp:CreatorTool}{\@pdfcreator}%
1505 \hyxmp@add@simple{xmp:BaseURL}{\@baseurl}%
1506 }
```

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.

```
1507 \gdef\hyxmp@photoshop@schema{%
1508   \edef\hyxmp@photoshop@data{\@pdfauthortitle\@pdfcaptionwriter}%
1509   \hyxmp@add@simple{photoshop:AuthorsPosition}{\@pdfauthortitle}%
1510   \hyxmp@add@simple{photoshop:CaptionWriter}{\@pdfcaptionwriter}%
1511 }
```

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.

```

1512 \newcommand*{\hyxmp@pdfa@id@schema}{%
1513   \ifHy@pdfa
1514     \hyxmp@add@simple{pdfaid:part}{\@pdfapart}%
1515     \hyxmp@add@simple{pdfaid:conformance}{\@pdfaconformance}%
1516   \fi
1517 }
```

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

```

1518 \newcommand*{\hyxmp@pdfua@id@schema}{%
1519   \hyxmp@add@simple{pdfuaid:part}{\@pdfuapart}%
1520 }
```

`\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.

```

1521 \newcommand*{\hyxmp@pdfx@id@schema}{%
1522   \@hyxmp@count=0\hyxmp@pdfx@major\relax
1523   \ifnum\@hyxmp@count=0
1524     \else
1525       \ifnum\@hyxmp@count=1
1526         \hyxmp@add@simple{pdfx:GTS_PDFXVersion}{PDF/X-1:2001}%
1527         \hyxmp@add@simple{pdfx:GTS_PDFXConformance}{\@pdfxstandard}%
1528       \else
1529         \ifnum\@hyxmp@count<4
1530           \hyxmp@add@simple{pdfx:GTS_PDFXVersion}{\@pdfxstandard}%
1531         \else
1532           \hyxmp@add@simple{pdfxid:GTS_PDFXVersion}{\@pdfxstandard}%
1533         \fi
1534       \fi
1535     \fi
1536 }
```

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.

```

1537 \begingroup
1538   \catcode'\&=12
1539   \catcode'\#=12
1540   \gdef\xmplinesep{&#xA;}
1541 \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.

```

1542 \newcommand*{\hyxmp@list@to@lines}[2]{%
1543   \@ifnotmtargexp{#2}{%
1544     \bgroup
1545     \hyxmp@add@to+xml{%
```

```

1546     \hyxmp@extra@indent_____<#1>%
1547     }%

```

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

```

1548     \def\@elt@first##1{%
1549         \hyxmp@add@to+xml{##1}%
1550         \let\@elt=\@elt@rest
1551     }%

```

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

```

1552     \def\@elt@rest##1{%
1553         \hyxmp@add@to+xml{\xmplinesep##1}%
1554     }%

```

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

```

1555     \let\@elt=\@elt@first
1556     \hyxmp@xmlify{#2}%
1557     \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
1558     \hyxmp@list
1559     \hyxmp@add@to+xml{</#1>^^J}%
1560     \egroup
1561 }%
1562 }

```

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

```

1563 \gdef\hyxmp@iptc@schema{%

```

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

```

1564 \ifx\hyxmp@iptc@data\@empty
1565 \else
1566     \hyxmp@add@to+xml{%
1567     _____<Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">^^J%
1568     }%

```

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

```

1569     \bgroup
1570     \edef\hyxmp@extra@indent{\hyxmp@extra@indent\space\space}%
1571     \hyxmp@list@to@lines{Iptc4xmpCore:CiAdrExtadr}{\@pdfcontactaddress}%
1572     \hyxmp@add@simple{Iptc4xmpCore:CiAdrCity}{\@pdfcontactcity}%
1573     \hyxmp@add@simple{Iptc4xmpCore:CiAdrRegion}{\@pdfcontactregion}%
1574     \hyxmp@add@simple{Iptc4xmpCore:CiAdrPcode}{\@pdfcontactpostcode}%
1575     \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.

```

1576     \def\xmplinesep{,}%
1577     \hyxmp@list@to@lines{Iptc4xmpCore:CTelWork}{\@pdfcontactphone}%
1578     \hyxmp@list@to@lines{Iptc4xmpCore:CiEmailWork}{\@pdfcontactemail}%
1579     \hyxmp@list@to@lines{Iptc4xmpCore:CIUrlWork}{\@pdfcontacturl}%
1580     \egroup
1581     \hyxmp@add@to@xml{%
1582     -----</Iptc4xmpCore:CreatorContactInfo>^^J%
1583     }%
1584     \fi
1585 }

```

3.6.10 The PRISM Basic Metadata schema

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

```

1586 \newcommand*{\hyxmp@prism@schema}{%
1587   \ifx\hyxmp@prism@data\@empty
1588   \else
1589     \hyxmp@add@simple{prism:complianceProfile}{three}%
1590     \fi
1591     \hyxmp@add@simple@lang{prism:subtitle}{\@pdfsubtitle}%
1592     \hyxmp@add@simple@lang{prism:publicationName}{\@pdfpublication}%
1593     \hyxmp@add@simple{prism:aggregationType}{\@pdfpubtype}%
1594     \hyxmp@add@simple@lang{prism:bookEdition}{\@pdfbookedition}%
1595     \hyxmp@add@simple{prism:volume}{\@pdfvolumenum}%
1596     \hyxmp@add@simple{prism:number}{\@pdfissuenum}%
1597     \hyxmp@add@simple{prism:pageRange}{\@pdfpagerange}%
1598     \hyxmp@add@simple{prism:isbn}{\@pdfisbn}%
1599     \hyxmp@add@simple{prism:issn}{\@pdfissn}%
1600     \hyxmp@add@simple{prism:eIssn}{\@pdfeissn}%
1601     \hyxmp@add@simple{prism:doi}{\@pdfdoi}%
1602     \hyxmp@add@simple{prism:url}{\@pdfurl}%
1603     \hyxmp@add@simple{prism:byteCount}{\@pdfbytes}%
1604     \hyxmp@add@simple{prism:pageCount}{\@pdfnumpages}%
1605 }

```

3.6.11 The Journal Article Versions (JAV) schema

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

```

1606 \newcommand*{\hyxmp@jav@schema}{%
1607   \hyxmp@add@simple{jav:journal_article_version}{\@pdfpubstatus}%
1608 }

```

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 Lua_{TEX} mechanisms and therefore include much less information when run from other _{TEX} engines.

```

1609 \newcommand*{\hyxmp@xmptpg@schema}{%
1610   \ifLuaTeX
1611     \luadirect{write_xmp_font_list(\the\hyxmp@cct)}%
1612   \fi
1613   \hyxmp@add@simple{xmpTPg:NPages}{\@pdfnumpages}%
1614 }

```

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

```

1615 \ifLuaTeX
1616   \newcatcodetable\hyxmp@cct
1617   \savecatcodetable\hyxmp@cct
1618 \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.

```

1619 \bgroup
1620   \catcode'\_ =11
1621   \gdef\hyxmp@prot@us_{_}%
1622 \egroup

```

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

```

1623 \ifLuaTeX
1624   \begin{luacode*}
1625   function write_xmp_font_list (cct)
1626     local function show_field(name, ...)
1627       for i = 1, select("#", ...) do
1628         local val = select(i, ...)
1629         if val then
1630           local xml = string.gsub(val, "&", "&amp;")
1631           xml = string.gsub(xml, "<", "&lt;")
1632           xml = string.gsub(xml, ">", "&gt;")
1633           xml = string.gsub(xml, "_", "\\hyxmp@prot@us ")
1634           tex.print(cct, "_____<stFnt:" .. name .. "> ..
1635                       xml .. "</stFnt:" .. name .. ">^^J%")
1636         end
1637       end
1638     end
1639   end
1640   tex.print(cct, "\\hyxmp@add@to@xml{%")
1641   tex.print(cct, "_____<xmpTPg:Fonts>^^J%")
1642   tex.print(cct, "_____<rdf:Bag>^^J%")
1643   for i, f in font.each() do
1644     tex.print(cct, "_____<rdf:li rdf:parseType=\"Resource\">^^J%")
1645     if f.filename then
1646       local fname = string.gsub(f.filename, "^harfloaded:(.*)", "%1")
1647       local info = fontloader.info(fname)
1648       if info then
1649         show_field("fontFace", info.fullname)
1650         show_field("fontFamily", info.familyname)

```



```

1651     show_field("fontName", info.fontname)
1652     show_field("versionString", info.version)
1653   end
1654   local baseName = string.gsub(f.filename, ".*[/\\](.*)", "%1")
1655   show_field("fontFileName", baseName)
1656   else
1657     show_field("fontName", f.psname, f.fullname, f.name)
1658   end
1659   if f.format and f.format ~= "unknown" then
1660     show_field("fontType", f.format)
1661   end
1662   tex.print(cct, "_____</rdf:li>^^J%")
1663 end
1664 tex.print(cct, "_____</rdf:Bag>^^J%")
1665 tex.print(cct, "_____</xmpTPg:Fonts>^^J%")
1666 tex.print(cct, "}")
1667 end
1668 \end{luacode*}
1669 \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 by the document.

```

1670 \newcommand*{\hyxmp@check@iptc@data}{%
1671   \edef\hyxmp@iptc@data{%
1672     \@pdfcontactaddress
1673     \@pdfcontactcity
1674     \@pdfcontactregion
1675     \@pdfcontactpostcode
1676     \@pdfcontactcountry
1677     \@pdfcontactphone
1678     \@pdfcontactemail
1679     \@pdfcontacturl
1680   }%
1681 }%

```

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

```

1682 \newcommand*{\hyxmp@check@prism@data}{%
1683   \edef\hyxmp@prism@data{%
1684     \@pdfbookedition
1685     \@pdfbytes
1686     \@pdfdoi
1687     \@pdfeissn
1688     \@pdfisbn
1689     \@pdfissn

```

```

1690 \pdfissuenum
1691 \pdfnumpages
1692 \pdfpagerange
1693 \pdfpublication
1694 \pdfpubtype
1695 \pdfsubtitle
1696 \pdfurl
1697 \pdfvolumenum
1698 }%
1699 }%

```

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

```

\hyxmp@jav@data
1700 \newcommand*{\hyxmp@check@jav@data}{%
1701 \edef\hyxmp@jav@data{%
1702 \pdfpubstatus
1703 }%
1704 }

```

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

```

1705 \newcommand*{\hyxmp@begin@extension@decls}{%
1706 \hyxmp@add@to@xml{%
1707 _____<pdfaExtension:schemas>^^J%
1708 _____<rdf:Bag>^^J%
1709 }%
1710 }

```

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

```

1711 \newcommand*{\hyxmp@end@extension@decls}{%
1712 \hyxmp@add@to@xml{%
1713 _____</rdf:Bag>^^J%
1714 _____</pdfaExtension:schemas>^^J%
1715 }%
1716 }

```

`\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.

```

1717 \newcommand*{\hyxmp@begin@ext@decl}[3]{%
1718 \hyxmp@add@to@xml{%
1719 _____<rdf:li rdf:parseType="Resource">^^J%
1720 _____<pdfaSchema:schema>#1</pdfaSchema:schema>^^J%
1721 _____<pdfaSchema:prefix>#2</pdfaSchema:prefix>^^J%
1722 _____<pdfaSchema:namespaceURI>#3</pdfaSchema:namespaceURI>^^J%
1723 _____<pdfaSchema:property>^^J%
1724 _____<rdf:Seq>^^J%
1725 }%
1726 }%

```

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

```

1727 \newcommand*{\hyxmp@end@ext@decl}{%
1728 \hyxmp@add@to@xml{%
1729 _____</rdf:Seq>^^J%

```

```

1730 -----</pdfaSchema:property>^^J%
1731 -----</rdf:li>^^J%
1732   }%
1733 }%

```

`\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.

```

1734 \newcommand{\hyxmp@declare@property}[4][Text]{%
1735   \hyxmp@add@to@xml{%
1736     -----<rdf:li rdf:parseType="Resource">^^J%
1737     -----<pdfaProperty:name>}%
1738   \xdef\hyxmp@xml{\hyxmp@xml#2}%
1739   \hyxmp@add@to@xml{</pdfaProperty:name>}%
1740     -----<pdfaProperty:valueType>#1</pdfaProperty:valueType>^^J%
1741     -----<pdfaProperty:category>#3</pdfaProperty:category>^^J%
1742     -----<pdfaProperty:description>#4</pdfaProperty:description>^^J%
1743     -----</rdf:li>^^J%
1744   }%
1745 }%

```

`\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.

```

1746 \newcommand{\hyxmp@declare@field}[3][Text]{%
1747   \hyxmp@add@to@xml{%
1748     -----<rdf:li rdf:parseType="Resource">^^J%
1749     -----<pdfaField:name>#2</pdfaField:name>^^J%
1750     -----<pdfaField:valueType>#1</pdfaField:valueType>^^J%
1751     -----<pdfaField:description>#3</pdfaField:description>^^J%
1752     -----</rdf:li>^^J%
1753   }%
1754 }

```

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

```

1755 \newcommand*{\hyxmp@pdf@extensions}{%
1756   \hyxmp@begin@ext@decl
1757     {Adobe PDF Schema}%
1758     {pdf}%
1759     {http://ns.adobe.com/pdf/1.3/}%
1760   \hyxmp@declare@property
1761     {Trapped}%
1762     {internal}%
1763     {Indication if the document has been modified to include trapping information}%
1764   \hyxmp@end@ext@decl
1765 }%

```

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

```

1766 \newcommand*{\hyxmp@mm@extensions}{%
1767   \hyxmp@begin@ext@decl
1768     {XMP Media Management Schema}%
1769     {xmpMM}%
1770     {http://ns.adobe.com/xap/1.0/mm/}%

```

```

1771 \hyxmp@declare@property
1772     [URI]
1773     {DocumentID}%
1774     {internal}%
1775     {UUID based identifier for all versions and renditions of a document}%
1776 \hyxmp@declare@property
1777     [URI]
1778     {InstanceID}%
1779     {internal}%
1780     {UUID based identifier for specific incarnation of a document}%
1781 \hyxmp@declare@property
1782     {VersionID}%
1783     {internal}%
1784     {Document version identifier}%
1785 \hyxmp@declare@property
1786     [RenditionClass]%
1787     {RenditionClass}%
1788     {internal}%
1789     {The manner in which a document is rendered}%
1790 \hyxmp@end@ext@decl
1791 }%

```

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

```

1792 \newcommand*{\hyxmp@pdfa@id@extensions}{%
1793   \hyxmp@begin@ext@decl
1794     {PDF/A Identification Schema}%
1795     {pdfaid}%
1796     {http://www.aiim.org/pdfa/ns/id/}%
1797   \hyxmp@declare@property
1798     [Integer]%
1799     {part}%
1800     {internal}%
1801     {Part of PDF/A standard}%
1802   \hyxmp@declare@property
1803     {conformance}%
1804     {internal}%
1805     {Conformance level of PDF/A standard}%
1806   \hyxmp@end@ext@decl
1807 }%

```

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

```

1808 \newcommand*{\hyxmp@pdfua@id@extensions}{%
1809   \hyxmp@begin@ext@decl
1810     {PDF/UA Universal Accessibility Schema}%
1811     {pdfuaid}%
1812     {http://www.aiim.org/pdfua/ns/id/}%
1813   \hyxmp@declare@property
1814     [Integer]%
1815     {part}%
1816     {internal}%
1817     {Part of ISO 14289 standard}%
1818   \hyxmp@end@ext@decl
1819 }%

```

`\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.

```

1820 \newcommand*{\hyxmp@pdfx@id@extensions}{%
1821   \ifx\hyxmp@pdfx@major\empty
1822   \else
1823     \hyxmp@begin@ext@decl
1824       {Adobe Document Info PDF/X eXtension Schema}%
1825       {pdfx}%
1826       {http://ns.adobe.com/pdfx/1.3/}%
1827     \hyxmp@declare@property
1828       {GTS_PDFXVersion}%
1829       {internal}%
1830       {ID of PDF/X standard}%
1831     \hyxmp@declare@property
1832       {GTS_PDFXConformance}%
1833       {internal}%
1834       {Conformance level of PDF/X standard}%
1835     \hyxmp@end@ext@decl
1836   \fi

```

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

```

1837 \@hyxmp@count=0\hyxmp@pdfx@major\relax
1838 \ifnum\@hyxmp@count>3
1839   \hyxmp@begin@ext@decl
1840     {PDF/X ID Schema}%
1841     {pdfxid}%
1842     {http://www.npes.org/pdfx/ns/id/}%
1843   \hyxmp@declare@property
1844     {GTS_PDFXVersion}%
1845     {internal}%
1846     {ID of PDF/X standard}%
1847   \hyxmp@end@ext@decl
1848 \fi
1849 }%

```

`\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.

```

1850 \newcommand*{\hyxmp@iptc@extensions}{%
1851   \hyxmp@begin@ext@decl
1852     {IPTC Core Schema}%
1853     {Iptc4xmpCore}%
1854     {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}%
1855   \hyxmp@declare@property
1856     [ContactInfo]
1857     {CreatorContactInfo}
1858     {external}
1859     {Document creator's contact information}

```

We can't call `\hyxmp@end@ext@decl` because we need first need to define the `lptc4xmpCore:ContactInfo` structure.

```

1860   \hyxmp@add@to+xml{%
1861     _____</rdf:Seq>^^J%

```

```

1862 -----</pdfaSchema:property>^^J%
1863 -----<pdfaSchema:valueType>^^J%
1864 -----<rdf:Seq>^^J%
1865 -----<rdf:li rdf:parseType="Resource">^^J%
1866 -----<pdfaType:type>ContactInfo</pdfaType:type>^^J%
1867 -----<pdfaType:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaTy
1868 -----<pdfaType:prefix>Iptc4xmpCore</pdfaType:prefix>^^J%
1869 -----<pdfaType:description>%
1870         Basic set of information to get in contact with a person%
1871 -----</pdfaType:description>^^J%
1872 -----<pdfaType:field>^^J%
1873 -----<rdf:Seq>^^J%
1874   }%
1875   \hyxmp@declare@field
1876     {CiAdrCity}%
1877     {Contact information city}%
1878   \hyxmp@declare@field
1879     {CiAdrCtry}%
1880     {Contact information country}%
1881   \hyxmp@declare@field
1882     {CiAdrExtadr}%
1883     {Contact information address}%
1884   \hyxmp@declare@field
1885     {CiAdrPcode}%
1886     {Contact information local postal code}%
1887   \hyxmp@declare@field
1888     {CiAdrRegion}%
1889     {Contact information regional information such as state or province}%
1890   \hyxmp@declare@field
1891     {CiEmailWork}%
1892     {Contact information email address(es)}%
1893   \hyxmp@declare@field
1894     {CiTelWork}%
1895     {Contact information telephone number(s)}%
1896   \hyxmp@declare@field
1897     {CiUrlWork}%
1898     {Contact information Web URL(s)}%
1899   \hyxmp@add@to+xml{%
1900 -----</rdf:Seq>^^J%
1901 -----</pdfaType:field>^^J%
1902 -----</rdf:li>^^J%
1903 -----</rdf:Seq>^^J%
1904 -----</pdfaSchema:valueType>^^J%
1905 -----</rdf:li>^^J%
1906   }%
1907 }

```

\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.

```

1908 \newcommand*{\hyxmp@prism@extensions}{%
1909   \hyxmp@begin@ext@decl
1910     {PRISM Basic Metadata}%

```

```

1911     {prism}%
1912     {http://prismstandard.org/namespaces/basic/3.0/}%
1913 \hyxmp@declare@property
1914     {complianceProfile}%
1915     {internal}%
1916     {PRISM specification compliance profile to which this document adheres}%
1917 \hyxmp@declare@property
1918     {publicationName}%
1919     {external}%
1920     {Publication name}%
1921 \hyxmp@declare@property
1922     {aggregationType}%
1923     {external}%
1924     {Publication type}%
1925 \hyxmp@declare@property
1926     {bookEdition}%
1927     {external}%
1928     {Edition of the book in which the document was published}%
1929 \hyxmp@declare@property
1930     {volume}%
1931     {external}%
1932     {Publication volume number}%
1933 \hyxmp@declare@property
1934     {number}%
1935     {external}%
1936     {Publication issue number within a volume}%
1937 \hyxmp@declare@property
1938     {pageRange}%
1939     {external}%
1940     {Page range for the document within the print version of its publication}%
1941 \hyxmp@declare@property
1942     {issn}%
1943     {external}%
1944     {ISSN for the printed publication in which the document was published}%
1945 \hyxmp@declare@property
1946     {eIssn}%
1947     {external}%
1948     {ISSN for the electronic publication in which the document was published}%
1949 \hyxmp@declare@property
1950     {isbn}%
1951     {external}%
1952     {ISBN for the publication in which the document was published}%
1953 \hyxmp@declare@property
1954     {doi}%
1955     {external}%
1956     {Digital Object Identifier for the document}%
1957 \hyxmp@declare@property
1958     [URL]
1959     {url}%
1960     {external}%
1961     {URL at which the document can be found}%
1962 \hyxmp@declare@property
1963     [Integer]
1964     {byteCount}%

```

```

1965     {internal}%
1966     {Approximate file size in octets}%
1967 \hyxmp@declare@property
1968     [Integer]
1969     {pageCount}%
1970     {internal}%
1971     {Number of pages in the print version of the document}%
1972 \hyxmp@declare@property
1973     {subtitle}%
1974     {external}%
1975     {Document's subtitle}%
1976 \hyxmp@end@ext@decl
1977 }%

```

`\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.

```

1978 \newcommand*{\hyxmp@jav@extensions}{%
1979 \hyxmp@begin@ext@decl
1980     {NISO/ALPSP Journal Article Versions}%
1981     {jav}%
1982     {http://www.niso.org/schemas/jav/1.0/}%
1983 \hyxmp@declare@property
1984     [Closed Choice of Text]%
1985     {journal_article_version}%
1986     {external}%
1987     {Article status, one of "AO", "SMUR", "AM", "P", "VoR", "CVoR", or "EVoR"}%
1988 \hyxmp@end@ext@decl
1989 }%

```

`\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.

```

1990 \newcommand*{\hyxmp@declare@extensions}{%
1991 \hyxmp@begin@extension@decls

```

Declare the Adobe PDF schema (always present).

```
1992 \hyxmp@pdf@extensions
```

Declare the XMP Media Management extensions (always present).

```
1993 \hyxmp@mm@extensions
```

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

```

1994 \ifHy@pdfa
1995 \hyxmp@pdfa@id@extensions
1996 \fi

```

Conditionally declare the PDF/UA Universal Accessibility extensions.

```

1997 \ifx\@pdfuapart\@empty
1998 \else
1999 \hyxmp@pdfua@id@extensions
2000 \fi

```


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

```
2001 \ifx\pdfxversion\@empty
2002 \else
2003   \hyxmp@pdfx@id@extensions
2004 \fi
```

Conditionally declare IPTC photo metadata extensions.

```
2005 \ifx\hyxmp@iptc@data\@empty
2006 \else
2007   \hyxmp@iptc@extensions
2008 \fi
```

Conditionally declare PRISM basic metadata extensions.

```
2009 \ifx\hyxmp@prism@data\@empty
2010 \else
2011   \hyxmp@prism@extensions
2012 \fi
```

Conditionally declare JAV metadata.

```
2013 \ifx\hyxmp@jav@data\@empty
2014 \else
2015   \hyxmp@jav@extensions
2016 \fi
```

```
2017 \hyxmp@end@extension@decls
2018 }
```

3.6.14 Combining schemata into an XMP packet

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

```
2019 \begingroup
2020 \ifhyxmp@unicodetex
2021   \lccode\!="FEFF %
2022   \lowercase{%
2023     \gdef\hyxmp@bom{!}
2024   }%
2025 \else
2026   \catcode\^^ef=12
2027   \catcode\^^bb=12
2028   \catcode\^^bf=12
2029   \gdef\hyxmp@bom{\^^ef\^^bb\^^bf}%
2030 \fi
2031 \endgroup
```

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

```
2032 \def\hyxmp@construct@packet{%
2033   \gdef\hyxmp+xml{}%
2034   \hyxmp@add@to+xml{<?xpacket begin="\hyxmp@bom" %
2035     id="W5M0MpCehiHzreSzNTczkc9d"?>^^J%
2036     <x:xmpmeta xmlns:x="adobe:ns:meta/">^^J%
2037     __<rdf:RDF %
2038     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\hyxmp@hash">^^J%
2039     ____<rdf:Description rdf:about="">^^J%
```

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

```

2040 -----xmlns:pdf="http://ns.adobe.com/pdf/1.3/"^^J%
2041 -----xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/"^^J%
2042 -----xmlns:dc="http://purl.org/dc/elements/1.1/"^^J%
2043 -----xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/"^^J%
2044 -----xmlns:xmp="http://ns.adobe.com/xap/1.0/"^^J%
2045 -----xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"^^J%
2046 -----xmlns:stEvt="http://ns.adobe.com/xap/1.0/sType/ResourceEvent\hyxmp@hash"^^J%
2047 -----xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/"^^J%
2048 -----xmlns:pdfuaid="http://www.aiim.org/pdfua/ns/id/"^^J%
2049 -----xmlns:pdfx="http://ns.adobe.com/pdfx/1.3/"^^J%
2050 }%

```

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

```

2051 \ifnum0\hyxmp@pdfx@major>3
2052   \hyxmp@add@to@xml{%
2053 -----xmlns:pdfxid="http://www.npes.org/pdfx/ns/id/"^^J%
2054   }%
2055 \fi

```

Revert to “include every namespace” mode.

```

2056 \hyxmp@add@to@xml{%
2057 -----xmlns:prism="http://prismstandard.org/namespaces/basic/3.0/"^^J%
2058 -----xmlns:jav="http://www.niso.org/schemas/jav/1.0/"^^J%
2059 -----xmlns:xmpTPg="http://ns.adobe.com/xap/1.0/t/pg/"^^J%
2060 -----xmlns:stFnt="http://ns.adobe.com/xap/1.0/sType/Font\hyxmp@hash"^^J%
2061 -----xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"^^J%
2062 -----xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"^^J%
2063 -----xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema\hyxmp@hash"^^J%
2064 -----xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property\hyxmp@hash"^^J%
2065 -----xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type\hyxmp@hash"^^J%
2066 -----xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field\hyxmp@hash">^^J%
2067 }%

```

Declare non-standard schemata.

```

2068 \hyxmp@check@iptc@data
2069 \hyxmp@check@prism@data
2070 \hyxmp@check@jav@data
2071 \hyxmp@declare@extensions

```

Insert all the metadata we know how to insert.

```

2072 \hyxmp@pdf@schema
2073 \hyxmp@xmpRights@schema
2074 \hyxmp@dc@schema
2075 \hyxmp@photoshop@schema
2076 \hyxmp@xmp@basic@schema
2077 \hyxmp@pdfa@id@schema
2078 \hyxmp@pdfua@id@schema
2079 \hyxmp@pdfx@id@schema
2080 \hyxmp@mm@schema

```

```

2081 \hyxmp@iptc@schema
2082 \hyxmp@prism@schema
2083 \hyxmp@jav@schema
2084 \hyxmp@xmptpg@schema
2085 \hyxmp@add@to+xml{%
2086 ____</rdf:Description>^^J%
2087 __</rdf:RDF>^^J%
2088 </x:xmpmeta>^^J%
2089 \hyxmp@padding
2090 <?xpacket end="w"?>^^J%
2091 }%
2092 }

```

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.

```

2093 \newcommand*{\hyxmp@embed@packet}{%
2094 \hyxmp@construct@packet
2095 \def\hyxmp@driver{hpdfTEX}%
2096 \ifx\hyxmp@driver\Hy@driver
2097 \hyxmp@embed@packet@pdfTEX
2098 \else
2099 \def\hyxmp@driver{hLUAteX}%
2100 \ifx\hyxmp@driver\Hy@driver
2101 \hyxmp@embed@packet@LUAteX
2102 \else
2103 \def\hyxmp@driver{hdvipdfm}%
2104 \ifx\hyxmp@driver\Hy@driver
2105 \hyxmp@embed@packet@dvipdfm
2106 \else
2107 \def\hyxmp@driver{hxetex}%
2108 \ifx\hyxmp@driver\Hy@driver
2109 \hyxmp@embed@packet@xetex
2110 \else
2111 \@ifundefined{pdfmark}{%
2112 \PackageWarningNoLine{hyperxmp}{%
2113 Unrecognized hyperref driver ‘\Hy@driver’.\MessageBreak
2114 \hyxmp@jobname.tex’s XMP metadata will *not* be\MessageBreak
2115 embedded in the resulting file}%
2116 }{%
2117 \hyxmp@embed@packet@pdfmark
2118 }%
2119 \fi
2120 \fi
2121 \fi
2122 \fi
2123 }

```

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.

```
2124 \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.

```
2125 \newcommand*{\hyxmp@embed@packet@pdftex}{%
2126   \bgroup
2127     \ifluatex
2128     \else
2129       \pdfcompresslevel=0
2130     \fi
2131     \immediate\pdfobj \ifluatex uncompressed\fi stream attr {%
2132       /Type /Metadata
2133       /Subtype /XML
2134     }{\hyxmp@xml}%
2135     \pdfcatalog {/Metadata \the\pdflastobj\space 0 R}%
2136   \egroup
2137 }
```

3.7.2 Embedding using LuaTeX 0.85+

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

```
2138 \newcommand*{\hyxmp@embed@packet@luatex}{%
2139   \immediate\pdfextension obj uncompressed stream attr {%
2140     /Type /Metadata
2141     /Subtype /XML
2142   }{\hyxmp@xml}%
2143   \pdfextension catalog {/Metadata \the\numexpr\pdffeedback lastobj\relax\space 0 R}%
2144 }
```

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.

```
2145 \newcommand*{\hyxmp@embed@packet@pdfmark}{%
2146   \pdfmark{%
```

```

2147     pdfmark=/NamespacePush
2148 }%
2149 \pdfmark{%
2150     pdfmark=/OBJ,
2151     Raw={/_objdef \string{hyxmp@Metadata\string} /type /stream}%
2152 }%
2153 \pdfmark{%
2154     pdfmark=/PUT,
2155     Raw={\string{hyxmp@Metadata\string}
2156         2 dict begin
2157             /Type /Metadata def
2158             /Subtype /XML def
2159             currentdict
2160         end
2161     }%
2162 }%
2163 \pdfmark{%
2164     pdfmark=/PUT,
2165     Raw={\string{hyxmp@Metadata\string} (\hyxmp@xml)}%
2166 }%
2167 \pdfmark{%
2168     pdfmark=/Metadata,
2169     Raw={\string{Catalog\string} \string{hyxmp@Metadata\string}}%
2170 }%
2171 \pdfmark{%
2172     pdfmark=/NamespacePop
2173 }%
2174 }

```

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.

```

2175 \newcommand*{\hyxmp@embed@packet@dvipdfm}{%
2176     \hyxmp@string@len{\hyxmp@xml}%
2177     \special{pdf: object @hyxmp@Metadata
2178         <<
2179             /Type /Metadata
2180             /Subtype /XML
2181             /Length \the\@hyxmp@count
2182         >>
2183         stream^^J\hyxmp@xml endstream%
2184     }%
2185     \special{pdf: docview
2186         <<
2187             /Metadata @hyxmp@Metadata
2188         >>
2189     }%
2190 }

```

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

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

```
2191 \newcommand*\hyxmp@string@len}[1]{%
2192   \@hyxmp@count=0
2193   \expandafter\hyxmp@count@spaces#1 {} %
2194   \expandafter\hyxmp@count@non@spaces#1{}%
2195 }
```

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

```
2196 \def\hyxmp@count@spaces#1 {%
2197   \def\hyxmp@one@token{#1}%
2198   \ifx\hyxmp@one@token\@empty
2199     \advance\@hyxmp@count by -1
2200   \else
2201     \advance\@hyxmp@count by 1
2202     \expandafter\hyxmp@count@spaces
2203   \fi
2204 }
```

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

```
2205 \newcommand*\hyxmp@count@non@spaces}[1]{%
2206   \def\hyxmp@one@token{#1}%
2207   \ifx\hyxmp@one@token\@empty
2208   \else
2209     \advance\@hyxmp@count by 1
2210     \expandafter\hyxmp@count@non@spaces
2211   \fi
2212 }
```

3.7.5 Embedding using $X_{\text{q}}\TeX$

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

```
2213 \newcommand*\hyxmp@embed@packet@xetex}{%
2214   \special{pdf:stream @hyxmp@Metadata (\hyxmp+xml)
2215     <<
2216       /Type /Metadata
2217       /Subtype /XML
2218     >>
2219   }%
2220   \special{pdf:put @catalog
2221     <<
2222       /Metadata @hyxmp@Metadata
2223     >>
2224   }%
2225 }
```

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.

```
2226 \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.

```
2227 \catcode'\="=\hyxmp@dq@code
```

4 Help Wanted

Comma handling Ideally, `\xmpquote` should automatically replace all commas with `\xmpcomma`. Unfortunately, my `TeX` skills are insufficient to pull that off. If you know a way to make `\xmpquote{Hello, world}` work with both Unicode and non-Unicode encodings and with all `TeX` engines (`pdfTeX`, `LuaTeX`, `XyTeX`, 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 9–11. For clarity, metadata values, either specified explicitly by the document or introduced automatically by `hyperxmp`, are colored blue.

```
<?xpacket begin="\357\273\277" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/">
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    <rdf:Description rdf:about=""
      xmlns:pdf="http://ns.adobe.com/pdf/1.3/"
      xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/"
      xmlns:xmp="http://ns.adobe.com/xap/1.0/"
      xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"
      xmlns:stEvt="http://ns.adobe.com/xap/1.0/sType/ResourceEvent#"
      xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/"
      xmlns:pdfuaid="http://www.aiim.org/pdfua/ns/id/"
      xmlns:pdfx="http://ns.adobe.com/pdfx/1.3/"
      xmlns:prism="http://prismstandard.org/namespaces/basic/3.0/"
      xmlns:jav="http://www.niso.org/schemas/jav/1.0/"
      xmlns:xmpTPg="http://ns.adobe.com/xap/1.0/t/pg/"
      xmlns:stFnt="http://ns.adobe.com/xap/1.0/sType/Font#"
      xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"
      xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"
      xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema#"
      xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property#"
      xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type#"
      xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field#">
```

```

<pdfaExtension:schemas>
  <rdf:Bag>
    :
    [over 200 lines of boilerplate definitions not shown]
    :
  </rdf:Bag>
</pdfaExtension:schemas>
<pdf:Keywords>
  energy quanta, Hertz effect, quantum physics
</pdf:Keywords>
<pdf:Producer>
  LuaHBTeX, Version 1.12.0 (TeX Live 2020)
</pdf:Producer>
<pdf:PDFVersion>1.5</pdf:PDFVersion>
<xmpRights:Marked>True</xmpRights:Marked>
<xmpRights:WebStatement>
  http://creativecommons.org/licenses/by-nc-nd/3.0/
</xmpRights:WebStatement>
<dc:format>application/pdf</dc:format>
<dc:title>
  <rdf:Alt>
    <rdf:li xml:lang="x-default">
      On a heuristic viewpoint concerning the production
      and transformation of light
    </rdf:li>
    <rdf:li xml:lang="en">
      On a heuristic viewpoint concerning the production
      and transformation of light
    </rdf:li>
    <rdf:li xml:lang="de">
      Über einen die Erzeugung und Verwandlung des Lichtes
      betreffenden heuristischen Gesichtspunkt
    </rdf:li>
  </rdf:Alt>
</dc:title>
<dc:description>
  <rdf:Alt>
    <rdf:li xml:lang="en">photoelectric effect</rdf:li>
  </rdf:Alt>
</dc:description>
<dc:rights>
  <rdf:Alt>
    <rdf:li xml:lang="en">Copyright (C) 1905, Albert Einstein</rdf:li>
  </rdf:Alt>
</dc:rights>
<dc:publisher>
  <rdf:Bag>
    <rdf:li>Wiley-VCH</rdf:li>
  </rdf:Bag>
</dc:publisher>
<dc:creator>

```



```

<rdf:Seq>
  <rdf:li>Albert Einstein</rdf:li>
</rdf:Seq>
</dc:creator>
<dc:subject>
  <rdf:Bag>
    <rdf:li>energy quanta</rdf:li>
    <rdf:li>Hertz effect</rdf:li>
    <rdf:li>quantum physics</rdf:li>
  </rdf:Bag>
</dc:subject>
<dc:date>
  <rdf:Seq>
    <rdf:li>1905-03-17</rdf:li>
  </rdf:Seq>
</dc:date>
<dc:language>
  <rdf:Bag>
    <rdf:li>en</rdf:li>
  </rdf:Bag>
</dc:language>
<dc:type>
  <rdf:Bag>
    <rdf:li>Text</rdf:li>
  </rdf:Bag>
</dc:type>
<dc:source>einstein.tex</dc:source>
<dc:identifier>info:lccn/50013519</dc:identifier>
<photoshop:AuthorsPosition>
  Technical Assistant, Level III
</photoshop:AuthorsPosition>
<photoshop:CaptionWriter>Scott Pakin</photoshop:CaptionWriter>
<xmp:CreateDate>2020-07-25T21:37:02-06:00</xmp:CreateDate>
<xmp:ModifyDate>2020-07-25T21:37:02-06:00</xmp:ModifyDate>
<xmp:MetadataDate>2020-07-25T21:37:02-06:00</xmp:MetadataDate>
<xmp:CreatorTool>LaTeX with hyperref package</xmp:CreatorTool>
<xmpMM:DocumentID>
  uuid:6d1ac9ec-4ff2-515a-954b-648eeb4853b0
</xmpMM:DocumentID>
<xmpMM:InstanceID>
  uuid:3e4c4182-b182-46c9-995f-754c41d13390
</xmpMM:InstanceID>
<xmpMM:VersionID>2.998e8</xmpMM:VersionID>
<xmpMM:RenditionClass>default</xmpMM:RenditionClass>
<Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">
  <Iptc4xmpCore:CiAdrExtadr>Kramgasse 49</Iptc4xmpCore:CiAdrExtadr>
  <Iptc4xmpCore:CiAdrCity>Bern</Iptc4xmpCore:CiAdrCity>
  <Iptc4xmpCore:CiAdrPcode>3011</Iptc4xmpCore:CiAdrPcode>
  <Iptc4xmpCore:CiAdrCtry>Switzerland</Iptc4xmpCore:CiAdrCtry>
  <Iptc4xmpCore:CiTelWork>031 312 00 91</Iptc4xmpCore:CiTelWork>
  <Iptc4xmpCore:CiEmailWork>aeinstein@ipi.ch</Iptc4xmpCore:CiEmailWork>
  <Iptc4xmpCore:CiUrlWork>
    http://einstein.biz/,
    https://www.facebook.com/AlbertEinstein

```

```

</Iptc4xmpCore: CiUrlWork>
</Iptc4xmpCore: CreatorContactInfo>
<prism: complianceProfile>three</prism: complianceProfile>
<prism: subtitle xml: lang="en-US">
  Putting that bum Maxwell in his place
</prism: subtitle>
<prism: publicationName xml: lang="de">
  Annalen der Physik
</prism: publicationName>
<prism: aggregationType>journal</prism: aggregationType>
<prism: volume>322</prism: volume>
<prism: number>6</prism: number>
<prism: pageRange>132-148</prism: pageRange>
<prism: issn>0003-3804</prism: issn>
<prism: eIssn>1521-3889</prism: eIssn>
<prism: doi>10.1002/andp.19053220607</prism: doi>
<prism: url>
  http://www.physik.uni-augsburg.de/annalen/history/einstein-papers/190517132-148.pdf
</prism: url>
<prism: byteCount>41197</prism: byteCount>
<prism: pageCount>1</prism: pageCount>
<jav: journal_article_version>VoR</jav: journal_article_version>
<xmpTPg: Fonts>
  <rdf: Bag>
    <rdf: li rdf: parseType="Resource">
      <stFnt: fontFace>LMRoman10-Regular</stFnt: fontFace>
      <stFnt: fontFamily>LM Roman 10</stFnt: fontFamily>
      <stFnt: fontName>LMRoman10-Regular</stFnt: fontName>
      <stFnt: versionString>
        2.004;PS 2.004;hotconv 1.0.49;makeotf.lib2.0.14853
      </stFnt: versionString>
      <stFnt: fontFileName>lmroman10-regular.otf</stFnt: fontFileName>
      <stFnt: fontType>opentype</stFnt: fontType>
    </rdf: li>
    <rdf: li rdf: parseType="Resource">
      <stFnt: fontFace>LMRoman17-Regular</stFnt: fontFace>
      <stFnt: fontFamily>LM Roman 17</stFnt: fontFamily>
      <stFnt: fontName>LMRoman17-Regular</stFnt: fontName>
      <stFnt: versionString>
        2.004;PS 2.004;hotconv 1.0.49;makeotf.lib2.0.14853
      </stFnt: versionString>
      <stFnt: fontFileName>lmroman17-regular.otf</stFnt: fontFileName>
      <stFnt: fontType>opentype</stFnt: fontType>
    </rdf: li>
    <rdf: li rdf: parseType="Resource">
      <stFnt: fontFace>LMRoman12-Regular</stFnt: fontFace>
      <stFnt: fontFamily>LM Roman 12</stFnt: fontFamily>
      <stFnt: fontName>LMRoman12-Regular</stFnt: fontName>
      <stFnt: versionString>
        2.004;PS 2.004;hotconv 1.0.49;makeotf.lib2.0.14853
      </stFnt: versionString>
      <stFnt: fontFileName>lmroman12-regular.otf</stFnt: fontFileName>
      <stFnt: fontType>opentype</stFnt: fontType>
    </rdf: li>
  </rdf: Bag>

```

```

<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://www.images.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://www.images.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.primstandard.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.primstandard.org/specifications/3.0/PRISM_CV_Spec_3.0.pdf.
- [10] International Press Telecommunications Council. *IPTC Photo Metadata: Core 1.1/Extension 1.1*, July 2010. Revision 1. Available from http://www.iptc.org/std/photometadata/specification/IPTC-PhotoMetadata-201007_1.pdf.
- [11] Internet Assigned Numbers Authority. Language subtag registry, January 11, 2011. Available from <http://www.iana.org/assignments/language-subtag-registry>.
- [12] Paul J. Leach, Michael Mealling, and Rich Salz. A Universally Unique IDentifier (UUID) URN namespace. Request for Comments 4122, Internet Engineering Task Force, Network Working Group, July 2005. Category: Standards Track. Available from <http://www.ietf.org/rfc/rfc4122.txt>.
- [13] PDF/A Competence Center, Berlin, Germany. *TechNote 0008: Pre-defined XMP Properties in PDF/A-1*, March 20, 2008. Available from http://www.pdfa.org/wp-content/uploads/2011/08/tn0008_predefined_xmp_properties_in_pdfa-1_2008-03-20.pdf.
- [14] PDF/A Competence Center, Berlin, Germany. *TechNote 0009: XMP Extension Schemas in PDF/A-1*, March 20, 2008. Available from http://www.pdfa.org/wp-content/uploads/2011/08/tn0009_xmp_extension_schemas_in_pdfa-1_2008-03-20.pdf.

- [15] Misha Wolf and Charles Wicksteed. Date and time formats. Note NOTE-datettime, World Wide Web Consortium (W3C), September 15, 1997. Available from <http://www.w3.org/TR/NOTE-datettime>.

Change History

v1.0	General: Initial version	1	Heiko Oberdiek’s major rewrite of the code to better support native-Unicode T _E X implementations (X _E T _E X and LuaT _E X)	1
v1.1	<code>\hyxmp@construct@packet</code> : Explicitly set the category codes of characters $\langle EF \rangle$, $\langle BB \rangle$, and $\langle BF \rangle$ to “letter”. Thanks to Daniel Schömer for the bug report	81	New <code>\AtBeginDocument</code> code from Heiko Oberdiek to properly encode <code>\@pdfmetalang</code>	33
v1.2	General: Added support for the X _E T _E X backend (<code>xdvipdfmx</code>)	1	<code>\hyxmp@add@simple</code> : Added this macro	54
	Added support for the Photoshop schema	1	<code>\hyxmp@add@toxml</code> : Updated also to replace commas	60
	Made the package compatible with <code>ngerman</code> . Thanks to Tobias Mueller for the bug report.	18	<code>\hyxmp@bom</code> : Added by Heiko Oberdiek	81
v1.3	General: Introduced the <code>pdfmetalang</code> package option, which enables an author to specify the language in which he wrote the document’s metadata	33	<code>\hyxmp@comma</code> : Added this macro	42
			<code>\hyxmp@construct@packet</code> : Modified by Heiko Oberdiek to use an appropriate BOM representation via <code>\hyxmp@bom</code>	81
v1.4	<code>\hyxmp@mm@schema</code> : Renamed the <code>xapMM</code> namespace prefix to <code>xmpMM</code>	67	<code>\hyxmp@crap@convert</code> : Added by Heiko Oberdiek	53
	<code>\hyxmp@rdf@dc</code> : Included metadata in the x-default language regardless of the specified metadata language	63	<code>\hyxmp@crap@test</code> : Added by Heiko Oberdiek	53
	<code>\hyxmp@xmpRights@schema</code> : Renamed the <code>xapRights</code> namespace prefix to <code>xmpRights</code>	66	<code>\hyxmp@dc@schema</code> : Added support for <code>dc:language</code> and <code>dc:source</code>	66
v1.5	General: Made the XMP inclusion more robust. Thanks to Heiko Oberdiek for the bug report and suggested modifications.	18	<code>\hyxmp@is@unicode</code> : Added by Heiko Oberdiek	50
v2.0	General: Added support for the XMP Basic schema and miscellaneous other bits of metadata	1	<code>\hyxmp@list@toxml</code> : Modified by Heiko Oberdiek to use the new Unicode-processing macros	64
			<code>\hyxmp@photoshop@schema</code> : Simplified using <code>\hyxmp@add@simple</code>	68
			<code>\hyxmp@ProcessKeyvalOptions</code> : Added this macro	30
			<code>\hyxmp@skiptorelax</code> : Added by Heiko Oberdiek	53
			<code>\hyxmp@skipzeros</code> : Added by Heiko Oberdiek	52
			<code>\hyxmp@SpaceOther</code> : Added by Heiko Oberdiek	52
			<code>\hyxmp@toxml</code> : Added by Heiko Oberdiek	50

Escaped parentheses written with pdfmarks to prevent dvips from line-wrapping the XMP packet	51	v2.3	<code>\hyxmp@iptc@extensions</code> : Gave the
<code>\hyxmp@toxml@unicodetex</code> : Added by Heiko Oberdiek	51		<code>lptc4xmpCore:CreatorContactInfo</code> fields a unique pdfaType:prefix to better support conversion of the document to PDF/A
<code>\hyxmp@xetex@crap</code> : Added by Heiko Oberdiek	52	v2.3a	<code>\hyxmp@detect@langs</code> : Bug fix: Redefine <code>\@pdflang</code> as <code>\@empty</code> when hyperref has set it to <code>\relax</code>
<code>\hyxmp@xmllify</code> : Completely rewritten by Heiko Oberdiek to better support Unicode-enabled T _E X programs	49	v2.3b	<code>\XMPTruncateList</code> : Made all definitions local to avoid spurious Too many unprocessed floats errors when running with memoir
<code>\hyxmp@xmp@basic@schema</code> : Added this macro	68	v2.4	General: Added support for the PDF/A Identification schema, as requested by Florian Breitwieser
<code>\hyxmp@xmpRights@schema</code> : Modified to include <code>xmpRights:Marked</code> only when <code>pdfcopyright</code> is specified and <code>xmpRights:WebStatement</code> only when <code>pdflicenseurl</code> is specified	66		<code>\hyxmp@add@simple@var</code> : Added this macro
<code>\hyxmp@zero</code> : Added by Heiko Oberdiek	54		<code>\hyxmp@create@uuid</code> : Modified this macro to produce a proper version 4 (random or pseudorandom) UUID
<code>\ifhyxmp@unicodetex</code> : Added by Heiko Oberdiek	49		<code>\hyxmp@dc@schema</code> : Made <code>dc:language</code> a Bag instead of an individual item so as to conform to the latest XMP specifications, a detail identified by Florian Breitwieser
<code>\ProcessKeyvalOptions</code> : Added this macro	30		<code>\hyxmp@parse@time</code> : Added this macro
<code>\xmpcomma</code> : Added this macro	42		<code>\hyxmp@parse@tz</code> : Added this macro
<code>\xmpquote</code> : Added this macro	42		<code>\hyxmp@parse@tz@char</code> : Added this macro
<code>\XMPTruncateList</code> : Added this macro	43		<code>\hyxmp@pdf@schema</code> : Made <code>\hyxmp@pdf@schema</code> <i>always</i> include the Adobe PDF schema, even when empty. Florian Breitwieser noted that this is necessary for PDF/A-1b compliance
v2.1			<code>\hyxmp@pdf@to@xmp@date</code> : Added this macro
General: Enabled hyperxmp and hyperref to be loaded in either order. This addresses a bug report by Yury Donskoy	28		<code>\hyxmp@pdfa@id@schema</code> : Added this macro
<code>\hypersetup</code> : Added this macro	30		<code>\hyxmp@today@xmp</code> : Modified the code to parse the time and
<code>\hyxmp@hypersetup</code> : Added this macro	30		
<code>\hyxmp@redefine@Hyp</code> : Added this macro	28		
v2.2			
General: Added support for the IPTC Photo Metadata schema	1		
<code>\hyxmp@iptc@extensions</code> : Added this macro to support PDF/A generation	77		
<code>\hyxmp@iptc@schema</code> : Added this macro	70		
<code>\hyxmp@list@to@lines</code> : Added this macro	69		
<code>\xmpcomma</code> : Changed the default from <code>\relax</code> to an ordinary comma	42		
<code>\xmplinesep</code> : Added this macro	69		

timezone from		that Acrobat’s PDF/A validator	
<code>\pdfcreationdate</code> , as proposed		seems to prefer <code>lptc4xmpCore</code>	70
by Florian Breitwieser	47	<code>\hyxmp@pdfa@id@schema</code> : Let the	
<code>\hyxmp@today@xmp@define</code> : Added		author specify the PDF/A part	
this macro	46	and conformance IDs, as	
<code>\hyxmp@xmp@to@pdf@date</code> : Added		requested by Leonid Sinev	69
this macro	44	v3.0	
<code>\xmp tilde</code> : Added this macro	43	General: Made the code compatible	
v2.5		with LuaTeX 0.85. Thanks to	
General: Enabled “_” to work		Robert Schlicht, Leonid Sinev,	
within email addresses, as		and David Carlisle for bug	
requested by Leonid Sinev	1	reports and to Leonid Sinev for	
<code>\hyxmp@add@to+xml</code> : Updated also		helping test the new hyperxmp	
to replace underscores and to		code	1
modify only the text being		<code>\hyxmp@embed@packet@luatex</code> :	
added, not the already-modified		Added this macro	84
text	60	<code>\hyxmp@today@xmp@define</code> :	
<code>\hyxmp@textunderscore</code> : Added		Modified to accept the name of	
this macro	20	a macro to define	46
<code>\hyxmp@uscore</code> : Added this macro	42	<code>\hyxmp@xmp@basic@schema</code> : Made	
v2.6		the XMP <code>xmp:CreateDate</code> ,	
General: Added support for a new		<code>xmp:ModifyDate</code> , and	
<code>pdfdate</code> key to explicitly specify		<code>xmp:MetadataDate</code> match the	
the document date (and		PDF <code>CreateDate</code>	68
optionally time)	1	v3.1	
v2.7		<code>\hyxmp@embed@packet@luatex</code> :	
<code>\hyxmp@auto@assign@data</code> :		Updated to use <code>\pdfextension</code>	
Automatically use <code>\title</code> and		<code>obj uncompress</code> as suggested	
<code>\author</code> if <code>pdftitle</code> and		by Hans Hagen	84
<code>pdfauthor</code> are left unspecified.		<code>\hyxmp@embed@packet@pdftex</code> :	
Thanks to Maciej Radziejewski		Leave the XMP packet—and	
for the suggestion	35	only the XMP	
v2.8		packet—uncompressed in both	
<code>\hyxmp@add@to+xml</code> : Corrected		<code>pdfTeX</code> and pre-0.85 LuaTeX	84
inadvertent lowercasing of		v3.2	
non-Latin characters when run		<code>\hyxmp@as@pdf@date</code> : Added this	
under XeLaTeX or LuaLaTeX		macro	44
(bug reported by Leonid Sinev)	60	<code>\hyxmp@as@xmp@date</code> : Added this	
v2.9		macro	43
General: Force inclusion of		<code>\hyxmp@today@xmp@define</code> :	
<code>dc:creator</code> , <code>dc:title</code> , and		Modified to include hours and	
<code>dc:description</code> —even if		minutes	46
empty—when <code>hyperref</code> is loaded		<code>\hyxmp@xmp@basic@schema</code> : Honor	
with the <code>pdfa</code> option (suggested		<code>hyperref</code> ’s <code>pdfcreationdate</code> and	
by Leonid Sinev)	1	<code>pdfmoddate</code> options plus a new	
Introduced the <code>pdftype</code> package		<code>pdfmetadate</code> option. Leonid	
option, which enables an author		Sinev requested this additional	
to specify the type of document		control and helped test the	
being produced	1	resulting hyperxmp code	68
<code>\hyxmp@iptc@schema</code> : Use		v3.3	
<code>lptc4xmpCore</code> instead of		General: Don’t overwrite an	
<code>lptc4ContInfo</code> as the		existing <code>pdfmetalang</code> with	
contact-information metadata		<code>pdflang</code> or <code>x-default</code> . This	
prefix. Leonid Sinev reports			

addresses a bug report by Niklas Beisert	33	<code>\hyxmp@declare@property</code> : Added this macro	75
<code>\@pdfsource</code> : Added this macro and the corresponding <code>pdfsource</code> option, at Niklas Beisert’s request	23	<code>\hyxmp@end@ext@decl</code> : Added this macro	74
<code>\hyxmp@rdf@dc</code> : Bug fix: Output the metadata language as correct XML even when <code>hyperref</code> is loaded with the <code>unicode</code> option	63	<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>	77
<code>\XMPLangAlt</code> : Added this macro based on a request—and some code—by Niklas Beisert to support metadata expressed in multiple languages	56	Rewrote to more closely honor the XMP specification	77
v3.4		<code>\hyxmp@iptc@schema</code> : Moved the definition of <code>\hyxmp@iptc@data</code> from here into <code>\hyxmp@check@iptc@data</code>	70
General: Use <code>ifmtarg</code> to test for empty arguments, including non-empty but all spaces	1	Renamed this macro to <code>\hyxmp@iptc@schema</code> from <code>\hyxmp@photometa@schema</code>	70
<code>\hyxmp@seed@string</code> : Correctly handle an author field of all spaces. Bug reported by Gaëtan Leurent	59	Rewrote this macro entirely to correct the use of fields within a structure	70
v3.5		<code>\hyxmp@mm@extensions</code> : Added this macro	75
<code>\hyxmp@DocumentID</code> : Added the <code>pdfdocumentid</code> option, at Michael Osipov’s request	23	<code>\hyxmp@mm@schema</code> : Include <code>xmpMM:VersionID</code> in the XMP packet	67
<code>\hyxmp@InstanceID</code> : Added the <code>pdfinstanceid</code> option, at Michael Osipov’s request	24	<code>\hyxmp@no@info@lists</code> : Added this macro	27
<code>\hyxmp@mm@schema</code> : Generate <code>\hyxmp@DocumentID</code> and <code>\hyxmp@InstanceID</code> only if the document does not already define these using the <code>pdfdocumentid</code> and <code>pdfinstanceid</code> options	67	<code>\hyxmp@pdfa@id@extensions</code> : Added this macro	76
<code>\hyxmp@seed@string</code> : Seed with the <code>TEX</code> timestamp in addition to the document-specified timestamp	59	<code>\hyxmp@prism@extensions</code> : Added this macro	78
v4.0		<code>\hyxmp@prism@schema</code> : Added this macro	71
General: Include all metadata within a single <code>rdf:Description</code> block	1	<code>\XMPTruncateList</code> : Deprecated this macro	43
<code>\hyxmp@add@simple@lang</code> : Added this macro	54	v4.1	
<code>\hyxmp@begin@ext@decl</code> : Added this macro	74	General: Updated the documentation to refer to <code>\pdfnumpages</code> by its correct name. Thanks to Volker RW Schaa for catching the discrepancy	1
<code>\hyxmp@declare@field</code> : Replaced <code>\hyxmp@declare@resource</code> with this macro	75	<code>\hyxmp@aep@toks</code> : Invoke <code>\hyxmp@no@info@lists</code> at the beginning of the document, for compatibility with both newer and older versions of <code>hyperref</code>	32
		<code>\hyxmp@singleton@dc</code> : Added this macro	65
		v5.0	
		General: Added support for PDF/UA standards, as requested	

by Robin Schwab	1	<code>\hyxmp@define@pdfproducer:</code>	Check for Lua \TeX before checking for pdf \TeX to work around luatex85's confusing <code>iftex</code> by defining <code>\pdfTeXversion</code> . Thanks to Robin Schwab for the bug report	61
Added support for PDF/X standards, as requested by Robin Schwab	1	<code>\hyxmp@timestamp:</code>	Don't rely on <code>\jobname.aux</code> existing to query the current time under X \LaTeX . Instead, use <code>\jobname.log</code> . Thanks to Ulrike Fischer for the bug report and for her suggestion to use the log file.	47
Define a default producer	61	v5.2	General: Introduced the <code>pdfidentifier</code> package option, which enables an author to specify a unique identifier for the document	1
Don't set any document dates (creation, modification, or metadata) from <code>pdfdate</code>	1	<code>\hyxmp@add@simple@pfx:</code>	Added this macro	55
<code>\@pdfrendition:</code> Added the <code>pdfrendition</code> option	24	<code>\hyxmp@assign@major@minor:</code>	Added this macro. <code>hyperxmp</code> now correctly specifies <code>pdf:PDFVersion</code> when generating PDF 2.0+. Thanks to Ulrike Fischer for alerting me to PDF 2.0's availability in the \TeX ecosystem and informing me how to activate it	61
<code>\@pdfxstandard:</code> Added this macro	23	<code>\hyxmp@cond@dc@identifier:</code>	Added this macro	65
<code>\hyxmp@add@simple:</code> Insert the tag name (#1) verbatim	54	<code>\ifdraft:</code>	Define <code>\ifdraft</code> only locally, at Niklas Beisert's request	24
<code>\hyxmp@check@standards:</code> Added this macro	31	v5.3	<code>\@if@def@and@nonempty:</code> Added this macro	20
<code>\hyxmp@check@std:</code> Added this macro	23	<code>\hyxmp@at@end:</code>	Use <code>\AtEndDocument</code> in all \TeX back ends that provide it. Thanks to Nelson Posse Lago for pointing out why <code>atenddvi</code> is best avoided if possible	19
<code>\hyxmp@declare@property:</code> Insert the property name (#2) verbatim	75	<code>\hyxmp@auto@assign@data:</code>	Consider other author-provided sources of metadata. Thanks to Robin Schwab for proposing that <code>hyperxmp</code> use the KOMA letter classes's metadata	35
<code>\hyxmp@define@pdfproducer:</code> Added this macro	61	<code>\hyxmp@dc@schema:</code>	Include all languages used in the document	
<code>\hyxmp@no@info@lists:</code> Renamed this macros from <code>\hyxmp@suppress@pdf@metadata</code> and rewrote it to replace, if possible, only Author and Keywords	27			
<code>\hyxmp@pdf@extensions:</code> Added this macro	75			
<code>\hyxmp@pdf@schema:</code> Honor <code>pdftrapped</code>	62			
<code>\hyxmp@pdfua@id@extensions:</code> Added this macro	76			
<code>\hyxmp@pdfua@id@schema:</code> Added this macro	69			
<code>\hyxmp@pdfx@id@extensions:</code> Added this macro	77			
<code>\hyxmp@pdfx@id@schema:</code> Added this macro	69			
<code>\hyxmp@today@pdf:</code> Added this macro	48			
<code>\hyxmp@today@xmp:</code> Support X \LaTeX 's <code>\filemoddate</code>	47			
<code>\hyxmp@today@xmp@define:</code> Modified to specify UTC	46			
v5.1				
<code>\hyxmp@banner@to@producer:</code> Prevent the category code of "@" from propagating past the <code>\begin{document}</code> . Thanks to Robert Schlicht for noticing this catcode "leak" and providing a correction	61			

in dc:language	66	Article Versions recommendation [1]	1
\hyxmp@detect@langs: Acquire the default language from the polyglossia package, if loaded. Thanks to Robin Schwab for bringing that package to my attention	40	Move most of the \AtEndPreamble code to \hyxmp@at@end	33
\hyxmp@parse@acmart: Added this macro	38	\hyxmp@aep@toks: Copy \title to pdftitle and \author to pdfauthor at the start of the document to improve consistency between XMP and PDF metadata	33
\hyxmp@set@koma@phones: Added this macro	33	Load hyperref automatically if the document does not do so explicitly, as requested by Robin Schwab	32
\hyxmp@use@first@valid: Added this macro	34	\hyxmp@auto@assign@data: Moved the language-detection and Xe _{La} TeX date-detection code here from the \hyxmp@at@end block	35
v5.4		Moved title and author autodetection to the \AtEndPreamble	35
General: Moved the automatic assignment of \@pdflang and \@pdfmetalang from \hyxmp@auto@assign@data to within a call to \hyxmp@at@end	33	Use Lua _{TeX} mechanisms, when available, to automatically compute the page count	35
\hyxmp@dc@schema: Bug fix: Use \hyxmp@today@xmp as the date only if \@pdfdate is undefined	66	\hyxmp@detect@langs: Set the language(s) immediately instead of deferring them to \hyxmp@set@dc@lang	40
\hyxmp@detect@langs: Added support for babel	41	Store the main language in \@pdflang. Thanks to Javier Bezos for his help with the hyperxmp code and for modifying babel for hyperxmp's benefit	41
Refactored language detection into a separate command	40	\hyxmp@jav@extensions: Added this macro	80
\hyxmp@parse@acmart: Bug fix: Correct a missing “else” argument in two invocations of \@if@def@and@nonempty	39	\hyxmp@jav@schema: Added this macro	71
v5.5		\hyxmp@mm@extensions: Corrected the type of xmpMM:RenditionClass. Thanks to Thorsten Wißmann for the bug report and patch . .	75
General: Automatically assign pdfnumpages and pdfbytes under pdf _{La} TeX and Lua _{La} TeX .	1	\hyxmp@query@self: Added this macro	37
Correctly handle source files with spaces in their name. Thanks to Peter Dyballa for the bug report	19	\hyxmp@rdf@dc: List x-default alternatives before language-specific alternatives, as dictated by the XMP specification [5]	63
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	Rewrite the core part of this macro to divide it into four, cleanly defined cases	63
Introduced the pdfpubstatus package option, which enables an author to specify the document's publication status. Thanks to Robin Schwab for pointing me to the Journal			

	<code>\hyxmp@set@koma@phones</code> : Support hyperlinks and other markup in <code>frommobilephone</code> and <code>fromphone</code> , as requested by Robin Schwab	33			
	<code>\hyxmp@xmptpg@schema</code> : Added this macro	71			
v5.6	General: Don't inadvertently replace underscores in filenames when writing font-related metadata	72			
	Make <code>write_xmp_font_list</code> robust to fonts loaded using <code>HarfBuzz</code> . Thanks to John Lienhard for the bug report . .	72			
	<code>\ifdraft</code> : 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> .	24			
v5.7	General: Do not automatically compute the PDF file size under pdfL ^A T _E X because this confuses <code>latexmk</code> . Thanks to John Collins, Nelson Posse Lago, Derek Dreyer, and the other contributors to acmart issue #413 , "Latexmk goes into an infinite loop even on sample files from ACM"	36			
	<code>\hyxmp@aep@toks</code> : As requested by Moritz Heckscher, define <code>\hyxmp@DocumentID</code> and <code>\hyxmp@InstanceID</code> at the end of the preamble instead of at the end of the document	67			
v5.8	General: Distribute an <code>add_byteCount</code> script and document some sample <code>latexmk</code> configuration code that invokes it. Thanks to John Collins for providing both of those	17			
	Take <code>--output-directory</code> into consideration when querying the output file size. Thanks to John Collins for pointing out that the user can change the output directory	36			
			v5.9	General: At Karl Berry's request, rename <code>add_byteCount</code> to the less generic-sounding <code>hyperxmp-add-bytecount</code> . . .	17
			v5.10	<code>\hyxmp+xml</code> : Include the <code>pdfxid</code> namespace only if the PDF/X version is 4 or greater. Thanks to John Lienhard for the bug report	82
			v5.11	General: Disable <code>hyperxmp</code> if L ^A T _E X3 document metadata is available. Document metadata implies the presence of PDF management, which completely breaks <code>hyperxmp</code>	18
				<code>\@hyxmp@count</code> : Added this macro to fix a bug with <code>pdfapart</code> . Thanks to John H. Lienhard and Kartik Singhal for their bug reports	22
				<code>\hyxmp@at@end</code> : Use <code>\AddToHook</code> when available. This addresses a bug reported on T _E X StackExchange by joHub and solved by Ulrike Fischer	19
				<code>\hyxmp@ProcessKeyvalOptions</code> : Bug fix: Restore <code>\ProcessKeyvalOptions</code> after first use. Thanks to Ulrike Fischer for the bug report . . .	30
				<code>\hyxmp@query@self</code> : Use <code>\thetotalpages</code> to compute the page count in an engine-independent manner. Thanks to Ulrike Fischer for recommending this mechanism	37
			v5.12	General: Require that <code>hyperref</code> be loaded before <code>hyperxmp</code>	28
			v5.13	<code>\ifdraft</code> : Rewrite <code>ifdraft</code> handling to avoid loading a package within a group, which L ^A T _E X soon will stop supporting. Thanks to Ulrike Fischer and Boris Veytsman for bringing this to my attention	24

Index

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

Symbols		
<code>\@acmBooktitle</code>	617	<code>\@pdfdoi</code>
<code>\@acmConference</code>	618	342, 590, 1440,
<code>\@acmDOI</code>	588, 589, 591	1446, 1601, 1686
<code>\@acmISBN</code>	598, 599, 602	<code>\@pdfeissn</code>
<code>\@acmNumber</code>	632	175, 343, 1441,
<code>\@acmVolume</code>	629	1447, 1600, 1687
<code>\@author</code>	414, 544	<code>\@pdfidentifier</code>
<code>\@baseurl</code>	325, 1505	193, 344,
<code>\@elt</code> <u>690</u> , <u>1387</u> , <u>1550</u> , <u>1555</u>		1440, 1444, 1451
<code>\@elt@first</code>	<u>1548</u>	<code>\@pdfisbn</code>
<code>\@elt@rest</code>	<u>1550</u> , <u>1552</u>	177,
<code>\@hyxmp@count</code>	89,	345, 604, 1441,
92, 93, 101, 102,		1449, 1598, 1688
107, 110, 812,		<code>\@pdfissn</code>
813, 814, 815,		173, 346, 1441,
817, 819, 820,		1448, 1599, 1689
821, 822, 824,		<code>\@pdfissuenum</code>
1134, 1137, 1142,		347, 631, 1596, 1690
1150, 1151, 1159,		<code>\@pdfkeywords</code>
1160, 1161, 1163,		238, <u>285</u> , 348
1167, 1169, 1170,		<code>\@pdflang</code>
1172, 1173, 1202,		349, 465, 470,
1204, 1219, 1228,		473, 640, <u>642</u> , 657
1233, 1240, 1241,		<code>\@pdflicenseurl</code>
1242, 1522, 1523,		<u>76</u> , 350, 1456, 1470
1525, 1529, 1837,		<code>\@pdfmetadatetitle</code>
1838, 2181, 2192,		61, 351, 1499, 1502
2199, 2201, 2209		<code>\@pdfmetalang</code> <u>82</u> , 469,
<code>\@if@def@and@nonempty</code>		471, 473, 1088,
.	<u>39</u> , 428,	1335, 1351, 1364
429, 435, 464,		<code>\@pdfmdate</code>
534, 587, 597, 1330		352, 1493, 1497
<code>\@ifclassloaded</code>	635	<code>\@pdfnumpages</code>
<code>\@ifmtarg</code>	37, 1093	171, 353, 536,
<code>\@ifmtargexp</code>		1604, 1613, 1691
37, 41, 375, 392,		<code>\@pdfpagerange</code>
443, 494, 829,		187, 354, 1597, 1692
1319, 1377, 1427,		<code>\@pdfproducer</code>
1444, 1476, 1477,		<u>1282</u> , <u>1289</u> , 1291
1487, 1493, 1499		<code>\@pdfpublication</code> <u>165</u> ,
<code>\@ifnextchar</code>	20, 1088	355, 615, 1592, 1693
<code>\@ifnotmtarg</code>		<code>\@pdfpublisher</code>
38, 85, 1082, 1091		<u>181</u> , 489, 612, 1426
<code>\@ifnotmtargexp</code>		<code>\@pdfpubstatus</code>
37, 391,		197, 1607, 1702
455, 1064, 1105,		<code>\@pdfpubtype</code>
1403, 1416, 1543		167,
<code>\@journalName</code>	616	356, 625, 1593, 1694
<code>\@latex@warning@no@line</code>		<code>\@pdfrendition</code> <u>157</u> , 1484
.	407	
<code>\@pdfaformance</code>	95, 382, 1515	
<code>\@pdfapart</code>	90,	
375, 381, 387, 1514		
<code>\@pdfauthor</code> 232, <u>264</u> ,		
326, 412, 1217, 1225		
<code>\@pdfauthorshorttitle</code>	78, 327, 1508, 1509	
<code>\@pdfbookedition</code>	179, 328, 1594, 1684	
<code>\@pdfbytes</code>	169,	
329, 538, 1603, 1685		
<code>\@pdfcaptionwriter</code>	80, 330, 1508, 1510	
<code>\@pdfcontactaddress</code>		
.	<u>199</u> ,	
331, 553, 1571, 1672		
<code>\@pdfcontactcity</code> <u>207</u> ,		
332, 559, 1572, 1673		
<code>\@pdfcontactcountry</code>		
.	<u>213</u> ,	
333, 571, 1575, 1676		
<code>\@pdfcontactemail</code>	217, 334,	
476, 547, 1578, 1678		
<code>\@pdfcontactphone</code>	<u>215</u> ,	
335, 480, 1577, 1677		
<code>\@pdfcontactpostcode</code>		
.	<u>211</u> ,	
336, 577, 1574, 1675		
<code>\@pdfcontactregion</code>	<u>209</u> ,	
337, 565, 1573, 1674		
<code>\@pdfcontacturl</code> <u>219</u> ,		
338, 483, 1579, 1679		
<code>\@pdfcopyright</code>	72, 339,	
1425, 1460, 1466		
<code>\@pdfcreationdate</code>	340,	
494, 495, 1487, 1491		
<code>\@pdfcreator</code>	1504	
<code>\@pdfdatetime</code>	<u>50</u> , 341, 1427, 1430	

hyperxmp-add-
 bytecount 17, 18, 99
 \hyxmp@is@unicode . 908
 \hyxmp@acm@isbn 597
 \hyxmp@acm@publisher
 611
 \hyxmp@acm@pubtype . 620
 \hyxmp@add@simple 1063,
 1313, 1422, 1437,
 1451, 1468, 1470,
 1481, 1482, 1483,
 1484, 1488, 1490,
 1494, 1496, 1500,
 1502, 1504, 1505,
 1509, 1510, 1514,
 1515, 1519, 1526,
 1527, 1530, 1532,
 1572, 1573, 1574,
 1575, 1589, 1593,
 1595, 1596, 1597,
 1598, 1599, 1600,
 1601, 1602, 1603,
 1604, 1607, 1613
 \hyxmp@add@simple@lang
 1081,
 1591, 1592, 1594
 \hyxmp@add@simple@lang@i
 1084, 1087
 \hyxmp@add@simple@lang@ii
 1088, 1090
 \hyxmp@add@simple@pfx
 1104, 1417
 \hyxmp@add@simple@var
 1072,
 1311, 1312, 1315
 \hyxmp@add@to+xml 1066,
 1068, 1076, 1094,
 1098, 1106, 1110,
 1112, 1231, 1257,
 1326, 1345, 1352,
 1358, 1365, 1370,
 1382, 1390, 1396,
 1405, 1545, 1549,
 1553, 1559, 1566,
 1581, 1706, 1712,
 1718, 1728, 1735,
 1739, 1747, 1860,
 1899, 2034,
 2052, 2056, 2085
 \hyxmp@address@val 545, 551,
 557, 563, 569, 575
 \hyxmp@aep@toks 25,
 308, 401, 1473, 2226
 \hyxmp@alt@description
 1116, 1126
 \hyxmp@alt@rights 1116, 1127
 \hyxmp@alt@title 1116, 1125
 \hyxmp@and 264
 \hyxmp@append@hex 1165, 1184,
 1185, 1186, 1190
 \hyxmp@append@hex@iii
 1183,
 1189, 1199, 1210
 \hyxmp@append@hex@iv
 1188,
 1194, 1195, 1197,
 1212, 1213, 1214
 \hyxmp@as@pdf@date . 734
 \hyxmp@as@xmp@date 56, 67,
 706, 844, 1491, 1497
 \hyxmp@assign@major@minor
 1295, 1314
 \hyxmp@at@end 9, 419
 \hyxmp@auto@assign@data
 420, 463
 \hyxmp@banner@to@producer
 1271, 1274, 1282
 \hyxmp@begin@ext@decl
 1717,
 1756, 1767, 1793,
 1809, 1823, 1839,
 1851, 1909, 1979
 \hyxmp@begin@extension@decls
 1705, 1991
 \hyxmp@big@prime 1139,
 1142, 1152, 1162
 \hyxmp@big@prime@ii
 1139, 1161
 \hyxmp@bom 2019, 2034
 \hyxmp@cct 1611, 1615
 \hyxmp@check@iptc@data
 1670, 2068
 \hyxmp@check@jav@data
 1700, 2070
 \hyxmp@check@prism@data
 1682, 2069
 \hyxmp@check@standards
 373, 421
 \hyxmp@check@std 112, 124, 125,
 126, 127, 128,
 129, 130, 131, 132
 \hyxmp@comma 201, 265, 286, 677
 \hyxmp@commas@to@list
 661, 697, 1388, 1557
 \hyxmp@commas@to@list@i
 663, 665
 \hyxmp@concat@metadata
 308, 322
 \hyxmp@cond@dc@identifier
 1414, 1446,
 1447, 1448, 1449
 \hyxmp@construct@packet
 2032, 2094
 \hyxmp@count@non@spaces
 2194, 2205
 \hyxmp@count@spaces
 2193, 2196
 \hyxmp@crap@convert
 990, 1024
 \hyxmp@crap@result 980, 1016
 \hyxmp@crap@test 987, 1012
 \hyxmp@create@uuid 1192, 1220, 1229
 \hyxmp@cur@lang 1122, 1130
 \hyxmp@dc@lang . 465,
 636, 642, 658, 1439
 \hyxmp@dc@schema 1421, 2074
 \hyxmp@declare@extensions
 1990, 2071
 \hyxmp@declare@field
 1746,
 1875, 1878, 1881,
 1884, 1887,
 1890, 1893, 1896
 \hyxmp@declare@property
 1734, 1760,
 1771, 1776, 1781,
 1785, 1797, 1802,
 1813, 1827, 1831,
 1843, 1855, 1913,
 1917, 1921, 1925,
 1929, 1933, 1937,
 1941, 1945, 1949,
 1953, 1957, 1962,
 1967, 1972, 1983

<code>\hyxmp@def@DocumentID</code>	<code>\hyxmp@iptc@data</code>	<code>1145, 2197,</code>
..... 1216 , 1476	.. 1564 , 1670 , 2005	2198 , 2206 , 2207
<code>\hyxmp@def@InstanceID</code>	<code>\hyxmp@iptc@extensions</code>	<code>\hyxmp@orig@ifdraft</code>
..... 1222 , 1477 1850 , 2007 144 , 163
<code>\hyxmp@define@pdfproducer</code>	<code>\hyxmp@iptc@schema</code>	<code>\hyxmp@padding</code>
..... 1268 , 1292 1563 , 2081	1255 , 2089
<code>\hyxmp@detect@langs</code>	<code>\hyxmp@is@unicode</code>	<code>\hyxmp@parse@acmart</code>
..... 467 , 637 876 , 893 , 908 492 , 542 , 635
<code>\hyxmp@DocumentID</code>	<code>\hyxmp@jav@data</code>	<code>\hyxmp@parse@time</code>
..... 138 , 1700 , 2013 715 , 717
1216 , 1476 , 1481	<code>\hyxmp@jav@extensions</code>	<code>\hyxmp@parse@tz</code>
<code>\hyxmp@dq@code</code> 1978 , 2015 724 , 727 , 731
.. 7 , 2227	<code>\hyxmp@jav@schema</code>	<code>\hyxmp@parse@tz@char</code>
<code>\hyxmp@driver</code> 1606 , 2083 719 , 721
.... 2093	<code>\hyxmp@jobname</code>	<code>\hyxmp@pdf@extensions</code>
<code>\hyxmp@embed@packet</code> 1755 , 1992
..... 423 , 2093	... 22 , 23 , 136 ,	<code>\hyxmp@pdf@schema</code>
<code>\hyxmp@embed@packet@dvipdfm</code>	367 , 526 , 842 , 1310 , 2072
..... 2105 , 2175	1217 , 1225 , 2114	<code>\hyxmp@pdf@to@xmp@date</code>
<code>\hyxmp@embed@packet@luatex</code>	<code>\hyxmp@koma@phones</code>	.. 708 , 713 , 836 , 839
..... 2101 , 2138 425 , 481	<code>\hyxmp@pdfa@id@extensions</code>
<code>\hyxmp@embed@packet@pdfmark</code>	<code>\hyxmp@LA@accept</code> 1792 , 1995
..... 2117 , 2145 1119 ,	<code>\hyxmp@pdfa@id@schema</code>
<code>\hyxmp@embed@packet@pdftex</code>	1125 , 1126 , 1127 1512 , 2077
..... 2097 , 2125	<code>\hyxmp@lang@name</code>	<code>\hyxmp@pdfauthor</code>
<code>\hyxmp@embed@packet@xetex</code>	.. 642	..
..... 2109 , 2213	<code>\hyxmp@lang@tag</code>	... 255 , 264 , 1433
<code>\hyxmp@end@ext@decl</code> 1455	<code>\hyxmp@pdfkeywords</code>
..... 1727 ,	<code>\hyxmp@list</code>	... 255 , 285 , 1434
1764 , 1790 , 1806 ,	... 1388 ,	<code>\hyxmp@pdfstringdef</code>
1818 , 1835 ,	1394 , 1557 , 1558 44 ,
1847 , 1976 , 1988	<code>\hyxmp@list@to@lines</code>	... 55 , 66 , 73 , 75 ,
<code>\hyxmp@end@extension@decls</code>	... 1542 , 1571 ,	77 , 79 , 81 , 83 , 93 ,
..... 1711 , 2017	1577 , 1578 , 1579	97 , 102 , 120 , 137 ,
<code>\hyxmp@extra@indent</code>	<code>\hyxmp@list@to@xml</code>	139 , 141 , 143 ,
..... 1062 , 1376 ,	164 , 166 , 168 ,
1066 , 1077 ,	1433 , 1434 , 1439	170 , 172 , 174 ,
1106 , 1546 , 1570	<code>\hyxmp@major@minor</code>	176 , 178 , 180 ,
<code>\hyxmp@first@char</code>	1295	182 , 184 , 186 ,
.. 704	<code>\hyxmp@mm@extensions</code>	188 , 190 , 192 ,
<code>\hyxmp@first@char@i</code> 1766 , 1993	194 , 196 , 198 ,
.... 704 , 707 , 735	<code>\hyxmp@mm@schema</code>	203 , 208 , 210 ,
<code>\hyxmp@gobbletwo</code> 1480 , 2080	212 , 214 , 216 ,
774 , 787	... 1133 , 1152 ,	218 , 220 , 430 ,
<code>\hyxmp@hash</code>	1162 , 1168 , 1203	432 , 436 , 1108 , 1121
.... 1251 , 2038 ,	<code>\hyxmp@multi@langsfalse</code>	<code>\hyxmp@pdfua@id@extensions</code>
2046 , 2060 , 2063 , 1317 , 1333 1808 , 1999
2064 , 2065 , 2066	<code>\hyxmp@multi@langstrue</code>	<code>\hyxmp@pdfua@id@schema</code>
<code>\hyxmp@Hyp@pdfauthor</code> 1317 , 1331 1518 , 2078
..... 258	<code>\hyxmp@new@xml</code>	<code>\hyxmp@pdfx@id@extensions</code>
<code>\hyxmp@Hyp@pdfkeywords</code>	1247 , 1248 1820 , 2003
..... 279	<code>\hyxmp@no@bad@parts</code>	<code>\hyxmp@pdfx@id@schema</code>
<code>\hyxmp@hypersetup</code> 84 , 92 , 101 1521 , 2079
.. 303	<code>\hyxmp@no@info@lists</code>	<code>\hyxmp@pdfx@major</code>
<code>\hyxmp@InstanceID</code> 221 , 245 , 405	..
..... 140 ,	<code>\hyxmp@num</code> 109 ,
1222 , 1477 , 1482 1024	
<code>\hyxmp@iprefix</code>	<code>\hyxmp@one@token</code>	
1108 , 1109 1141 ,	

	118, 134 , 1522 , 1821, 1837 , 2051		
<code>\hyxmp@photoshop@data</code> 1507	<code>\hyxmp@singleton@dc</code>	... 1402 , 1426 , 1428 , 1430 , 1432
<code>\hyxmp@photoshop@schema</code> 1507 , 2075	<code>\hyxmp@skiptorelax</code> 1017 , 1023
<code>\hyxmp@prev@pdf@size</code> 524 , 539	<code>\hyxmp@skipzeros</code>	.. 975
<code>\hyxmp@prism@data</code>	.. 1587 , 1682 , 2009	<code>\hyxmp@SpaceOther</code> 984 , 997
<code>\hyxmp@prism@extensions</code> 1908 , 2011	<code>\hyxmp@standards</code>	.. 386
<code>\hyxmp@prism@schema</code> 1586 , 2082	<code>\hyxmp@string@len</code> 2176 , 2191
<code>\hyxmp@ProcessKeyvalOptions</code> 297	<code>\hyxmp@strip@isbn@date</code> 597
<code>\hyxmp@prot@us</code>	... 1619	<code>\hyxmp@sublist</code> 666 , 667 , 670 , 671
<code>\hyxmp@query@self</code> 499 , 533	<code>\hyxmp@suppress@pdf@info</code> 222
<code>\hyxmp@rand@num</code>	... 1158 , 1167 , 1202 , 1219 , 1228	<code>\hyxmp@temp@list</code>	.. 690
<code>\hyxmp@rdf@dc</code>	.. 1318 , 1423 , 1424 , 1425	<code>\hyxmp@temp@str</code>	... 690
<code>\hyxmp@redefine@Hyp</code> 257 , 300 , 305	<code>\hyxmp@text</code> 872 , 950 , 980 , 1024
<code>\hyxmp@remove@this</code> 1286 , 1289	<code>\hyxmp@textunderscore</code> 44
<code>\hyxmp@rights</code>	.. 1455 , 1458 , 1462 , 1464	<code>\hyxmp@timestamp</code>	.. 841
<code>\hyxmp@seed@rng</code> 1141 , 1218 , 1227	<code>\hyxmp@today@pdf</code>	495 , 851
<code>\hyxmp@seed@rng@i</code> 1143 , 1145	<code>\hyxmp@today@xmp</code> 829 , 834 , 852 , 1225 , 1428 , 1488 , 1494 , 1500
<code>\hyxmp@seed@string</code> 1216 , 1222	<code>\hyxmp@today@xmp@define</code>	... 800 , 849 , 1223
<code>\hyxmp@set@jobname</code> 19 , 24	<code>\hyxmp@toxml</code>	.. 902 , 925
<code>\hyxmp@set@jobname@dbl</code> 20 , 22	<code>\hyxmp@toxml@unicodetex</code> 890 , 950
<code>\hyxmp@set@jobname@plain</code> 20 , 23	<code>\hyxmp@trimb</code>	.. 858 , 861
<code>\hyxmp@set@koma@phones</code> 425 , 479	<code>\hyxmp@trimc</code>	.. 861 , 862
<code>\hyxmp@set@pdfx@major</code> 104 , 134	<code>\hyxmp@trimspace</code> 670 , 854
<code>\hyxmp@set@pdfx@major@i</code> 104 , 105	<code>\hyxmp@try</code> 980
<code>\hyxmp@set@pdfx@major@ii</code> 106 , 109	<code>\hyxmp@try@today</code>	828 , 835 , 838 , 841 , 848
<code>\hyxmp@set@rand@num</code>	.. 1158 , 1166 , 1201	<code>\hyxmp@unicodetexfalse</code> 864
		<code>\hyxmp@unicodetextrue</code> 864
		<code>\hyxmp@uscore</code>	.. 46 , 681
		<code>\hyxmp@use@first@valid</code>	.. 408 , 412 , 442 , 476 , 480 , 483 , 486 , 489 , 538 , 547 , 553 , 559 , 565 , 571 , 577 , 590 , 604 , 612 , 615 , 625 , 628 , 631
		<code>\hyxmp@use@first@valid@i</code> 444 , 448
		<code>\hyxmp@value</code>	1121 , 1325
		<code>\hyxmp@warn@if@no@metadata</code> 322 , 422
		<code>\hyxmp@x@default</code> 471 , 1267 , 1335 , 1346 , 1353
		<code>\hyxmp@xetex@crap</code> 881 , 980
		<code>\hyxmp@xml</code>	1067 , 1069 , 1107 , 1113 , 1248 , 1255 , 1738 , 2032 , 2134 , 2142 , 2165 , 2176 , 2183 , 2214
		<code>\hyxmp@xmllified</code> 872 , 1068 , 1077 , 1084 , 1095 , 1099 , 1110 , 1112 , 1325 , 1354 , 1359 , 1366 , 1388 , 1408 , 1415 , 1445 , 1557
		<code>\hyxmp@xmlify</code> 872 , 1065 , 1075 , 1083 , 1092 , 1109 , 1111 , 1324 , 1351 , 1357 , 1364 , 1387 , 1404 , 1556
		<code>\hyxmp@xmp@basic@schema</code> 1486 , 2076
		<code>\hyxmp@xmp@to@pdf@date</code> 738 , 741 , 852
		<code>\hyxmp@xmp@to@pdf@date@i</code> 742 , 744
		<code>\hyxmp@xmp@to@pdf@date@ii</code> 747 , 750
		<code>\hyxmp@xmp@to@pdf@date@iii</code> 753 , 756
		<code>\hyxmp@xmp@to@pdf@date@iv</code> 759 , 762
		<code>\hyxmp@xmp@to@pdf@date@v</code> 765 , 768
		<code>\hyxmp@xmp@to@pdf@date@vi</code> 771 , 775
		<code>\hyxmp@xmp@to@pdf@date@vii</code> 778 , 781 , 791
		<code>\hyxmp@xmp@to@pdf@date@viii</code> 794 , 797
		<code>\hyxmp@xmpRights@schema</code> 1454 , 2073
		<code>\hyxmp@xmptpg@schema</code> 1609 , 2084
		<code>\hyxmp@zero</code> 1033 , 1040 ,

	1047, 1053, 1058		
I			
IETF	6	jav:journal_article_version	2
\if@ACM@journal	620	\jobname	24
\if@tempswa	1323, 1381	Journal Article Versions	
\IfDocumentMetadataTF	1, 6	schema	59, 71
ifdraft (package)	18, 24, 99	K	
\ifdraft	144	keeppdfinfo (option)	12, 27
\iffalse	1318, 1376	Keywords	11, 27, 62, 97
\ifHy@pdfa	374, 1423, 1424, 1433, 1513, 1994	Koma (class)	15, 33, 97
\ifhyxmp@multi@langs	1317, 1336, 1350	\KV@Hyp@pdfauthor	264
\ifhyxmp@unicodetex	864, 875, 1234, 2020	\KV@Hyp@pdfkeywords	285
\ifLuaTeX	34, 501, 1270, 1610, 1615, 1623	kvoptions (package)	20, 30
ifluatex (package)	84	L	
\ifluatex	2127, 2131	latexmk	17, 18, 36, 99
ifmtarg (package)	20, 96	LF	69
\ifoptiondraft	153	\LocaleForEach	643
\ifoptionfinal	154	Lua	72
\ifPDFTeX	1273	luacode (package)	20
\IfSubStr	588, 589, 598, 599	\luadirect	525, 1611
iftex (package)	20, 97	LuaL ^A T _E X	9, 11, 13, 14, 17, 36, 47, 62, 95, 98
ifthen (package)	20	LuaT _E X	20, 36, 49, 51, 71, 84, 87, 93, 95, 97, 98
\ifthenelse	113	luatex85 (package)	97
\ifXeTeX	880, 1276	\luatexbanner	1271
Info	11, 12, 27, 32, 35, 61	M	
intcalc (package)	20	\makeatletter	1286
\intcalcDiv	1029, 1036, 1043	memoir (package)	94
\intcalcMod	1031, 1038, 1045	Metadata	83, 86
IPTC	12, 26, 59, 70, 73, 77, 81, 94, 105, 108	\month	802, 803, 805
IPTC Photo Metadata	59, 69–71	N	
lptc4xmpCore:Contact-Info	70, 77	NAK	20, 42, 49
lptc4xmpCore:Creator-ContactInfo	2, 3, 70, 94	nativepdf (option)	84
ISBN	2, 7, 25, 39	\newcatcodetable	1616
ISO	6, 7, 15, 22, 56	\newcount	89
ISSN	2, 7, 25	\newif	864, 1317
J		\newtoks	25
JAV	74, 80, 81	\next	54, 65, 114, 121, 144, 222, 449, 451, 457, 461, 665, 843, 846, 1145, 2001
		ngerman (package)	18, 93
		NISO	7, 71
		\number	1027, 1029, 1031, 1036, 1038, 1043, 1045
		\numexpr	2143
		O	
		options	
		baseurl	5, 7, 15, 20, 26, 68
		draft	8
		dvipdf	84
		dvips	84
		dvipsone	84
		dviwindo	84
		keeppdfinfo	12, 27
		nativepdf	84
		pdfa	9, 22, 31, 95
		pdfaconformance	5, 9, 69
		pdfapart	5, 9, 22, 69, 99
		pdfauthor	5, 6, 11, 14, 15, 20, 28, 29, 32, 66, 95, 98
		pdfauthortitle	5, 6, 15
		pdfbookedition	5, 7
		pdfbytes	5, 9, 37, 98
		pdfcaptionwriter	5, 6
		pdfcontactaddress	5, 6, 12, 13
		pdfcontactcity	5, 6
		pdfcontactcountry	5, 6
		pdfcontactemail	5, 6
		pdfcontactphone	5, 6
		pdfcontactpostcode	5, 6
		pdfcontactregion	5, 6
		pdfcontacturl	5, 6, 15
		pdfcopyright	5, 6, 66, 67, 94
		pdfcreationdate	5, 8, 35, 95
		pdfdate	5, 7, 8, 15, 21, 59, 66, 95, 97
		pdfdocumentid	5, 6, 96
		pdfdoi	5, 7
		pdfeissn	5, 7
		pdfidentifier	5, 7, 66, 97
		pdfinstanceid	5, 6, 96
		pdfisbn	5–7
		pdfissn	5, 7
		pdfissuenumber	5, 7, 11, 14, 20, 28, 66
		pdflang	5–7, 15, 20, 40, 41, 55, 66, 95
		pdflicenseurl	5, 6, 15, 67, 94
		pdfmark	84
		pdfmetadata	5, 8, 21, 95

pdfmetalang . . . 5, 6,
15, 55, 63, 64, 93, 95
pdfmoddate . . . 5, 8, 95
pdfnumpages . . . 5, 9, 98
pdfpagerange
. 5, 7, 16, 17
pdfproducer . . . 5, 20, 61
pdfpublication . . . 5–7
pdfpublisher 5, 7
pdfpubstatus 5, 7, 8, 98
pdfpubtype 5, 7
pdfrendition . . . 5, 8, 97
pdfsource 5, 9, 96
pdfsubject 5, 11, 20, 66
pdfsubtitle 5
pdftitle 5, 11,
15, 20, 32, 66, 95, 98
pdftrapped
. 5, 8, 9, 20, 97
pdftype . . . 5, 8, 66, 95
pdfuapart . . . 5, 9, 22, 69
pdfurl 5, 7
pdfversionid . . . 5, 6, 67
pdfvolumenum . . . 5, 7
pdfxstandard
. 5, 9, 22, 23, 69, 82
ps2pdf 84
textures 84
unicode 15, 96
vtxpdfmark 84

P

\PackageError 249, 1337
packages
acmart 99
atenddvi 19, 97
babel 15,
18, 30, 35, 40, 41, 98
doclicense 98
etoolbox 20
gitver 6
hyperref 1,
4–6, 8, 9, 11, 15,
18–20, 22, 27, 28,
30–32, 41, 61, 62,
83, 84, 94–96, 98, 99
hyperxmp
. 1, 2, 4–9, 11–20,
22–24, 26, 28, 30,
32–36, 41–43, 49,
57, 61, 62, 73,
84, 87, 94, 95, 97–99
ifdraft 18, 24, 99
ifluatex 84
ifmtarg 20, 96
iftex 20, 97
ifthen 20
intcalc 20
kvoptions 20, 30
luacode 20
luatex85 97
memoir 94
ngerman 18, 93
pdfescape 20
pdfx 4
polyglossia . . . 15, 18,
30, 33, 35, 40, 41, 98
stringenc 20
texdate 16
totpages 16, 17
xmpincl 4
\PackageWarning
. 2, 86, 122, 691
\PackageWarningNoLine
. 224,
366, 376, 393, 2112
\patchcmd 230, 236
PDF 1–5, 8,
11–14, 16–18, 21,
27, 31–33, 35–37,
41, 43, 44, 48, 49,
51, 59, 61, 62,
71, 75, 80, 81,
83, 84, 86, 94,
95, 97–99, 101, 108
Author . . . 11, 27, 97
CreationDate . . . 35, 95
Info 11,
12, 27, 32, 35, 61
Keywords 11, 27, 62, 97
Metadata 83, 86
Producer 62
Subject 11
Title 11
PDF/A . . . 3, 9, 11, 12,
22, 27, 32, 61, 62,
68, 69, 73, 76–78,
80, 94, 95, 106, 108
PDF/A Identification
schema . . . 59, 68–69
PDF/UA 3, 9, 22, 32, 69,
76, 80, 96, 106, 108
PDF/UA Identification
schema . . . 59, 68–69
PDF/X 3, 9, 22,
23, 32, 69, 77, 81,
82, 97, 99, 106, 108
PDF/X Identification
schema . . . 59, 68–69
pdf:Keywords 2, 11, 62
pdf:PDFVersion . . . 3, 62, 97
pdf:Producer 3, 61, 62
pdf:trapped 3
\PDF@FinishDoc
. 223, 231, 237
pdfa (option) 9, 22, 31, 95
pdfaconformance (op-
tion) 5, 9, 69
pdfaid:conformance . . . 3
pdfaid:part 3
pdfapart (option)
. 5, 9, 22, 69, 99
pdfaType:prefix 94
pdfauthor (option) . . .
. 6, 11, 14, 15, 20,
28, 29, 32, 66, 95, 98
pdfauthortitle (option)
. 5, 6, 15
pdfbookedition (option)
. 5, 7
pdfbytes (option)
. 5, 9, 37, 98
pdfcaptionwriter (op-
tion) 5, 6
\pdfcatalog 2135
\pdfcompresslevel . . . 2129
pdfcontactaddress (op-
tion) . . . 5, 6, 12, 13
pdfcontactcity (option)
. 5, 6
pdfcontactcountry (op-
tion) 5, 6
pdfcontactemail (op-
tion) 5, 6
pdfcontactphone (op-
tion) 5, 6
pdfcontactpostcode (op-
tion) 5, 6
pdfcontactregion (op-
tion) 5, 6
pdfcontacturl (option)
. 5, 6, 15
pdfcopyright (option) . .
. 5, 6, 66, 67, 94
pdfcreationdate (option)
. 5, 8, 35, 95
\pdfcreationdate . . . 836
pdfdate (option) 5, 7, 8,
15, 21, 59, 66, 95, 97
PDFDocEncoding
. 28, 49, 50

pdfdocumentid (option) 5, 6, 96
pdfdoi (option) 5, 7
pdfeissn (option) 5, 7
pdfescape (package) 20
\pdfextension 2139, 2143
\pdffeedback . 839, 2143
pdfidentifier (option) 5, 7, 66, 97
pdfinstanceid (option) 5, 6, 96
pdfisbn (option) 5–7
pdfissn (option) 5, 7
pdfissuenum (option) 5, 7
pdfkeywords (option) 5, 11, 14, 20, 28, 66
pdflang (option) 5–7, 15, 20, 40, 41, 55, 66, 95
\pdflastobj 2135
pdfL^AT_EX 4, 9, 11, 14, 36, 47, 62, 98, 99
pdflicenseurl (option) 5, 6, 15, 67, 94
\pdfmajorversion . 1303
pdfmark (option) 84
\pdfmark 2146, 2149, 2153, 2163, 2167, 2171
pdfmetadate (option) 5, 8, 21, 95
pdfmetalang (option) 5, 6, 15, 55, 63, 64, 93, 95
\pdfminorversion . 1299
pdfmoddate (option) 5, 8, 95
pdfnumpages (option) 5, 9, 98
\pdfobj 2131
pdfpagerange (option) 5, 7, 16, 17
pdfproducer (option) 5, 20, 61
pdfpublication (option) 5–7
pdfpublisher (option) 5, 7
pdfpubstatus (option) 5, 7, 8, 98
pdfpubtype (option) 5, 7
pdfrendition (option) 5, 8, 97
pdfsource (option) 5, 9, 96
\pdfstringdef 47
pdfsubject (option) 5, 11, 20, 66
pdfsubtitle (option) 5
pdfT_EX 50, 84, 87, 95, 97
\pdftexbanner 1274
pdftitle (option) 5, 11, 15, 20, 32, 66, 95, 98
pdftrapped (option) 5, 8, 9, 20, 97
pdftype (option) 5, 8, 66, 95
pdfuaid:part 3
pdfuapart (option) 5, 9, 22, 69
pdfurl (option) 5, 7
\pdfvariable 1307
pdfversionid (option) 5, 6, 67
pdfvolumenum (option) 5, 7
pdfx (package) 4
pdfxid:GTS_PDFXVersion 3
pdfxstandard (option) 5, 9, 22, 23, 69, 82
Perl 17
Photoshop schema 59, 68
photoshop:AuthorsPosition 3, 68
photoshop:Caption-Writer 2, 68
PI 59
polyglossia (package) 15, 18, 30, 33, 35, 40, 41, 98
\postcode 575
PRISM 7, 17, 59, 71, 73, 78, 81, 107, 108
PRISM Basic Metadata schema . 17, 59, 71
prism:aggregationType . 3
prism:bookEdition 2
prism:byteCount 2, 17
prism:doi 2
prism:elssn 2
prism:isbn 2
prism:issn 2
prism:number 2
prism:pageCount 3
prism:pageRange 3
prism:publicationName . 3
prism:subtitle 3
prism:url 3
prism:volume 3
\ProcessKeyvalOptions 297
Producer 62
properties, XMP
dc:creator 2, 11, 66, 95
dc:date 2, 66
dc:description 3, 11, 55, 66, 95
dc:format 2
dc:identifier . 3, 65, 66
dc:language 2, 15, 30, 66, 93, 94, 98
dc:publisher 3
dc:rights 2, 16, 55, 66
dc:source 2, 66, 93
dc:subject 2, 66
dc:title 3, 11, 55, 65, 95
dc:type 3, 66
lptc4xmpCore:ContactInfo 70, 77
lptc4xmpCore:CreatorContactInfo 2, 3, 70, 94
jav:journal_article_version 2
pdf:Keywords 2, 11, 62
pdf:PDFVersion 3, 62, 97
pdf:Producer 3, 61, 62
pdf:trapped 3
pdfaid:conformance . 3
pdfaid:part 3
pdfaType:prefix 94
pdfuaid:part 3
pdfxid:GTS_PDFXVersion 3
photoshop:Author-Position 3, 68
photoshop:Caption-Writer 2, 68
prism:aggregation-Type 3
prism:bookEdition 2
prism:byteCount . 2, 17
prism:doi 2
prism:elssn 2
prism:isbn 2
prism:issn 2
prism:number 2
prism:pageCount 3
prism:pageRange 3
prism:publication-Name 3
prism:subtitle 3
prism:url 3
prism:volume 3
xmp:BaseURL 2

xmp:CreateDate	2, 35, 95	xmp:CreatorTool	3	xmp:MetadataDate	2, 95	xmp:ModifyDate	2, 95	xmpMM:DocumentID	3, 56, 67, 80	xmpMM:InstanceID	3, 56, 67, 80	xmpMM:RenditionClass	3, 98	xmpMM:VersionID	3, 67, 96	xmpRights:Marked	2, 66, 94	xmpRights:WebStatement	2, 66, 94	ps2pdf (option)	84	ps2pdf	11																								
Q																																															
\Q	854, 863																																														
R																																															
RDF	64	rdf:Description	96	rdf:li	2	rdf:Seq	2	\renewcommand	298	\RequirePackage	11, 26, 27, 28, 29, 30, 31, 32, 33, 35, 149, 404, 2124																																				
S																																															
\savecatcodetable	1617	\scantokens	1283, 1286	schemata																																											
Adobe PDF	59, 61–62	Dublin Core	2, 59, 62–66	IPTC Photo Metadata	59, 69–71	Journal Article Versions	59, 71	PDF/A Identification	59, 68–69	PDF/UA Identification	59, 68–69	PDF/X Identification	59, 68–69	Photoshop	59, 68	PRISM Basic Metadata	17, 59, 71																														
XMP Basic	59, 68	XMP Media Management	59, 67	XMP Paged-Text	59, 71–73	XMP Rights Management	59, 66–67	\scr@fromemail@var	477	\scr@frommobilephone@var	430, 432	\scr@fromname@var	413	\scr@fromphone@var	430, 436	\scr@fromurl@var	484	\scr@subject@var	409	scrltr2 (class)	15	\SE->pdfdoc@03	870	\SE->pdfdoc@15	871	\setbox	585	\setkeys	1131	\special	2177, 2185, 2214, 2220	\state	563	\streetaddress	551	stringenc (package)	20	\StringEncodingConvert	877, 883, 894, 897, 992	Subject	11						
T																																															
TeX	16, 19, 20, 46, 47, 49–51, 56, 57, 59, 61, 67, 86, 87, 97	texdate (package)	16	Text	75	\textunderscore	45, 46, 48	textures (option)	84	\thetotalpages	536	\time	812, 820	Title	11	totpages (package)	16, 17																														
U																																															
\undefined	453	Unicode	15, 20, 49–53, 65, 70, 81, 87, 93	unicode (option)	15, 96	URI	7	URL	2, 3, 6, 7, 15, 21, 26, 27, 66–68, 70	UTF-16BE	50	UTF-32BE	49, 50	UTF-8	50	UUID	3, 6, 23, 24, 56, 58, 59, 67, 94																														
V																																															
\vfuzz	862	vtexpdfmark (option)	84																																												
X																																															
\x	980	x-default	6, 15, 35, 55, 60, 63, 64, 93, 95, 98	xdvipdfmx	14, 35, 86	X _q LaTeX	11, 14, 35, 47, 62, 95, 97	X _f TeX	49, 51, 52, 86, 87, 93, 97, 98	\XeTeXrevision	1277	\XeTeXversion	1277	XML	1, 2, 12, 41, 49–51, 54, 55, 59, 62–65, 69, 70, 74, 81, 96	XMP	1, 2, 4–8, 11–19, 21, 23, 27, 32, 33, 35, 38, 39, 41–44, 47, 48, 51, 54–56, 59–64, 66–68, 71–73, 75, 80, 84–87, 93–96, 98, 108	properties	see properties, XMP	XMP Basic schema	59, 68	XMP Media Management schema	59, 67	XMP Paged-Text schema	59, 71–73	XMP Rights Management schema	59, 66–67	xmp:BaseURL	2	xmp:CreateDate	2, 35, 95	xmp:CreatorTool	3	xmp:MetadataDate	2, 95	xmp:ModifyDate	2, 95	\xmpcomma	201, 204, 264, 285, 676	xmpincl (package)	4	\XMLLangAlt	1128, 1338	\xmplinesep	1537, 1553, 1576	xmpMM:DocumentID	3, 56, 67, 80

xmpMM:InstanceID ..	\xmpquote	202,	\xmptilde	<u>686</u>
..... 3, 56, 67, 80		205, <u>264</u> , <u>285</u> , <u>685</u>	\XMPTruncateList ..	<u>690</u>
xmpMM:RenditionClass	xmpRights:Marked	2, 66, 94	\xpg@bcp@loaded ...	658
..... 3, 98	xmpRights:WebState-			
xmpMM:VersionID 3, 67, 96	ment ...	2, 66, 94	Y	
			\year	801