

# The hyperxmp package\*

Scott Pakin  
scott+hyxmp@pakin.org

June 13, 2020

## Abstract

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

## 1 Introduction

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

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

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

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

---

\*This document corresponds to `hyperxmp` v5.3, dated 2020/06/13.

```

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

```

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

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

**What metadata does hyperxmp process?** hyperxmp knows how to embed all of the following types of metadata within a document:

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

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

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

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

`xmpincl` can embed XMP only when running under pdf $\LaTeX$  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.

## 2 Usage

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

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

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

- `pdfaconformance`
- `pdfcontactcountry`
- `pdfdocumentid`
- `pdfapart`
- `pdfcontactemail`
- `pdfdoi`
- `pdfauthortitle`
- `pdfcontactphone`
- `pdfeissn`
- `pdfbookedition`
- `pdfcontactpostcode`
- `pdfidentifier`
- `pdfbytes`
- `pdfcontactregion`
- `pdfinstanceid`
- `pdfcaptionwriter`
- `pdfcontacturl`
- `pdfisbn`
- `pdfcontactaddress`
- `pdfcopyright`
- `pdfissn`
- `pdfcontactcity`
- `pdfdate`
- `pdfissuenum`

- pdflicenseurl
- pdfpublisher
- pdfuapart
- pdfmetadate
- pdfpubtype
- pdfurl
- pdfmetalang
- pdfrendition
- pdfversionid
- pdfnumpages
- pdfsource
- pdfvolumenum
- pdfpagerange
- pdfsubtitle
- pdfxstandard
- pdfpublication
- pdftype

## 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`. **pdfauthortitle** indicates the primary author’s position or title. **pdfcaptionwriter** specifies the name of the person who added the metadata to the document.

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

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

**pdfmetalang** indicates the natural language in which certain metadata—specifically, the document’s title, subject, and copyright statement—are written. The language should be specified using an IETF language tag [10], for example, “en” for English, “en-US” for specifically United States English, “de” for German, and so forth. If `pdfmetalang` is not specified, `hyperxmp` assumes the metadata

pdflang	language is the same as the document language (hyperref's pdflang option). If neither pdfmetalang nor pdflang is specified, hyperxmp uses only "x-default" as the metadata language. Note that "x-default" metadata are always included in addition to the specified metadata language, as the user reading the document may not have specified a language preference.
pdfdocumentid pdfinstanceid	XMP can include a universally unique identifier (UUID) for each document and for each instance of a given document. By default, hyperxmp assigns a version 4 (i.e., pseudorandom) UUID [11] for each of these. 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 <code>uuid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx</code> , where "x" is a lowercase hexadecimal number. For example, <code>uuid:53ab7f19-a48c-5177-8bb2-403ad907f632</code> is a valid argument to pdfdocumentid (or pdfinstanceid). See Leach, Mealling, and Salz's UUID specification document for details on how to produce the various forms of UUIDs [11]. A more freeform mechanism than 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 5.3 to match the package version); or labeled using any other approach. One possibility is to use a revision number or commit hash from the version-control software maintaining the document. For example, the <code>\gitVer</code> macro from the <code>gitver</code> package is an expandable (see Note 8 on page 16) version of the current Git hash that can suitably be passed to pdfversionid. If not specified, pdfversionid defaults to 1.
pdfisbn pdfissn pdfeissn pdfdoi	Already-published documents can be identified in a number of ways. pdfisbn specifies the ISBN. pdfissn refers to the ISSN of the <i>print</i> version of the document while pdfeissn refers to the ISSN of the <i>electronic</i> version of the document. pdfdoi 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> .
pdfurl baseurl	pdfurl points to the complete URL for the document. In contrast, baseurl points one level up and is used to resolve relative URLs.
pdfidentifier	pdfidentifier provides an alternative mechanism to uniquely identify a document. Its advantage relative to pdfisbn, pdfissn, pdfdoi, etc. is its flexibility; any of a wide variety of identification types can be used. <sup>1</sup> pdfidentifier'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 pdfidentifier 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 pdfidentifier is not specified explicitly, hyperxmp 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.

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

pdfpublication	<p>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>.</p>
pdfpublisher pdfpubtype	<p><code>pdfpubtype</code> indicates the type of publication in which the document was published. This should be one of the PRISM aggregation types [8] such as <code>book</code>, <code>journal</code>, <code>magazine</code>, <code>manual</code>, <code>report</code>, or <code>whitepaper</code>. For publications in journals, magazines, and similar periodicals, a document can specify the volume number with <code>pdfvolumenum</code> and the issue number within the volume with <code>pdfissuenum</code>.</p>
pdfvolumenum pdfissuenum pdfpagerange	<p><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. 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>.</p>
pdfbookedition	
pdfnumpages	<p>The number of pages in the published, print version of the document can be expressed with <code>pdfnumpages</code>. Note 9 on page 16 explains how to automatically assign a value to <code>pdfnumpages</code>.</p>
pdfdate	<p>XMP metadata can include a number of dates (in fact, timestamps, as they include both date and time components). <code>pdfdate</code> specifies the document date. It is analogous to the <math>\text{\LaTeX}</math> <code>\date</code> command, and, like <code>\date</code>, defaults to the date the document was built. It must be specified in either XMP format [4] or PDF format [3]. XMP dates are written in the form <code>YYYY-MM-DDThh:mm:ss+TT:tt</code>.<sup>2</sup> A W3C recommendation [14] discusses this format in more detail, but as an example, 14 hours, 15 minutes, 9 seconds past midnight U.S. Mountain Daylight Time (UTC-6) on the 23rd day of September in the year 2014 should be written as <code>2014-09-23T14:15:09-06:00</code>. This can be truncated (with loss of information) to <code>2014-09-23T14:15:09</code>, <code>2014-09-23T14:15</code>, <code>2014-09-23</code>, <code>2014-09</code>, or <code>2014</code> but no other subsets. PDF dates are written in the form <code>D:YYYYMMDDhhmmss+TT'tt'</code>. The same date in the preceding example would be written as <code>D:20140923141509-06'00'</code> in PDF format.</p>
pdfcreationdate pdfmoddate pdfmetadate	<p>The document's creation date, modification date, and metadata date are normally set automatically, but <code>pdfcreationdate</code>, <code>pdfmoddate</code>, and <code>pdfmetadate</code> can be used to override the defaults. Like <code>pdfdate</code>, <code>pdfmetadate</code> can be specified in either XMP or PDF format. However, because <code>hyperref</code> defines <code>pdfcreationdate</code> and <code>pdfmoddate</code> and expects these to be written as PDF dates, <code>hyperxmp</code> concomitantly</p>

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

	accepts these two dates only in PDF format as well. Note that it's rare that a document would need to specify any of <code>pdfcreationdate</code> , <code>pdfmoddate</code> , or <code>pdfmetadate</code> .
<code>pdftype</code>	<code>pdftype</code> describes the type of document being produced. This refers to “the nature or genre of the resource” [4] such as <code>poem</code> , <code>novel</code> or <code>working paper</code> , as opposed to the file format (always <code>application/pdf</code> when generated by <code>hyperxmp</code> ). Although <code>pdftype</code> can be assigned an arbitrary piece of text, the XMP specification recommends selecting types from a “controlled vocabulary” such as the DCMI Type Vocabulary [5]. The DCMI Type Vocabulary currently consists of only <code>Collection</code> , <code>Dataset</code> , <code>Event</code> , <code>Image</code> , <code>InteractiveResource</code> , <code>MovingImage</code> , <code>PhysicalObject</code> , <code>Service</code> , <code>Software</code> , <code>Sound</code> , <code>StillImage</code> , and <code>Text</code> . <code>pdftype</code> defaults to <code>Text</code> , which refers to “books, letters, dissertations, poems, newspapers, articles, archives of mailing lists,” [5] and other forms of text—all things L <sup>A</sup> T <sub>E</sub> X is commonly used to typeset.
<code>pdfrendition</code>	Sometimes a base document is rendered in different forms. <code>pdfrendition</code> indicates the particular rendition the current document instance represents. The value should come from the following controlled vocabulary [4]: <code>default</code> , <code>draft</code> , <code>low-res</code> , <code>proof</code> , <code>screen</code> , and <code>thumbnail</code> . <code>hyperxmp</code> 's default value is <code>default</code> , which indicates the master document, unless the <code>draft</code> option is passed to <code>\documentclass</code> , in which case <code>hyperxmp</code> defaults to <code>draft</code> .
<code>pdfbytes</code>	The <code>pdfbytes</code> option expresses the document's file size in bytes. The intention is for this to be used to display an estimate of download time to a user or to serve as a quick check on whether a file was transmitted correctly between systems. This feature is easiest to use in conjunction with pdfT <sub>E</sub> X's <code>\pdffilesizesize</code> primitive: “ <code>pdfbytes={\pdffilesizesize{\jobname.pdf}}</code> ”. Note that this requires a second run of <code>pdftex</code> because it queries the size of the PDF file from the <i>previous</i> run.
<code>pdftrapped</code>	<code>hyperxmp</code> honors <code>hyperref</code> 's <code>pdftrapped</code> option. A document can indicate whether it employs color trapping by specifying <code>pdftrapped=True</code> or <code>pdftrapped=False</code> . ( <code>pdftrapped=Unknown</code> is also allowed.) A current limitation of <code>hyperxmp</code> is that if a value other than <code>False</code> is provided, a document will additionally need to specify <code>keeppdfinfo</code> (page 13) to ensure that the PDF Info dictionary specifies the correct trapping value.
<code>pdfapart</code> <code>pdfaconformance</code>	<code>pdfapart</code> and <code>pdfaconformance</code> , are used in conjunction with <code>hyperref</code> 's <code>pdfa</code> option to claim a particular PDF/A standard by which the document abides. They default to <code>pdfapart=1</code> and <code>pdfaconformance=B</code> , 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 <code>pdfuapart</code> to indicate the PDF/UA conformance level. For example, <code>pdfuapart=1</code> asserts that the document respects PDF/UA-1. <code>pdfxstandard</code> indicates the particular PDF/X standard by which the document abides. Unlike <code>pdfapart</code> and <code>pdfaconformance</code> , which accept a number and a letter, respectively, <code>pdfxstandard</code> expects a textual identification of a standard name. The following are the PDF/X standard names that are considered acceptable at the time of this writing.
<code>pdfuapart</code> <code>pdfxstandard</code>	



- 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.

`pdfsource` A rarely needed option, `pdfsource`, overrides the name of the  $\text{\LaTeX}$  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.

It is usually more convenient to provide values for the preceding options using `hyperref`'s `\hypersetup` command than on the `\usepackage` command line. See the `hyperref` manual for more information.

## 2.2 A complete example

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

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

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

\hypersetup{%
  pdftitle={%
    On a heuristic viewpoint concerning the production and
    transformation of light},
  pdfsubtitle={[[en-US]Putting that bum Maxwell in his place]},
  pdfauthor={Albert Einstein},
  pdfauthortitle={\xmpquote{Technical Assistant\xmpcomma\ Level III}},
  pdfdate={1905-03-17},
  pdfcopyright={Copyright (C) 1905, Albert Einstein},
  pdfsubject={photoelectric effect},
  pdfkeywords={energy quanta, Hertz effect, quantum physics},
  pdflicenseurl={http://creativecommons.org/licenses/by-nc-nd/3.0/},
  pdfcaptionwriter={Scott Pakin},
  pdfcontactaddress={Kramgasse 49},
  pdfcontactcity={Bern},
  pdfcontactpostcode={3011},
```

```

pdfcontactcountry={Switzerland},
pdfcontactphone={031 312 00 91},
pdfcontactemail={aeinstein@ipi.ch},
pdfcontacturl={%
    http://einstein.biz/,
    https://www.facebook.com/AlbertEinstein
},
pdfdocumentid={uuid:6d1ac9ec-4ff2-515a-954b-648eeb4853b0},
pdfversionid={2.998e8},
pdfpublication={[de]Annalen der Physik},
pdfpublisher={Wiley-VCH},
pdfpubtype={journal},
pdfvolumenum={322},
pdfissuenum={6},
pdfpagerange={132-148},
pdfnumpages={17},
pdfissn={0003-3804},
pdfeissn={1521-3889},
pdflang={en},
pdfmetalang={en},
pdfurl={http://www.physik.uni-augsburg.de/annalen/history/einstein-papers/1905_17_132-148.pdf},
pdfdoi={10.1002/andp.19053220607},
pdfidentifier={info:lccn/50013519},
pdfbytes={\pdffilesize{\jobname.pdf}} % Requires pdflatex
}
\XMLLangAlt{de}{pdftitle={Über einen die Erzeugung und Verwandlung des
    Lichtes betreffenden heuristischen Gesichtspunkt}}

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

```

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

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

Unfortunately, the  $\LaTeX$  + Dvips + Ghostscript path doesn't work. Ghostscript bug report #690066, closed with "WONTFIX" status on 2012-05-28, explains that

Ghostscript doesn't honor the `Metadata` tag needed to inject a custom XMP packet. Instead, Ghostscript fabricates an XMP packet of its own based on the metadata it finds in the PDF file's Info dictionary (Author, Title, Subject, and Keywords).

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

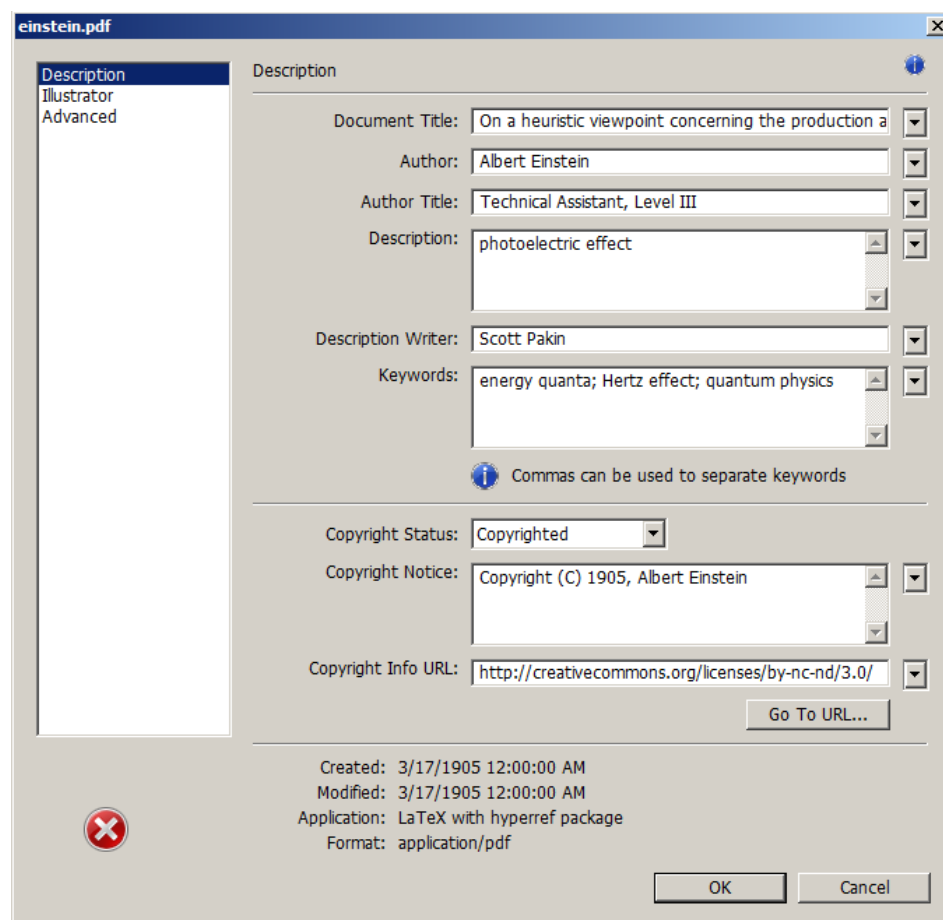


Figure 1: XMP metadata as it appears in Adobe Acrobat

## 2.3 Usage notes

**Note 1: Conflicting metadata in PDF/A documents** A PDF file includes an Info dictionary containing Author, Title, Subject, and Keywords keys. The

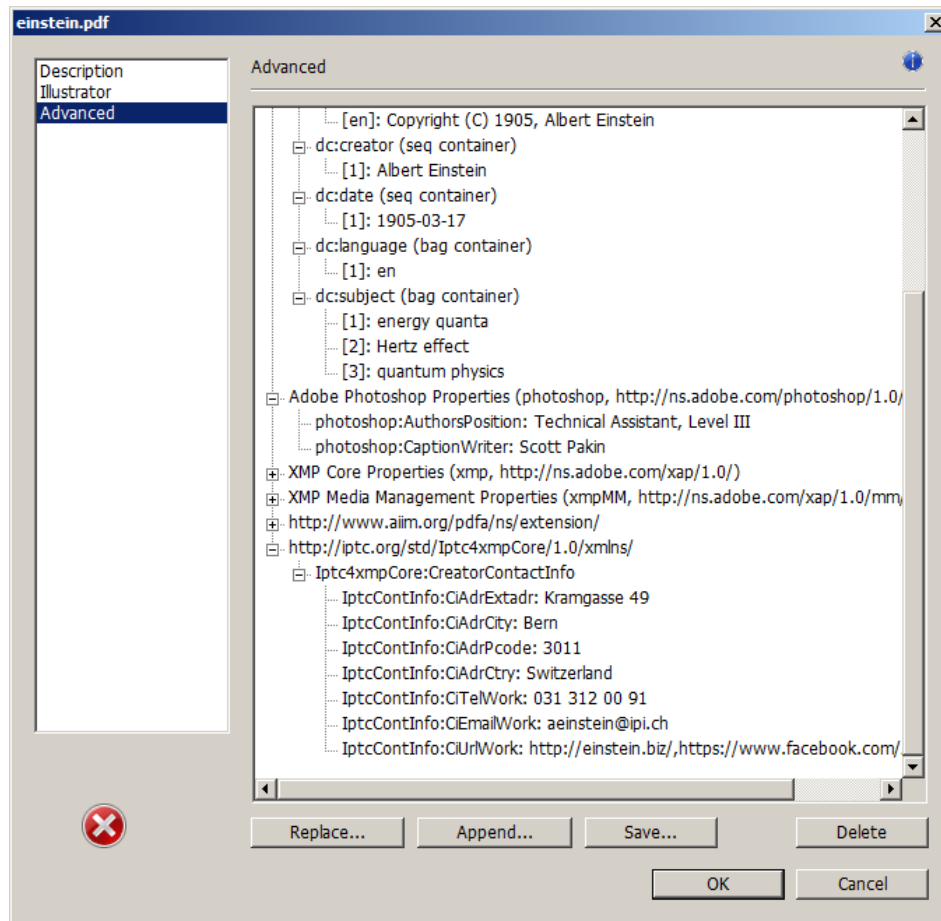


Figure 2: Additional XMP metadata as it appears in Adobe Acrobat

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

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

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

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

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

```
\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 Lua $\text{\LaTeX}$  earlier than 0.85, `hyperxmp` leaves all PDF objects uncompressed. This is due to Lua $\text{\LaTeX}$  treating object compression as a global parameter, unlike pdf $\text{\LaTeX}$ , which treats it as a local parameter. Hence, when `hyperxmp` requests that the XMP packet be left uncompressed, Lua $\text{\LaTeX}$  in fact leaves *all* PDF streams uncompressed. Beginning with version 3.0, `hyperxmp` includes a workaround that correctly leaves only the XMP metadata uncompressed, but this workaround is implemented only for Lua $\text{\LaTeX}$  v0.85 onwards.
2. X $\text{\LaTeX}$  (or, more precisely, the `xdvipdfmx` back end) exhibits the opposite problem. It compresses *all* PDF objects, including the ones containing XMP metadata. While Adobe Acrobat can still detect and utilize the XMP metadata, non-PDF-aware applications are unlikely to see the metadata. Three

options to consider are to (1) use a different program (e.g., Lua<sub>La</sub>T<sub>E</sub>X), (2) pass the `--output-driver="xdvipdfmx -z0"` option to X<sub>La</sub>T<sub>E</sub>X to instruct `xdvipdfmx` to turn off all compression (which will of course make the PDF file substantially larger), or (3) postprocess the generated PDF file by loading it into the commercial version of Adobe Acrobat and re-saving it with the Save As... menu option.

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

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

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

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

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

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

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

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

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

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

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

**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`

```
\XMPLangAlt {⟨language⟩} { ⟨option⟩=⟨text⟩, ... }
```

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

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

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

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

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

Rather, it generates a `dc:rights` tag of the form “Copyright © =2=0=by-1by=02020, Scott Pakin”. The garbage in that line corresponds to the remnants of the `\printfddate` code after expanding all of the  $\text{\TeX}$  primitives and certain other control sequences it uses to the empty string. For example, “`\global\advance\texd@yr by-1`” expands to “by-1”.

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

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

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

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

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



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

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

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

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

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

## 3 Implementation

This section presents the commented L<sup>A</sup>T<sub>E</sub>X source code for `hyperxmp`. Read this section only if you want to learn how `hyperxmp` is implemented.

### 3.1 Initial preparation

`\hyxmp@dq@code` The `ngerman` package redefines “ ” as an active character, which causes problems for `hyperxmp` when it tries to use that character. We therefore save the double-quote character's current category code in `\hyxmp@dq@code` and mark the character as category code 12 (“other”). The original category code is restored at the end of the package code (Section 3.7).

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

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

```

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

```

### 3.2 Integration with hyperref

An important design decision underlying `hyperxmp` is that the package should integrate seamlessly with `hyperref`. To that end, `hyperxmp` takes XMP metadata from `hyperref`'s `baseurl`, `pdfauthor`, `pdfkeywords`, `pdflang`, `pdfproducer`, `pdfsubject`, `pdftrapped`, and `pdftitle` options. It also introduces a number of new options, which are listed on pages 4–5. For consistency with `hyperref`'s document-metadata naming conventions (which are in turn based on  $\text{\LaTeX}$ '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  $\text{\TeX}$  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.

```

9 \RequirePackage{kvoptions}
10 \RequirePackage{pdfescape}
11 \RequirePackage{stringenc}
12 \RequirePackage{intcalc}
13 \RequirePackage{iftex}
14 \RequirePackage{ifmtarg}
15 \RequirePackage{etoolbox}
16 \RequirePackage{ifthen}

```

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

```

17 \def\ifmtargexp#1{\expandafter\ifmtarg\expandafter{#1}}
18 \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`.

```

19 \newcommand*\if@def@and@nonempty[3]{%
20   \ifundefined{#1}{#3}{%

```

```

21 \expandafter\@ifmtargexp\expandafter{\csname#1\endcsname}{#3}{#2}%
22 }%
23 }

```

`\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.

```

24 \newcommand{\hyxmp@pdfstringdef}[2]{%
25 \let\hyxmp@textunderscore=\textunderscore
26 \let\textunderscore=\hyxmp@uscore
27 \pdfstringdef{#1}{#2}%
28 \let\textunderscore=\hyxmp@textunderscore
29 }

```

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

```

30 \def\@pdfdatetime{}
31 \define@key{Hyp}{pdfdate}{%
32 \begingroup
33 \Hy@unicodetofalse

```

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

```

34 \edef\next{%
35 \noexpand\hyxmp@pdfstringdef\noexpand\@pdfdatetime{%
36 \noexpand\hyxmp@as@xmp@date{#1}}}%
37 }%
38 \next
39 \endgroup
40 }

```

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

```

41 \def\@pdfmetadatettime{}
42 \define@key{Hyp}{pdfmetadate}{%
43 \begingroup
44 \Hy@unicodetofalse

```

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

```

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

```

```

50 \endgroup
51 }

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

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

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

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

\@pdfcaptionwriter Prepare to store the name of the person who inserted the hyperxmp metadata.
60 \def\@pdfcaptionwriter{}
61 \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.
62 \def\@pdfmetalang{}
63 \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.
64 \def\hyxmp@no@bad@parts#1\relax{%
65   \ifnotmtarg{#1}{%
66     \PackageWarning{hyperxmp}{pdfapart and pdfuapart must be numeric}%
67   }%
68 }

\@pdfapart Prepare to store the PDF/A part ID, which defaults to "1" if pdfa is passed to
hyperref.
69 \def\@pdfapart{}
70 \define@key{Hyp}{pdfapart}{%
71   \afterassignment\hyxmp@no@bad@parts\@tempcnta=0#1\relax
72   \hyxmp@pdfstringdef\@pdfapart{\the\@tempcnta}%
73 }

\@pdfaconformance Prepare to store the PDF/A conformance ID, which defaults to "b" if pdfa is passed
to hyperref and \@pdfapart is empty.
74 \def\@pdfaconformance{}
75 \define@key{Hyp}{pdfaconformance}{%
76   \uppercase{\hyxmp@pdfstringdef\@pdfaconformance{#1}}%
77 }

```

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

```

78 \def\@pdfuapart{}
79 \define@key{Hyp}{pdfuapart}{%
80   \afterassignment\hyxmp@no@bad@parts\@tempcnta=0#1\relax
81   \hyxmp@pdfstringdef\@pdfuapart{\the\@tempcnta}%
82 }

```

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

```

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

```

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

```

84 \def\hyxmp@set@pdfx@major@i PDF/X-{%
85   \afterassignment\hyxmp@set@pdfx@major@ii
86   \@tempcnta=%
87 }

```

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

```

88 \def\hyxmp@set@pdfx@major@ii#1!{%
89   \edef\hyxmp@pdfx@major{\the\@tempcnta}%
90 }

```

`\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.

```

91 \newcommand*\hyxmp@check@std[2]{%
92   \ifthenelse{\equal{#1}{#2}}{%
93     {\global\let\next=\relax}%
94     {}%
95 }%

```

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

```

96 \def\@pdfxstandard{}
97 \def\hyxmp@pdfx@major{}
98 \define@key{Hyp}{pdfxstandard}{%
99   \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 65 above the definition of `\hyxmp@pdfx@id@schema`, for example.)

```

100 \gdef\next{%
101   \PackageWarning{hyperxmp}{Unrecognized PDF/X standard ‘#1’}%
102 }%

```

```

103 \hyxmp@check@std{#1}{PDF/X-1a:2001}%
104 \hyxmp@check@std{#1}{PDF/X-1a:2003}%
105 \hyxmp@check@std{#1}{PDF/X-3:2002}%
106 \hyxmp@check@std{#1}{PDF/X-3:2003}%
107 \hyxmp@check@std{#1}{PDF/X-4}%
108 \hyxmp@check@std{#1}{PDF/X-4p}%
109 \hyxmp@check@std{#1}{PDF/X-5g}%
110 \hyxmp@check@std{#1}{PDF/X-5n}%
111 \hyxmp@check@std{#1}{PDF/X-5pg}%
112 \next

\hyxmp@pdfx@major Parse the PDF/X major version number from pdfxstandard and assign it to
\hyxmp@pdfx@major.
113 \hyxmp@set@pdfx@major{#1}%
114 }

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

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

\hyxmp@InstanceID Prepare to store a UUID that represents the current instance of the document.
119 \def\hyxmp@InstanceID{}
120 \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".
121 \def\@pdfversionid{1}
122 \define@key{Hyp}{pdfversionid}{\hyxmp@pdfstringdef\@pdfversionid{#1}}

\ifdraft Use the ifdraft package to determine if this is a draft or final document.
123 \begingroup
124 \let\ifdraft=\relax
125 \RequirePackage{ifdraft}

\@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.
126 \ifdraft{%
127 \gdef\@pdfrendition{draft}%
128 }{%
129 \gdef\@pdfrendition{default}%
130 }
131 \endgroup
132 \define@key{Hyp}{pdfrendition}{\hyxmp@pdfstringdef\@pdfrendition{#1}}

```

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

```

133 \def\@pdfpublication{}
134 \define@key{Hyp}{pdfpublication}{\hyxmp@pdfstringdef\@pdfpublication{#1}}

```

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

```

135 \def\@pdfpubtype{}
136 \define@key{Hyp}{pdfpubtype}{\hyxmp@pdfstringdef\@pdfpubtype{#1}}

```

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

```

137 \def\@pdfbytes{}
138 \define@key{Hyp}{pdfbytes}{\hyxmp@pdfstringdef\@pdfbytes{#1}}

```

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

```

139 \def\@pdfnumpages{}
140 \define@key{Hyp}{pdfnumpages}{\hyxmp@pdfstringdef\@pdfnumpages{#1}}

```

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

```

141 \def\@pdfissn{}
142 \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.

```

143 \def\@pdfeissn{}
144 \define@key{Hyp}{pdfeissn}{\hyxmp@pdfstringdef\@pdfeissn{#1}}

```

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

```

145 \def\@pdfisbn{}
146 \define@key{Hyp}{pdfisbn}{\hyxmp@pdfstringdef\@pdfisbn{#1}}

```

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

```

147 \def\@pdfbookedition{}
148 \define@key{Hyp}{pdfbookedition}{\hyxmp@pdfstringdef\@pdfbookedition{#1}}

```

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

```

149 \def\@pdfpublisher{}
150 \define@key{Hyp}{pdfpublisher}{\hyxmp@pdfstringdef\@pdfpublisher{#1}}

```

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

```

151 \def\@pdfvolumenum{}
152 \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.

```

153 \def\@pdfissuenum{}
154 \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.

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

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

```
157 \def\@pdfdoi{}
158 \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.

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

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

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

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

```
163 \def\@pdfsubtitle{}
164 \define@key{Hyp}{pdfsubtitle}{\hyxmp@pdfstringdef\@pdfsubtitle{#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.

```
165 \def\@pdfcontactaddress{}
166 \define@key{Hyp}{pdfcontactaddress}{%
167   \let\xmpcomma=\hyxmp@comma
168   \def\xmpquote##1{##1}%
169   \hyxmp@pdfstringdef\@pdfcontactaddress{#1}%
170   \def\xmpcomma{,}%
171   \let\xmpquote=\relax
172 }
```

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

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



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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

185 \def\@pdfcontacturl{}
186 \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`.

```

187 \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.

```

\next
188 \def\hyxmp@suppress@pdf@info{%
189   \global\let\PDF@FinishDoc=\@empty
190   \PackageWarningNoLine{hyperxmp}{%
191     Suppressing the _entire_ PDF Info dictionary.\MessageBreak
192     Please notify the hyperxmp maintainer%
193   }%
194 }%
195 \let\next=\relax
196 \patchcmd
197   {\PDF@FinishDoc}%
198   {\Author(\@pdfauthor)}%
199   {}%
200   {}%
201   {\let\next=\hyxmp@suppress@pdf@info}%
202 \patchcmd

```

```

203     {\PDF@FinishDoc}%
204     {/Keywords(\@pdfkeywords)}}%
205     {}%
206     {}%
207     {\let\next=\hyxmp@suppress@pdf@info}%
208     \next
209 }

210 \define@key{Hyp}{keeppdfinfo}[true]{%
211     \gdef\hyxmp@no@info@lists{}}%
212 }

```

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

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

```

\hyxmp@pdfauthor    Prepare to store the name of the author and a list of keywords.
\hyxmp@pdfkeywords  213 \def\hyxmp@pdfauthor{}
                    214 \def\hyxmp@pdfkeywords{}

\hyxmp@redefine@Hyp If not already redefined, redefine hyperref's pdfauthor and pdfkeywords options to
                    properly handle \xmpcomma and \xmpquote.
                    215 \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.
                    216 \@ifundefined{KV@Hyp@pdfauthor}{}{%
                    217     \@ifundefined{hyxmp@Hyp@pdfauthor}{%
                    218         \expandafter\let\expandafter\hyxmp@Hyp@pdfauthor
                    219             \csname KV@Hyp@pdfauthor\endcsname
                    220     }{}%
                    221 }%

\KV@Hyp@pdfauthor    Redefine \KV@Hyp@pdfauthor to process its argument twice. The first time,
\hyxmp@pdfauthor     \xmpcomma is defined as a placeholder character (\hyxmp@comma) and \xmpquote
\hyxmp@pdfauthor     \xmpquote as the identity function. The result is stored in \hyxmp@pdfauthor for use in
\hyxmp@pdfauthor     \hyxmp@and
\hyxmp@pdfauthor     \and
\hyxmp@pdfauthor     \hyxmp@pdfauthor
\hyxmp@pdfauthor     \@pdfauthor

```

structured lists (those surrounding each entry with `<rdf:li>`). The second time, `\xmpcomma` is defined as an ordinary comma, and `\xmpquote` is defined as a macro that puts its argument within double quotes. The result is stored in `\@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.

```

222 \define@key{Hyp}{pdfauthor}{%
223   \let\xmpcomma=\hyxmp@comma
224   \def\xmpquote####1{####1}%
225   \let\hyxmp@and=\and
226   \def\and{,}%
227   \hyxmp@Hyp@pdfauthor{##1}%
228   \global\let\hyxmp@pdfauthor=\@pdfauthor
229   \def\and{and\space}%
230   \def\xmpcomma{,}%
231   \def\xmpquote####1{"####1"%
232   \hyxmp@Hyp@pdfauthor{##1}%
233   \def\xmpcomma{,}%
234   \let\xmpquote=\relax
235   \let\and=\hyxmp@and
236 }%
```

`\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`.

```

237 \@ifundefined{KV@Hyp@pdfkeywords}{-}{%
238   \@ifundefined{hyxmp@Hyp@pdfkeywords}{-}{%
239     \expandafter\let\expandafter\hyxmp@Hyp@pdfkeywords
240     \csname KV@Hyp@pdfkeywords\endcsname
241   }{}%
242 }%
```

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

```

243 \define@key{Hyp}{pdfkeywords}{%
244   \let\xmpcomma=\hyxmp@comma
245   \def\xmpquote####1{####1}%
246   \hyxmp@Hyp@pdfkeywords{##1}%
247   \global\let\hyxmp@pdfkeywords=\@pdfkeywords
248   \def\xmpcomma{,}%
249   \def\xmpquote####1{"####1"%
250   \hyxmp@Hyp@pdfkeywords{##1}%
```

```

251     \def\xmpcomma{,%
252     \let\xmpquote=\relax
253     }%
254 }

```

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

```

255 \let\hyxmp@ProcessKeyvalOptions=\ProcessKeyvalOptions
256 \renewcommand*{\ProcessKeyvalOptions}{%
257   \hyxmp@redefine@Hyp
258   \hyxmp@ProcessKeyvalOptions
259 }

```

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

```

260 \let\hyxmp@hypersetup=\hypersetup
261 \def\hypersetup{%
262   \hyxmp@redefine@Hyp
263   \hyxmp@hypersetup
264 }

```

`\hyxmp@find@metadata`   Issue a warning message if the author failed to specify any metadata at all. This excludes metadata that is included automatically such as the current timestamp. Note that we don't consider `\@pdfmetalang` as metadata as that value is meaningful only when used in conjunction with other information. We also don't examine `\@pdfapart` or `\@pdfaconformance` because those have nonempty default values.

`\hyxmp@concat@metadata`

```

265 \newcommand*{\hyxmp@find@metadata}{%
266   \edef\hyxmp@concat@metadata{%
267     \@baseurl
268     \@pdfauthor
269     \@pdfauthortitle
270     \@pdfbookedition
271     \@pdfbytes
272     \@pdfcaptionwriter
273     \@pdfcontactaddress
274     \@pdfcontactcity
275     \@pdfcontactcountry
276     \@pdfcontactemail
277     \@pdfcontactphone
278     \@pdfcontactpostcode
279     \@pdfcontactregion
280     \@pdfcontacturl
281     \@pdfcopyright
282     \@pdfcreationdate
283     \@pdfdatetime
284     \@pdfdoi
285     \@pdfeissn
286     \@pdfidentifier
287     \@pdfisbn

```

```

288 \pdfissn
289 \pdfissuenum
290 \pdfkeywords
291 \pdflang
292 \pdflicenseurl
293 \pdfmetadatetitle
294 \pdfmoddate
295 \pdfnumpages
296 \pdfpagerange
297 \pdfpublication
298 \pdfpubtype
299 \pdfsubject
300 \pdfsubtitle
301 \pdftitle
302 \pdfuapart
303 \pdfurl
304 \pdfvolumenum
305 \pdfxstandard
306 }%
307 \ifx\hyxmp@concat@metadata\empty
308 \PackageWarningNoLine{hyperxmp}{%
309 \jobname.tex did not specify any metadata to\MessageBreak
310 include in the XMP packet.\space\space Please see the\MessageBreak
311 hyperxmp documentation for instructions on how to\MessageBreak
312 provide metadata values to hyperxmp}%
313 \fi
314 }

```

`\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.

```

315 \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.
316 \ifHy@pdfa
317 \ifmtargexp{\pdfapart}{%
318 \PackageWarningNoLine{hyperxmp}{%
319 'pdfa' was passed to hyperref, but 'pdfapart' was\MessageBreak
320 not specified.\space\space Setting pdfapart to '1' and\MessageBreak
321 pdfaconformance to 'B'%
322 }%
323 \gdef\pdfapart{1}%
324 \gdef\pdfaconformance{B}%
325 }%
326 {}%
327 \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).

```

328 \edef\hyxmp@standards{%
329   \@pdfapart
330   \@pdfxstandard
331   \@pdfuapart
332 }%

```

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

```

333 \ifnotmtargexp{\hyxmp@standards}{%
334   \ifmtargexp{\@pdftitle}{%
335     \PackageWarningNoLine{hyperxmp}{%
336       Missing pdftitle (required for PDF standards\MessageBreak
337       compliance)%
338     }%
339   }%
340 }%
341 }%
342 }

```

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

```

343 \newcommand*{\hyxmp@set@koma@phones}{%
344   \if@def@and@nonempty{\scr@frommobilephone@var}{%
345     \if@def@and@nonempty{\scr@fromphone@var}{%
346       \edef\hyxmp@koma@phones{\scr@frommobilephone@var,\scr@fromphone@var}%
347     }{%
348       \edef\hyxmp@koma@phones{\scr@frommobilephone@var}%
349     }%
350 }{%
351   \if@def@and@nonempty{\scr@fromphone@var}{%
352     \edef\hyxmp@koma@phones{\scr@fromphone@var}%
353   }{%
354   }%
355 }%
356 }

```

`\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.

```

357 \newcommand*{\hyxmp@use@first@valid}[3]{%
358   \ifmtargexp{#2}{%
359     \hyxmp@use@first@valid@i{#1}#3,!,%
360   }%
361   {}%
362 }

```

`\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`.

```

363 \def\hyxmp@use@first@valid@i#1#2,{%
364   \def\next{\hyxmp@use@first@valid@i{#1}}%
365   \ifx#2!%
366     \let\next=\relax
367   \else
368     \ifx#2\undefined
369     \else
370       \@ifnotmtargexp{#2}{%
371         \hypersetup{#1={#2}}%
372         \def\next##1!,{}%
373       }%
374     \fi
375   \fi
376   \next
377 }

```

`\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).

```

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

```

If the author explicitly specified the language to use for the document's metadata, we use that. If not, we use the document language, specified to `hyperref` with the `pdflang` option. If the author did not specify a language, we use `x-default` as the metadata language.

```

379   \ifx\@pdfmetalang\@empty
380     \ifx\@pdflang\@empty
381       \let\@pdfmetalang=\hyxmp@x@default
382     \else
383       \edef\@pdfmetalang{\@pdflang}%
384     \fi
385   \fi
386   \hyxmp@xmllify\@pdfmetalang

```

If the author left `pdftitle` blank but specified `\title`, use the title for `pdftitle`. Do likewise for various other metadata: identify author-provided information that can be co-opted for use as XMP metadata.

```

387   \hyxmp@use@first@valid{pdftitle}{\@pdftitle}{%
388     \scr@subject@var,%
389     \@title
390   }%
391   \hyxmp@use@first@valid{pdfauthor}{\@pdfauthor}{%
392     \scr@fromname@var,%
393     \@author
394   }%
395   \hyxmp@use@first@valid{pdfcontactemail}{\@pdfcontactemail}{%
396     \scr@fromemail@var
397   }%
398   \hyxmp@set@koma@phones
399   \hyxmp@use@first@valid{pdfcontactphone}{\@pdfcontactphone}{%
400     \hyxmp@koma@phones

```

```

401 }%
402 \hyxmp@use@first@valid{pdfcontacturl}{\@pdfcontacturl}{%
403   \scr@fromurl@var
404 }%
405 \hyxmp@use@first@valid{pdfsubtitle}{\@pdfsubtitle}{%
406   \@subtitle
407 }%
408 \hyxmp@use@first@valid{pdfpublisher}{\@pdfpublisher}{%
409   \@publishers
410 }%

```

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.

```

411 \hyxmp@parse@acmart
412 }

```

`\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.

```

413 \newcommand*{\hyxmp@parse@acmart}{%
414   \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.

```

415   \let\@author=\@gobble

```

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

```

\hyxmp@address@val
416   \def\email##1##2{%
417     \def\hyxmp@address@val{##2}%
418     \hyxmp@use@first@valid{pdfcontactemail}{\@pdfcontactemail}{%
419       \hyxmp@address@val
420     }%
421   }%

```

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

```

\hyxmp@address@val
422   \def\streetaddress##1{%
423     \def\hyxmp@address@val{##1}%
424     \hyxmp@use@first@valid{pdfcontactaddress}{\@pdfcontactaddress}{%
425       \hyxmp@address@val
426     }%
427   }%

```

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

```

\hyxmp@address@val
428   \def\city##1{%
429     \def\hyxmp@address@val{##1}%

```



```

430      \hyxmp@use@first@valid{pdfcontactcity}{\@pdfcontactcity}{%
431      \hyxmp@address@val
432      }%
433      }%

\state \state wraps the author's state or region name.
\hyxmp@address@val 434      \def\state##1{%
435      \def\hyxmp@address@val{##1}%
436      \hyxmp@use@first@valid{pdfcontactregion}{\@pdfcontactregion}{%
437      \hyxmp@address@val
438      }%
439      }%

\country \country wraps the author's country name.
\hyxmp@address@val 440      \def\country##1{%
441      \def\hyxmp@address@val{##1}%
442      \hyxmp@use@first@valid{pdfcontactcountry}{\@pdfcontactcountry}{%
443      \hyxmp@address@val
444      }%
445      }%

\postcode \postcode wraps the author's postal code.
\hyxmp@address@val 446      \def\postcode##1{%
447      \def\hyxmp@address@val{##1}%
448      \hyxmp@use@first@valid{pdfcontactpostcode}{\@pdfcontactpostcode}{%
449      \hyxmp@address@val
450      }%
451      }%

\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.

452      \def\affiliation##1##2{%
453      ##2%
454      \let\affiliation=\@gobbletwo
455      }%

```

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.

```

456      \setbox0=\hbox{\addresses}%
457      \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.

```

458 \if@def@and@nonempty{@acmDOI}{%
459   \IfSubStr{\@acmDOI}{10.1145/1122445.1122456}{-}{%
460     \IfSubStr{\@acmDOI}{10.1145/nnnnnnn.nnnnnnn}{-}{%
461       \hyxmp@use@first@valid{pdfdoi}{\@pdfdoi}{%
462         \@acmDOI
463       }%
464     }%
465   }%
466 }%
```

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

```

467 \if@def@and@nonempty{@acmISBN}{%
468   \IfSubStr{\@acmISBN}{XXXX}{-}{%
469     \IfSubStr{\@acmISBN}{xxxx}{-}{%
470       \def\hyxmp@strip@isbn@date##1/##2!{##1}%
471       \edef\hyxmp@acm@isbn{%
472         \expandafter\hyxmp@strip@isbn@date\@acmISBN/!%
473       }%
474       \hyxmp@use@first@valid{pdfisbn}{\@pdfisbn}{%
475         \hyxmp@acm@isbn
476       }%
477     }%
478   }%
479 }%
```

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

```

480 \def\hyxmp@acm@publisher{Association for Computing Machinery}%
481 \hyxmp@use@first@valid{pdfpublisher}{\@pdfpublisher}{%
482   \hyxmp@acm@publisher
483 }%
```

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

```

484 \hyxmp@use@first@valid{pdfpublication}{\@pdfpublication}{%
485   \@journalName,%
486   \@acmBooktitle,%
487   \@acmConference
488 }%
```

`\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).

```

489 \if@ACM@journal
490   \def\hyxmp@acm@pubtype{journal}%
```

```

491 \else
492   \def\hyxmp@acm@pubtype{book}%
493 \fi
494 \hyxmp@use@first@valid{pdfpubtype}{\@pdfpubtype}{%
495   \hyxmp@acm@pubtype
496 }%

```

Journal articles have a volume and issue number.

```

497 \hyxmp@use@first@valid{pdfvolumenum}{\@pdfvolumenum}{%
498   \@acmVolume
499 }%
500 \hyxmp@use@first@valid{pdfissuenum}{\@pdfissuenum}{%
501   \@acmNumber
502 }%
503 }

```

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

```

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

```

`\hyxmp@set@dc@lang` `\hyxmp@dc@lang` `\@pdflang` is used in both the PDF document catalog (the `Lang` key, written to the PDF file by `hyperref`) and in the Dublin Core `dc:language` tag (Section 3.5.3). Normally, these are the same. However, `Lang` accepts only a single language while `dc:language` accepts multiple languages. If the document loads `polyglossia` and does not specify `pdflang` in `\hypersetup`, `\hyxmp@set@dc@lang` is redefined below to set `\hyxmp@dc@lang` to `polyglossia`'s list of used languages (once this is known). Otherwise, `\hyxmp@set@dc@lang` assigns the language specified by `pdflang` (if any) to `\hyxmp@dc@lang`.

```

505 \newcommand*{\hyxmp@set@dc@lang}{%
506   \let\hyxmp@dc@lang=\@pdflang
507 }

```

Rather than load `hyperref` ourselves we let the author do it then verify he actually did. This approach gives the author the flexibility to load `hyperxmp` and `hyperref` in either order and to call `\hypersetup` anywhere in the document's preamble, not just before `hyperxmp` is loaded.

```

508 \AtEndPreamble{%
509   \@ifpackageloaded{hyperref}{%

```

In older versions of `hyperref`, `\@pdflang` is set to `\@empty` if `pdflang` is not specified. In newer versions of `hyperref`, `\@pdflang` is set to `\relax` if `pdflang` is not specified. The latter is a bit problematic for `hyperxmp` because it makes `\@pdflang` non-expandable, which causes a literal “`\@pdflang`” to be written as XMP metadata. To avoid that situation we redefine `\@pdflang` as `\@empty` if we see it set to `\relax`. Actually, we first check if the `polyglossia` package has specified the document's main language in `\mainbcp47id`. If so, we use that instead of `\@empty` and redefine `\hyxmp@set@dc@lang` to consider `polyglossia`'s list of used languages.

```

510   \ifx\@pdflang\relax
511     \@ifundefined{mainbcp47id}{%

```

```

512     \let\@pdflang=\@empty
513   }{%
514     \xdef\@pdflang{\csname mainbcp47id\endcsname}%
515     \renewcommand*{\hyxmp@set@dc@lang}{%
516       \edef\hyxmp@dc@lang{\xpg@bcp@loaded}%
517     }%
518   }%
519   \fi

```

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

```

520   \hyxmp@auto@assign@data

```

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

```

521   \@ifundefined{XeTeXversion}{-}{%
522     \@ifmtargexp{\@pdfcreationdate}{%
523       \let\@pdfcreationdate=\hyxmp@today@pdf
524     }%
525   }%
526   }%

```

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

```

527   \hyxmp@check@standards

```

Older versions of `hyperref` write the Info dictionary to the PDF file at the end of the document. New 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.

```

528   \hyxmp@no@info@lists

```

We wait until the end of the document to construct the XMP packet and write it to the PDF document catalog. This gives the author ample opportunity to provide metadata to `hyperref` and thereby `hyperxmp`.

```

529   \hyxmp@at@end{%
530     \hyxmp@find@metadata
531     \hyxmp@embed@packet
532   }%
533 }{%
534   \PackageWarningNoLine{hyperxmp}{%
535     \jobname.tex failed to include a\MessageBreak
536     \string\usepackage\string{hyperref}\string}

```

```

537         in the preamble.\MessageBreak
538         Consequently, all hyperxmp functionality will be\MessageBreak
539         disabled}%
540     }%
541 }

```

A number of classes either require or recommend that authors declare various class-specific metadata *after* the `\begin{document}`. So where do we invoke `\hyxmp@auto@assign@data`? On one hand, we want to invoke it before the `\begin{document}` because this may obviate hyperxmp’s “*<job name>* did not specify any metadata” warning message. On the other hand, we want to invoke it after the `\begin{document}` so it picks up metadata not specified in the preamble. Our solution is to invoke `\hyxmp@auto@assign@data` twice: both before the `\begin{document}` at at the `\end{document}`.

```

542 \hyxmp@at@end{\hyxmp@auto@assign@data}

```

### 3.3 Manipulating author-supplied data

The author provides metadata information to hyperxmp via package options to `hyperref` or via `hyperref`’s `\hypersetup` command. The functions in this section convert author-supplied lists (e.g., `pdfkeywords={foo, bar, baz}`) into L<sup>A</sup>T<sub>E</sub>X lists (e.g., `\@elt {foo} \@elt {bar} \@elt {baz}`) that can be more easily manipulated (Section 3.3.1); parse dates in both PDF and XMP formats (Section 3.3.2; trim spaces off the ends of strings (Section 3.3.3); convert text to XML (e.g., from `<scott+hyxmp@pakin.org>` to `&lt;scott+hyxmp@pakin.org&gt;`;) (Section 3.3.4); simplify the pretty-printing of a begin tag, XML text, and end tag (Section 3.3.5; and provide metadata in multiple languages (Section 3.3.6).

#### 3.3.1 List manipulation

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

```

\hyxmp@commas@to@list  Given a macro name (#1) and a comma-separated list (#2), define the macro name
                        as the elements of the list, each preceded by \@elt. (Executing the macro therefore
                        applies \@elt to each element in turn.)
543 \newcommand*{\hyxmp@commas@to@list}[2]{%
544   \gdef#1{%
545     \expandafter\hyxmp@commas@to@list@i\expandafter#1#2,,%
546 }

\hyxmp@commas@to@list@i  Recursively construct macro #1 from comma-separated list #2. Stop if #2 is empty.
\next
547 \def\hyxmp@commas@to@list@i#1#2,{%
548   \gdef\hyxmp@sublist{#2}%
549   \ifx\hyxmp@sublist@empty
550     \let\next=\relax
551   \else
552     \hyxmp@trimspaces\hyxmp@sublist

```

```

553   \@cons{#1}{\hyxmp@sublist}}}%
554   \def\next{\hyxmp@commas@to@list@i{#1}}}%
555   \fi
556   \next
557 }

```

`\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.

```
558 \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.

```

559 \bgroup
560   \catcode'\^^C=11
561   \gdef\hyxmp@comma{^^C}
562 \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.

```

563 \bgroup
564   \catcode'\^^U=11
565   \gdef\hyxmp@uscore{^^U}
566 \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.

```
567 \let\xmpquote=\relax
```

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

```

568 \bgroup
569   \catcode'\~=12%
570   \gdef\xmptilde{~}%
571 \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

`\hyxmp@temp@str`

`\hyxmp@temp@list`

`\@elt`

can then write “\XMPTruncateList{pdfauthor}” and have Adobe Acrobat display the author list correctly.

```

572 \newcommand{\XMPTruncateList}[1]{%
573   \PackageWarning{hyperxmp}{%
574     \noexpand\XMPTruncateList has been deprecated since\MessageBreak
575     hyperxmp 4.0 and may be removed in future\MessageBreak
576     versions of the package. \noexpand\XMPTruncateList\MessageBreak
577     was found}%
578   \edef\hyxmp@temp@str{\csname hyxmp@#1\endcsname}%
579   \hyxmp@commas@to@list{\hyxmp@temp@list}{\hyxmp@temp@str}%
580   \def\@elt##1{%
581     \expandafter\gdef\csname @#1\endcsname{##1}%
582     \let\@elt=\@gobble
583   }
584   \hyxmp@temp@list
585 }}

```

### 3.3.2 Date manipulation

hyperxmp needs to manipulate two types of date (really, timestamp) formats: PDF format and XMP format. PDF timestamps are of the form “D:YYYYMMDDhhmmss+TT’tt” (e.g., D:20200613135809-06’00’) [3], while XMP timestamps are of the form “YYYY-MM-DDThh:mm:ss+TT:tt” (e.g., 2020-06-13T13:58:09-06:00) [4]. The \hyxmp@as@pdf@date and \hyxmp@as@xmp@date macros defined in this section facilitate timestamp conversions to PDF and XMP formats, respectively.

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

```

\hyxmp@first@char@i 586 \def\hyxmp@first@char#1{\hyxmp@first@char@i#1\relax}
587 \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.

```

588 \def\hyxmp@as@xmp@date#1{%
589   \expandafter\ifnum\expandafter'\hyxmp@first@char@i#1\relax='D
590     \hyxmp@pdf@to@xmp@date{#1}%
591   \else
592     #1%
593   \fi
594 }

```

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

```

595 \def\hyxmp@pdf@to@xmp@date#1:#2#3#4#5#6#7#8#9{%
596   #2#3#4#5-#6#7-#8#9%
597   \hyxmp@parse@time
598 }

```

`\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`.

```

599 \def\hyxmp@parse@time#1#2#3#4#5#6{%
600   T#1#2:#3#4:#5#6%
601   \hyxmp@parse@tz@char
602 }

```

`\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.

```

603 \def\hyxmp@parse@tz@char#1{%
604   #1%
605   \ifx#1-%
606     \expandafter\hyxmp@parse@tz
607   \else
608     \ifx#1+%
609       \expandafter\hyxmp@parse@tz
610     \fi
611   \fi
612 }

```

`\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.

```

613 \def\hyxmp@parse@tz#1'#2' {%
614   #1:#2%
615 }

```

`\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.

```

616 \def\hyxmp@as@pdf@date#1{%
617   \expandafter\ifx\hyxmp@first@char@i#1\relax D%
618   #1%
619   \else
620     \hyxmp@xmp@to@pdf@date{#1}%
621   \fi
622 }

```

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

```

623 \def\hyxmp@xmp@to@pdf@date#1{%

```



```

624 D:\hyxmp@xmp@to@pdf@date@i#1\relax\relax
625 }

\hyxmp@xmp@to@pdf@date@i Parse the year for \hyxmp@xmp@to@pdf@date.
626 \def\hyxmp@xmp@to@pdf@date@i#1#2#3#4#5#6{%
627   #1#2#3#4%
628   \ifx#5-%
629     \expandafter\hyxmp@xmp@to@pdf@date@ii\expandafter#6%
630   \fi
631 }

\hyxmp@xmp@to@pdf@date@ii Parse the month for \hyxmp@xmp@to@pdf@date.
632 \def\hyxmp@xmp@to@pdf@date@ii#1#2#3#4{%
633   #1#2%
634   \ifx#3-%
635     \expandafter\hyxmp@xmp@to@pdf@date@iii\expandafter#4%
636   \fi
637 }

\hyxmp@xmp@to@pdf@date@iii Parse the day for \hyxmp@xmp@to@pdf@date.
638 \def\hyxmp@xmp@to@pdf@date@iii#1#2#3#4{%
639   #1#2%
640   \ifx#3T%
641     \expandafter\hyxmp@xmp@to@pdf@date@iv\expandafter#4%
642   \fi
643 }

\hyxmp@xmp@to@pdf@date@iv Parse the hour for \hyxmp@xmp@to@pdf@date.
644 \def\hyxmp@xmp@to@pdf@date@iv#1#2#3#4{%
645   #1#2%
646   \ifx#3:%
647     \expandafter\hyxmp@xmp@to@pdf@date@v\expandafter#4%
648   \fi
649 }

\hyxmp@xmp@to@pdf@date@v Parse the minute for \hyxmp@xmp@to@pdf@date.
650 \def\hyxmp@xmp@to@pdf@date@v#1#2#3#4{%
651   #1#2%
652   \ifx#3:%
653     \expandafter\hyxmp@xmp@to@pdf@date@vi\expandafter#4%
654   \fi
655 }

\hyxmp@gobbletwo This is exactly the same as LATEX 2ε's \@gobbletwo but needs to be a different
literal for \hyxmp@xmp@to@pdf@date@vii's pattern-matching to work.
656 \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).

```

657 \def\hyxmp@xmp@to@pdf@date@vi#1#2#3#4{%
658   #1#2%
659   \ifx#3+%
660     +\expandafter\hyxmp@xmp@to@pdf@date@vii
661   \fi
662   \ifx#3-%
663     -\expandafter\hyxmp@xmp@to@pdf@date@vii
664   \fi
665   \ifx#3Z%
666     Z%
667   \fi
668   \ifx#3\relax
669     \expandafter\hyxmp@gobbletwo
670   \fi
671   \@gobbletwo #4%
672 }

```

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

```

673 \def\hyxmp@xmp@to@pdf@date@vii#1\@gobbletwo#2#3#4#5{%
674   #2#3%
675   \ifx#4:%
676     \expandafter\hyxmp@xmp@to@pdf@date@viii\expandafter#5%
677   \fi
678 }

```

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

```

679 \def\hyxmp@xmp@to@pdf@date@viii#1#2#3#4{%
680   '#1#2'%
681 }

```

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

```

682 \def\hyxmp@today@xmp@define#1{%
  The date is a straightforward representation of  $\text{\TeX}$ 's \year, \month, and \day
  primitives, with the latter two zero-padded to two digits apiece.
683   \xdef#1{\the\year}%
684   \ifnum\month<10
685     \xdef#1{#1-0\the\month}%
686   \else
687     \xdef#1{#1-\the\month}%
688   \fi
689   \ifnum\day<10
690     \xdef#1{#1-0\the\day}%
691   \else
692     \xdef#1{#1-\the\day}%
693   \fi

```

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

```

694 \@tempcnta=\time
695 \divide\@tempcnta by 60
696 \ifnum\@tempcnta<10
697   \xdef#1{#1T0\the\@tempcnta}%
698 \else
699   \xdef#1{#1T\the\@tempcnta}%
700 \fi
701 \multiply\@tempcnta by -60
702 \advance\@tempcnta by \time
703 \ifnum\@tempcnta<10
704   \xdef#1{#1:0\the\@tempcnta}%
705 \else
706   \xdef#1{#1:\the\@tempcnta}%
707 \fi
708 \xdef#1{#1Z}%
709 }

```

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

```

710 \def\hyxmp@try@today#1#2{%
711   \@ifmtargexp{\hyxmp@today@xmp}{%
712     \@ifundefined{#1}{#2}%
713   }%
714   {}%
715 }

```

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

```

716 \def\hyxmp@today@xmp{}

```

Case 1: `\pdfcreationdate` is defined (pdfLaTeX and pre-0.85 LuaLaTeX).

```

717 \hyxmp@try@today{\pdfcreationdate}{%
718   \edef\hyxmp@today@xmp{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
719 }

```

Case 2: `\pdffeedback` is defined (LuaLaTeX 0.85+).

```

720 \hyxmp@try@today{\pdffeedback}{%
721   \edef\hyxmp@today@xmp{\expandafter\hyxmp@pdf@to@xmp@date\pdffeedback creationdate}%
722 }

```

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

```

723 \hyxmp@try@today{\filemoddate}{%
724   \edef\hyxmp@today@xmp{\filemoddate{\jobname.log}}%
725   \edef\next{%

```

```

726 \edef\noexpand\hyxmp@today@xmp{\noexpand\hyxmp@as@xmp@date{\hyxmp@today@xmp}}%
727 }%
728 \next
729 }%

```

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

```

730 \hyxmp@try@today{year}{%
731 \hyxmp@today@xmp@define\hyxmp@today@xmp
732 }

```

`\hyxmp@today@pdf` Define `\hyxmp@today@pdf` as the current date and (if available) time and timezone in PDF date format [3]. To do so we simply convert `\hyxmp@today@xmp`, defined above, from XMP to PDF using `\hyxmp@xmp@to@pdf@date`.

```

733 \expandafter\edef\expandafter\hyxmp@today@pdf\expandafter{%
734 \expandafter\hyxmp@xmp@to@pdf@date\expandafter{\hyxmp@today@xmp}%
735 }

```

### 3.3.3 Trimming leading and trailing spaces

To make it easier for XMP processors to manipulate our output we define a `\hyxmp@trimspaces` macro to strip leading and trailing spaces from various data fields.

`\hyxmp@trimspaces` Redefine a macro as its previous value but without leading or trailing spaces. This code—as well as that for its helper macros, `\hyxmp@trimb` and `\hyxmp@trimc`—was taken almost verbatim from a solution to an *Around the Bend* puzzle [6]. Inline comments are also taken from the solution text.

```

736 \catcode'\Q=3
\hyxmp@trimspaces\x redefines \x to have the same replacement text sans leading
and trailing space tokens.
737 \newcommand{\hyxmp@trimspaces}[1]{%
Use grouping to emulate a multi-token afterassignment queue.
738 \begingroup
Put “\toks 0 {” into the afterassignment queue.
739 \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.
740 \expandafter\hyxmp@trimb\expandafter\noexpand#1Q Q}%
Transfer the trimmed text back into #1.
741 \edef#1{\the\toks0}%
742 }

```

`\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.

```

743 \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.

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

### 3.3.4 Converting text to XML

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

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

```
746 \newif\ifhyxmp@unicodetex
747 \ifnum64='^^^^0040\relax
748   \hyxmp@unicodetextrue
749 \else
750   \hyxmp@unicodetexfalse
751 \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.

```
752 \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.

```
753 \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 `&lt;`; all occurrences of “>” replaced with `&gt;`; and all occurrences of “&” replaced with `&amp;`.

```
754 \newcommand*{\hyxmp@xmlify}[1]{%
755   \gdef\hyxmp@xmlified{%
      Escaped PDF string → PDFDocEncoding/Unicode
756   \EdefUnescapeString\hyxmp@text{#1}%
757   \ifhyxmp@unicodetex
```

PDFDocEncoding/Unicode → UTF-32BE

```

758 \hyxmp@is@unicode\hyxmp@text{%
759 \StringEncodingConvert
760 \hyxmp@text\hyxmp@text{utf16be}{utf32be}%
761 }{%
762 \ifXeTeX
763 \hyxmp@xetex@crap
764 \else
765 \StringEncodingConvert
766 \hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
767 \fi
768 }%

```

UTF-32BE → UTF-32BE as hex string

```

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

```

UTF-32BE → XML in ASCII

```

770 \edef\hyxmp@text{%
771 \expandafter
772 }\expandafter\hyxmp@toxml@unicodetex\hyxmp@text
773 \relax\relax\relax\relax\relax\relax\relax\relax
774 \else

```

PDFDocEncoding/Unicode → UTF-8

```

775 \hyxmp@is@unicode\hyxmp@text{%
776 \StringEncodingConvert
777 \hyxmp@text\hyxmp@text{utf16be}{utf8}%
778 }{%
779 \StringEncodingConvert
780 \hyxmp@text\hyxmp@text{pdfdoc}{utf8}%
781 }%

```

UTF-8 → UTF-8 as hex string

```

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

```

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

```

783 \edef\hyxmp@text{%
784 \expandafter\hyxmp@toxml\hyxmp@text\@empty\@empty
785 }%

```

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

```

786 \EdefUnescapeHex\hyxmp@text{\hyxmp@text}%
787 \fi
788 \global\let\hyxmp@xmlified\hyxmp@text
789 }

```

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

```

790 \begingroup
791 \lccode'\<=254 %
792 \lccode'\>=255 %
793 \catcode254=12 %

```

```

794 \catcode255=12 %
795 \lowercase{\endgroup
796 \def\hyxmp@is@unicode#1{%
797   \expandafter\hyxmp@@is@unicode#1<>\@nil
798 }%
799 \def\hyxmp@@is@unicode#1<>#2\@nil{%
800   \ifx\#1\%
801     \expandafter\@firstoftwo
802   \else
803     \expandafter\@secondoftwo
804   \fi
805 }%
806 }

```

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

```

807 \def\hyxmp@toxml#1#2{%
808   \ifx#1\@empty
809   \else
810     \ifnum"#1#2='\& %
811       26616D703B% &
812     \else\ifnum"#1#2='\< %
813       266C743B% &lt;
814     \else\ifnum"#1#2='\> %
815       2667743B% &gt;
816   \else

```

`dvips` wraps text when generating most PostScript code but preserves line breaks within strings. Unfortunately, `dvips` fails to observe the special case in the PostScript specification that “[b]alanced pairs of parentheses in the string require no special treatment” [2]. Consequently, XMP data containing parentheses (e.g., “Copyright (C) 1605 Miguel de Cervantes”) confuse `dvips` into thinking that the string has ended after the closing parenthesis and that line breaks can subsequently be injected safely into the document at arbitrary points for formatting purposes. This leads to erroneous display by PDF viewers, which honor line breaks within XMP tags. The solution is to insert a backslash before all parentheses when in `pdfmark`-generating mode to convince `dvips` that the entire XMP packet must be treated as a single, not-to-be-modified string.

```

817   \@ifundefined{pdfmark}{%
818     #1#2%
819   }{%
820     \ifnum"#1#2='\( %
821       5C28% \(
822     \else\ifnum"#1#2='\) %
823       5C29% \)
824     \else
825       #1#2%
826     \fi\fi
827   }%

```

```

828     \fi\fi\fi
829     \expandafter\hyxmp@toxml
830   \fi
831 }

```

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

```

832 \def\hyxmp@toxml@unicodetex#1#2#3#4#5#6#7#8{%
833   \ifx#1\relax
834   \else
835     \ifnum"#1#2#3#4#5#6#7#8>127 %
836       \uccode'\*="#1#2#3#4#5#6#7#8\relax
837       \uppercase{%
838         \edef\hyxmp@text{\hyxmp@text *}%
839       }%
840     \else\ifnum"#7#8='< %
841       \edef\hyxmp@text{\hyxmp@text &lt;}%
842     \else\ifnum"#7#8='& %
843       \edef\hyxmp@text{\hyxmp@text &amp}%
844     \else\ifnum"#7#8='> %
845       \edef\hyxmp@text{\hyxmp@text &gt;}%
846     \else\ifnum"#7#8='\ %
847       \edef\hyxmp@text{\hyxmp@text\space}%
848     \else
849       \uccode'\*="#7#8\relax
850       \uppercase{%
851         \edef\hyxmp@text{\hyxmp@text *}%
852       }%
853     \fi\fi\fi\fi\fi
854     \expandafter\hyxmp@toxml@unicodetex
855   \fi
856 }

```

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

```

857 \def\hyxmp@skipzeros#1{%
858   \ifx#10%
859     \expandafter\hyxmp@skipzeros
860   \fi
861 }

```

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

```

\hyxmp@xetex@crap \hyxmp@try 862 \begingroup
\hyxmp@crap@result 863 \def\x#1{\endgroup
\hyxmp@text 864 \def\hyxmp@xetex@crap{%
865   \edef\hyxmp@try{%
866     \expandafter\hyxmp@SpaceOther\hyxmp@text#1\@nil
867   }%
868   \let\hyxmp@crap@result=N%

```



```

869 \expandafter\hyxmp@crap@test\hyxmp@try\relax
870 \ifx\hyxmp@crap@result Y%
871 \let\hyxmp@text\@empty
872 \expandafter\hyxmp@crap@convert\hyxmp@try\relax
873 \else
874 \StringEncodingConvert\hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
875 \fi
876 }%
877 }
878 \x{ }

```

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

```

879 \begingroup
880 \catcode'\~=12 %
881 \lccode'\~=\' %
882 \lowercase{\endgroup
883 \def\hyxmp@SpaceOther#1 #2\@nil{%
884 #1%
885 \ifx\relax#2\relax
886 \expandafter\@gobble
887 \else
888 ~%
889 \expandafter\@firstofone
890 \fi
891 {\hyxmp@SpaceOther#2\@nil}%
892 }%
893 }

```

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

```

894 \def\hyxmp@crap@test#1{%
895 \ifx#1\relax
896 \else
897 \ifnum'#1>127 %
898 \let\hyxmp@crap@result=Y%
899 \expandafter\expandafter\expandafter\hyxmp@skiptorelax
900 \else
901 \expandafter\expandafter\expandafter\hyxmp@crap@test
902 \fi
903 \fi
904 }

```

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

```

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

```

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

```

\hyxmp@num 906 \def\hyxmp@crap@convert#1{%
\hyxmp@text 907 \ifx#1\relax
908 \else
909 \edef\hyxmp@num{\number'#1}%

```

```

910 \ifnum\hyxmp@num>"FFFFFF %
911 \lccode'\!=\intcalcDiv{\hyxmp@num}{\number"1000000}\relax
912 \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
913 \edef\hyxmp@num{\intcalcMod{\hyxmp@num}{\number"1000000}}%
914 \else
915 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
916 \fi
917 \ifnum\hyxmp@num>"FFFF %
918 \lccode'\!=\intcalcDiv{\hyxmp@num}{\number"10000}\relax
919 \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
920 \edef\hyxmp@num{\intcalcMod{\hyxmp@num}{\number"10000}}%
921 \else
922 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
923 \fi
924 \ifnum\hyxmp@num>"FF %
925 \lccode'\!=\intcalcDiv{\hyxmp@num}{\number"100}\relax
926 \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
927 \edef\hyxmp@num{\intcalcMod{\hyxmp@num}{\number"100}}%
928 \else
929 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
930 \fi
931 \ifnum\hyxmp@num>0 %
932 \lccode'\!=\hyxmp@num\relax
933 \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
934 \else
935 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
936 \fi
937 \expandafter\hyxmp@crap@convert
938 \fi
939 }

```

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

```

940 \begingroup
941 \catcode0=12 %
942 \gdef\hyxmp@zero{~00}%
943 \endgroup

```

### 3.3.5 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.

```

944 \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.

```

945 \newcommand*{\hyxmp@add@simple}[2]{%
946   \@ifnotmtargexp{#2}{%
947     \hyxmp@xmllify{#2}%
948     \hyxmp@add@to@xml{\hyxmp@extra@indent_<}}%
949   \xdef\hyxmp@xml{\hyxmp@xml#1}%
950   \hyxmp@add@to@xml{>\hyxmp@xmllified</}%
951   \xdef\hyxmp@xml{\hyxmp@xml#1>^^J}%
952 }%
953 }

```

`\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.

```

954 \newcommand*{\hyxmp@add@simple@var}[2]{%
955   \expandafter\ifx\csname#2\endcsname\relax
956   \else
957     \hyxmp@xmllify{\csname#2\endcsname}%
958     \hyxmp@add@to@xml{%
959       \hyxmp@extra@indent_<#1>\hyxmp@xmllified</#1>^^J%
960     }%
961   \fi
962 }

```

`\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.

```

963 \newcommand*{\hyxmp@add@simple@lang}[2]{%
964   \@ifnotmtarg{#2}{%
965     \hyxmp@xmllify{#2}%
966     \expandafter\hyxmp@add@simple@lang@i\hyxmp@xmllified\relax{#1}%
967   }%
968 }

```

`\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.

```

969 \newcommand*{\hyxmp@add@simple@lang@i}[1]{%
970   \@ifnextchar[\hyxmp@add@simple@lang@ii{\hyxmp@add@simple@lang@ii[]}%
971 }

```

`\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.

```

972 \def\hyxmp@add@simple@lang@ii[#1]#2\relax#3{%
973   \ifnotmtarg{#2}{%
974     \hyxmp+xmlify{#2}%
975     \@ifmtarg{#1}{%
976       \hyxmp@add@to+xml{%
977         <#3>\hyxmp+xmlified</#3>^^J%
978       }%
979     }%
980     \hyxmp@add@to+xml{%
981       <#3 xml:lang="#1">\hyxmp+xmlified</#3>^^J%
982     }%
983   }%
984 }%
985 }

```

`\hyxmp@add@simple@pfx` Given an XMP tag (#1), a—typically hard-wired—prefix string (#2), and a main string (#2), 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.

```

986 \newcommand*{\hyxmp@add@simple@pfx}[3]{%
987   \ifnotmtargexp{#3}{%
988     \hyxmp@add@to+xml{\hyxmp@extra@indent_____<}%
989     \xdef\hyxmp+xml{\hyxmp+xml#1}%
990     \hyxmp@pdfstringdef\hyxmp@iprefix{#2}%
991     \hyxmp+xmlify{\hyxmp@iprefix}%
992     \hyxmp@add@to+xml{>\hyxmp+xmlified}%
993     \hyxmp+xmlify{#3}%
994     \hyxmp@add@to+xml{\hyxmp+xmlified</}%
995     \xdef\hyxmp+xml{\hyxmp+xml#1>^^J}%
996   }%
997 }

```

### 3.3.6 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}`”.

```

998 \def\hyxmp@alt@title{}

```

```

999 \def\hyxmp@alt@description{}
1000 \def\hyxmp@alt@rights{}

\hyxmp@LA@accept This macro wraps \define@key to make the option “#1=⟨value⟩” append ⟨value⟩
to list #2.
1001 \newcommand{\hyxmp@LA@accept}[2]{%
1002   \define@key{hyxmp@LA}{#1}{%

\hyxmp@value As Niklas Beisert observed, if the option passed to the current key contains LATEX
code, this code will be included in the XMP packet, which is undesirable. Hence,
we first clean up the string using \hyxmp@pdfstringdef.
1003   \hyxmp@pdfstringdef\hyxmp@value{##1}%
1004   \xdef#2{#2\noexpand\do {\hyxmp@cur@lang} {\hyxmp@value}}%
1005 }
1006 }

Define ⟨key⟩=⟨value⟩ options for appending to each of the \hyxmp@alt⟨tag⟩
lists.
1007 \hyxmp@LA@accept{pdftitle}{\hyxmp@alt@title}
1008 \hyxmp@LA@accept{pdfsubject}{\hyxmp@alt@description}
1009 \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.
1010 \newcommand{\XMPLangAlt}[2]{%
1011   \let\do=\relax

\hyxmp@cur@lang Store the provided language, which will be used during option processing.
1012   \edef\hyxmp@cur@lang{#1}%
1013   \setkeys{hyxmp@LA}{#2}%
1014 }

```

### 3.4 UUID generation

We use a linear congruential generator to produce pseudorandom version 4 UUIDs [11]. True, this method has its flaws but it’s simple to implement in T<sub>E</sub>X and is good enough for producing the XMP xmpMM:DocumentID and xmpMM:InstanceID fields.

```

\hyxmp@modulo@a Replace the contents of \@tempcnta with the contents modulo #1. Note that
\@tempcntb is overwritten in the process.
1015 \def\hyxmp@modulo@a#1{%
1016   \@tempcntb=\@tempcnta
1017   \divide\@tempcntb by #1
1018   \multiply\@tempcntb by #1

```

```

1019 \advance\@tempcnta by -\@tempcntb
1020 }

\hyxmp@big@prime Define a couple of large prime numbers that can still be stored in a TEX counter.
\hyxmp@big@prime@ii 1021 \def\hyxmp@big@prime{536870923}
1022 \def\hyxmp@big@prime@ii{536870027}

\hyxmp@seed@rng Seed hyperxmp's random-number generator from a given piece of text.
\hyxmp@one@token 1023 \def\hyxmp@seed@rng#1{%
1024 \@tempcnta=\hyxmp@big@prime
1025 \futurelet\hyxmp@one@token\hyxmp@seed@rng@i#1\@empty
1026 }

\hyxmp@seed@rng@i Do all of the work for \hyxmp@seed@rng. For each character code  $c$  of the input
\hyxmp@one@token text, assign  $\@tempcnta \leftarrow 3 \cdot \@tempcnta + c \pmod{\hyxmp@big@prime}$ .
\next 1027 \def\hyxmp@seed@rng@i{%
1028 \ifx\hyxmp@one@token\@empty
1029 \let\next=\relax
1030 \else
1031 \def\next##1{%
1032 \multiply\@tempcnta by 3
1033 \advance\@tempcnta by '##1
1034 \hyxmp@modulo@a{\hyxmp@big@prime}%
1035 \futurelet\hyxmp@one@token\hyxmp@seed@rng@i
1036 }%
1037 \fi
1038 \next
1039 }

\hyxmp@set@rand@num Advance \hyxmp@rand@num to the next pseudorandom number in the se-
\hyxmp@rand@num quence. Specifically, we assign  $\hyxmp@rand@num \leftarrow 3 \cdot \hyxmp@rand@num + \hyxmp@big@prime@ii \pmod{\hyxmp@big@prime}$ . Note that both \@tempcnta
and \@tempcntb are overwritten in the process.
1040 \def\hyxmp@set@rand@num{%
1041 \@tempcnta=\hyxmp@rand@num
1042 \multiply\@tempcnta by 3
1043 \advance\@tempcnta by \hyxmp@big@prime@ii
1044 \hyxmp@modulo@a{\hyxmp@big@prime}%
1045 \xdef\hyxmp@rand@num{\the\@tempcnta}%
1046 }

\hyxmp@append@hex Append a randomly selected hexadecimal digit to macro #1. Note that both
\@tempcnta and \@tempcntb are overwritten in the process.
1047 \def\hyxmp@append@hex#1{%
1048 \hyxmp@set@rand@num
1049 \@tempcnta=\hyxmp@rand@num
1050 \hyxmp@modulo@a{16}%
1051 \ifnum\@tempcnta<10
1052 \xdef#1{#1\the\@tempcnta}%
1053 \else

```

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

```

1054     \advance\@tempcnta by -10
1055     \ifcase\@tempcnta
1056         \xdef#1{#1a}%
1057         \or\xdef#1{#1b}%
1058         \or\xdef#1{#1c}%
1059         \or\xdef#1{#1d}%
1060         \or\xdef#1{#1e}%
1061         \or\xdef#1{#1f}%
1062     \fi
1063 \fi
1064 }

```

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

```

1065 \def\hyxmp@append@hex@iii#1{%
1066     \hyxmp@append@hex#1%
1067     \hyxmp@append@hex#1%
1068     \hyxmp@append@hex#1%
1069 }

```

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

```

1070 \def\hyxmp@append@hex@iv#1{%
1071     \hyxmp@append@hex@iii#1%
1072     \hyxmp@append@hex#1%
1073 }

```

`\hyxmp@create@uuid` As per the definition of a version 4 UUID [11], define macro `#1` as a UUID of the form “`uuid:xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx`” in which each “`x`” is a lowercase hexadecimal digit and “`y`” is one of “8”, “9”, “a”, or “b”. We assume that the random-number generator is already seeded. Note that `\hyxmp@create@uuid` overwrites both `\@tempcnta` and `\@tempcntb`.

```

1074 \def\hyxmp@create@uuid#1{%
1075     \def#1{uuid:}%
1076     \hyxmp@append@hex@iv#1%
1077     \hyxmp@append@hex@iv#1%
1078     \g@addto@macro#1{-}%
1079     \hyxmp@append@hex@iv#1%
1080     \g@addto@macro#1{-4}%
1081     \hyxmp@append@hex@iii#1%
1082     \g@addto@macro#1{-}%

```

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

```

1083     \hyxmp@set@rand@num
1084     \@tempcnta=\hyxmp@rand@num
1085     \hyxmp@modulo@a{4}%
1086     \ifcase\@tempcnta
1087         \g@addto@macro#1{8}%
1088         \or\g@addto@macro#1{9}%
1089         \or\g@addto@macro#1{a}%

```

```

1090     \or\g@addto@macro#1{b}%
1091     \fi
1092     \hyxmp@append@hex@iii#1%
1093     \g@addto@macro#1{-}%
1094     \hyxmp@append@hex@iv#1%
1095     \hyxmp@append@hex@iv#1%
1096     \hyxmp@append@hex@iv#1%
1097 }

\hyxmp@def@DocumentID Seed the random-number generator with a function of the current filename, PDF
    \hyxmp@DocumentID document title, and PDF author, then invoke \hyxmp@create@uuid to define
    \hyxmp@seed@string \hyxmp@DocumentID as a random UUID.
1098 \newcommand*{\hyxmp@def@DocumentID}{%
1099     \edef\hyxmp@seed@string{\jobname:\@pdftitle:\@pdfauthor}%
1100     \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
1101     \edef\hyxmp@rand@num{\the\@tempcnta}%
1102     \hyxmp@create@uuid\hyxmp@DocumentID
1103 }

\hyxmp@def@InstanceID Seed the random-number generator with a function of the current filename,
    \hyxmp@InstanceID PDF document title, PDF author, and the current timestamp, then invoke
    \hyxmp@seed@string \hyxmp@create@uuid to define \hyxmp@InstanceID as a random UUID. For the
    current timestamp, we use both the document-specified timestamp from pdfdate
    and the TEX time. The former can be more precise (to sub-seconds) but may be
    less random (as it depends on manual document modifications) while the latter
    is typically less precise (to minutes) but may be more random (as it is updated
    automatically).
1104 \newcommand*{\hyxmp@def@InstanceID}{%
1105     \hyxmp@today@xmp@define{\hyxmp@seed@string}%
1106     \edef\hyxmp@seed@string{%
1107         \jobname:\@pdftitle:\@pdfauthor:\hyxmp@today@xmp:\hyxmp@seed@string
1108     }%
1109     \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
1110     \edef\hyxmp@rand@num{\the\@tempcnta}%
1111     \hyxmp@create@uuid\hyxmp@InstanceID
1112 }

```

### 3.5 Constructing the XMP packet

An XMP packet “shall consist of the following, in order: a header PI, the serialized XMP data model (the XMP packet) with optional white-space padding, and a trailer PI” [4]. (“PI” is an abbreviation for “processing instructions”). The serialized XMP includes blocks of XML for various XMP schemata: Adobe PDF (Section 3.5.2), Dublin Core (Section 3.5.3), XMP Rights Management (Section 3.5.4), XMP Media Management (Section 3.5.5), XMP Basic (Section 3.5.6), Photoshop (Section 3.5.7), IPTC Photo Metadata (Section 3.5.9), and PDF/\* Identification (Section 3.5.8). The `\hyxmp@construct@packet` macro (Section 3.5.12) constructs the XMP packet into `\hyxmp+xml`. It first writes the appropriate XML header, then calls the various



schema-writing macros, then injects `\hyxmp@padding` as padding, and finally writes the appropriate XML trailer.

### 3.5.1 XMP utility functions

`\hyxmp@add@to+xml` Given a piece of text, replace all underscores with category-code 11 (“other”) spaces and all `^C` characters with commas, then append the result to the `\hyxmp+xml` macro.

```

1113 \newcommand*{\hyxmp@add@to+xml}[1]{%
1114   \bgroup
1115     \@tempcnta=0
1116     \ifhyxmp@unicodetex
1117       \@tempcntb=65536%
1118     \else
1119       \@tempcntb=256%
1120     \fi
1121     \loop
1122       \lccode\@tempcnta=\@tempcnta
1123       \advance\@tempcnta by 1
1124       \ifnum\@tempcnta<\@tempcntb
1125         \repeat
1126         \lccode'\_='\ \relax
1127         \lccode'\^C='\,\relax
1128         \lccode'\^U='\_\relax
1129         \lowercase{\xdef\hyxmp@new+xml{#1}}%
1130         \xdef\hyxmp+xml{\hyxmp+xml\hyxmp@new+xml}%
1131     \egroup
1132 }
```

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

```

1133 \bgroup
1134 \catcode'\#=11
1135 \gdef\hyxmp@hash{#}
1136 \egroup
```

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

```

1137 \bgroup
1138 \xdef\hyxmp+xml{}%
1139 \hyxmp@add@to+xml{%
1140   -----^^J%
1141 }
1142 \xdef\hyxmp@padding{\hyxmp+xml}%
1143 \egroup
1144 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
1145 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
1146 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
```

```

1147 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
1148 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}

\hyxmp@x@default Define an x-default string that we can use in comparisons with \@pdfmetalang.
1149 \newcommand*{\hyxmp@x@default}{x-default}

```

### 3.5.2 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.

1150 \newcommand*{\hyxmp@define@pdfproducer}{%
1151   \gdef\@pdfproducer{TeX}
1152   \ifLuaTeX
1153     \expandafter\hyxmp@banner@to@producer\expandafter{\luatexbanner}%
1154   \else
1155     \ifPDFTeX
1156       \expandafter\hyxmp@banner@to@producer\expandafter{\pdftexbanner}%
1157     \else
1158       \ifXeTeX
1159         \edef\@pdfproducer{XeTeX version \the\XeTeXversion\XeTeXrevision}%
1160       \fi
1161     \fi
1162   \fi
1163 }

\@pdfproducer Define \@pdfproducer as the TeX engine's banner string (e.g., "This is pdfTeX,
\hyxmp@banner@to@producer Version 3.14159265-2.6-1.40.21 (TeX Live 2020) kpathsea version
6.3.2"), removing the initial "This is" if possible (specifically, when  $\varepsilon$ -TeX's
\scantokens primitive is available).

1164 \def\hyxmp@banner@to@producer#1{%
1165   \ifx\scantokens\relax
1166     \gdef\@pdfproducer{#1}%
1167   \else
1168     {\scantokens{\makeatletter\hyxmp@remove@this#1\relax}}%
1169   \fi
1170 }

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

1171 \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.

```
1172 \AtBeginDocument{%
1173   \ifx\pdfproducer\relax
1174     \hyxmp@define@pdfproducer
1175   \fi
1176 }
```

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

`\hyxmp@major@minor`

```
1177 \newcommand*{\hyxmp@assign@major@minor}{%
1178   \@ifundefined{pdfvariable}{%
1179     \@ifundefined{pdfminorversion}{%
```

Case 1: Neither `\pdfvariable` nor `\pdfminorversion` is defined (X<sub>Y</sub>L<sup>A</sup>T<sub>E</sub>X and regular L<sup>A</sup>T<sub>E</sub>X).

```
1180   }{%
```

Case 2: `\pdfminorversion` is defined (pdfL<sup>A</sup>T<sub>E</sub>X and pre-0.85 LuaL<sup>A</sup>T<sub>E</sub>X).

```
1181     \xdef\hyxmp@major@minor{\the\pdfminorversion}%
1182     \@ifundefined{pdfmajorversion}{%
```

Case 2(a): `\pdfmajorversion` is not defined (older versions of pdfL<sup>A</sup>T<sub>E</sub>X and LuaL<sup>A</sup>T<sub>E</sub>X).

```
1183       \xdef\hyxmp@major@minor{1.\hyxmp@major@minor}%
1184     }{%
```

Case 2(b): `\pdfmajorversion` is defined (pdfL<sup>A</sup>T<sub>E</sub>X 1.40.21+).

```
1185       \xdef\hyxmp@major@minor{\the\pdfmajorversion.\hyxmp@major@minor}%
1186     }%
1187   }%
1188 }
```

Case 3: `\pdfvariable` is defined (LuaL<sup>A</sup>T<sub>E</sub>X 0.85+).

```
1189   \xdef\hyxmp@major@minor{\the\pdfvariable majorversion.\the\pdfvariable minorversion}%
1190   }%
1191 }
```

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

```
1192 \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.

```

1193 \hyxmp@add@simple@var{pdf:Producer}{@pdfproducer}%
1194 \hyxmp@add@simple@var{pdf:Keywords}{@pdfkeywords}%
1195 \hyxmp@add@simple{pdf:Trapped}{@pdftrapped}%
1196 \hyxmp@assign@major@minor
1197 \hyxmp@add@simple@var{pdf:PDFVersion}{hyxmp@major@minor}%
1198 }

```

### 3.5.3 The Dublin Core schema

`\hyxmp@rdf@dc` Given an optional `\if<something>` statement (#1), a Dublin Core property (#2) and a macro containing some `\pdfstringdef`-defined text (#3), append the appropriate block of XML to the `\hyxmp@xml` macro.

```

1199 \newcommand*{\hyxmp@rdf@dc}[3][\iffalse]{%
    Set \@tempswatrue only if the given text is nonempty or the provided conditional
    evaluates to TRUE.

```

```

1200 \ifmtargexp{#3}{\@tempswafalse}{\@tempswatrue}%
1201 #1
1202 \@tempswatrue
1203 \fi

```

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

```

1204 \if@tempswa
1205 \hyxmp@xmlify{#3}%

```

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

```

1206 \let\hyxmp@value=\hyxmp@xmlified
1207 \hyxmp@add@to@xml{%
1208 -----<dc:#2>^^J%
1209 -----<rdf:Alt>^^J%
1210 }%
1211 \ifx\@pdfmetalang\hyxmp@x@default
1212 \else
1213 \hyxmp@xmlify{\@pdfmetalang}%
1214 \hyxmp@add@to@xml{%
1215 -----<rdf:li xml:lang="\hyxmp@xmlified">\hyxmp@value</rdf:li>^^J%
1216 }%
1217 \fi
1218 \hyxmp@add@to@xml{%
1219 -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@value</rdf:li>^^J%
1220 }%

```

Include variants of the text expressed in other languages, as specified by the author using `\XMPLangAlt` (Section 3.3.6).

```

1221 \def\do##1##2{
1222 \hyxmp@xmlify{##2}%
1223 \hyxmp@add@to@xml{%

```

```

1224 -----<rdf:li xml:lang="##1">\hyxmp@xmlified</rdf:li>^^J%
1225     }%
1226 }%
1227 \csname hyxmp@alt@#2\endcsname
    Complete this XMP element.
1228 \hyxmp@add@to@xml{%
1229 -----</rdf:Alt>^^J%
1230 -----</dc:#2>^^J%
1231     }%
1232 \fi
1233 }%

```

`\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.

```

1234 \newcommand*{\hyxmp@list@to@xml}[4][\iffalse]{%
    Set \@tempswattrue only if the given list is nonempty or the provided conditional
    evaluates to TRUE.
1235 \ifmtargexp{#4}{\@tempswafalse}{\@tempswattrue}%
1236 #1
1237 \@tempswattrue
1238 \fi
    Append the corresponding XML only if \@tempswattrue.
1239 \if@tempswa
1240 \hyxmp@add@to@xml{%
1241 -----<dc:#2>^^J%
1242 -----<rdf:#3>^^J%
1243     }%
1244 \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`.

```

1245 \hyxmp@xmlify{#4}%
1246 \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
1247 \def\@elt#1{%
1248 \hyxmp@add@to@xml{%
1249 -----<rdf:li>##1</rdf:li>^^J%
1250     }%
1251 }%
1252 \hyxmp@list
1253 \egroup
1254 \hyxmp@add@to@xml{%
1255 -----</rdf:#3>^^J%
1256 -----</dc:#2>^^J%
1257     }%
1258 \fi
1259 }

```

`\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.

```

1260 \newcommand{\hyxmp@singleton@dc}[3][Bag]{%
1261   \ifnotmtargexp{#3}{%
1262     \hyxmp@xmlify{#3}%
1263     \hyxmp@add@to@xml{%
1264       <dc:#2>^^J%
1265       <rdf:#1>^^J%
1266       <rdf:li>\hyxmp@xmlified</rdf:li>^^J%
1267       </rdf:#1>^^J%
1268     </dc:#2>^^J%
1269   }%
1270 }
1271 }
```

`\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).

```

1272 \newcommand*{\hyxmp@cond@dc@identifier}[2]{%
1273   \ifx\hyxmp@xmlified@empty
1274     \ifnotmtargexp{#2}{%
1275       \hyxmp@add@simple@pfx{dc:identifier}{#1}{#2}%
1276     }%
1277   \fi
1278 }
```

`\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:date` property using the date the document was run through L<sup>A</sup>T<sub>E</sub>X and the `dc:source` property using the base name of the source file with `.tex` appended.

```

1279 \newcommand*{\hyxmp@dc@schema}{%
1280   \hyxmp@add@simple{dc:format}{application/pdf}%
1281   \hyxmp@rdf@dc[\ifHy@pdfa]{title}{\@pdftitle}%
1282   \hyxmp@rdf@dc[\ifHy@pdfa]{description}{\@pdfsubject}%
1283   \hyxmp@rdf@dc{rights}{\@pdfcopyright}%
1284   \hyxmp@singleton@dc{publisher}{\@pdfpublisher}%
1285   \hyxmp@singleton@dc[Seq]{date}{\hyxmp@today@xmp}%
1286   \hyxmp@singleton@dc{type}{\@pdftype}%
1287   \hyxmp@list@to@xml[\ifHy@pdfa]{creator}{Seq}{\hyxmp@pdfauthor}%
1288   \hyxmp@list@to@xml{subject}{Bag}{\hyxmp@pdfkeywords}%
1289   \ifx\@pdfsource@empty
```

```

1290 \else
1291   \hyxmp@add@simple{dc:source}{\@pdfsource}%
1292 \fi

Set \hyxmp@dc@lang to either the single language \@pdflang (which may be
unspecified) or to a list of languages used in the document via polyglossia. See
Section 3.2 for the definition of \hyxmp@set@dc@lang.

1293 \hyxmp@set@dc@lang
1294 \hyxmp@list@to@xml{language}{Bag}{\hyxmp@dc@lang}%

If \@pdfidentifier is empty, try setting it to each of \@pdfdoi, \@pdfeissn,
\@pdfissn, and \@pdfisbn, in turn, with proper syntactic adjustments.

1295 \@ifmtargexp{\@pdfidentifier}{%
1296   \let\hyxmp@xmlified=\@empty
1297   \hyxmp@cond@dc@identifier{info:doi/}{\@pdfdoi}%
1298   \hyxmp@cond@dc@identifier{urn:ISSN:}{\@pdfeissn}%
1299   \hyxmp@cond@dc@identifier{urn:ISSN:}{\@pdfissn}%
1300   \hyxmp@cond@dc@identifier{urn:ISBN:}{\@pdfisbn}%
1301 }{%
1302   \hyxmp@add@simple{dc:identifier}{\@pdfidentifier}%
1303 }%
1304 }

```

### 3.5.4 The XMP Rights Management schema

`\hyxmp@xmpRights@schema` Add properties defined by the XMP Rights Management schema to the `\hyxmp@xml` macro. Currently, these are only the `xmpRights:Marked` property and the `xmpRights:WebStatement` property. If the author specified a copyright statement we mark the document as copyrighted. If the author specified a license statement we include the URL in the metadata.

```

1305 \newcommand*{\hyxmp@xmpRights@schema}{%

```

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

```

1306 \let\hyxmp@rights=\@empty
1307 \ifx\@pdflicenseurl\@empty
1308 \else
1309   \def\hyxmp@rights{YES}%
1310 \fi
1311 \ifx\@pdfcopyright\@empty
1312 \else
1313   \def\hyxmp@rights{YES}%
1314 \fi

```

Include the license-statement URL and/or the copyright indication. The copyright statement itself is included by `\hyxmp@dc@schema` in Section 3.5.3.

```

1315 \ifx\hyxmp@rights\@empty
1316 \else
1317   \ifx\@pdfcopyright\@empty
1318   \else
1319     \hyxmp@add@simple{xmpRights:Marked}{True}%

```

```

1320     \fi
1321     \hyxmp@add@simple{xmpRights:WebStatement}{\@pdflicenseurl}%
1322 \fi
1323 }

```

### 3.5.5 The XMP Media Management schema

`\hyxmp@mm@schema` Add properties defined by the XMP Media Management schema to the `\hyxmp@xml` macro. According to the XMP specification, the `xmpMM:DocumentID` property is supposed to uniquely identify a document, and the `xmpMM:InstanceID` property is supposed to change with each save operation [4]. As seen in Section 3.4, we do what we can to honor this intention from within a T<sub>E</sub>X-based workflow. We additionally support the `xmpMM:VersionID` property, whose value is supplied by the author using `pdfversionid`.

```

1324 \gdef\hyxmp@mm@schema{%
1325   \ifmtargexp{\hyxmp@DocumentID}{\hyxmp@def@DocumentID}{}%
1326   \ifmtargexp{\hyxmp@InstanceID}{\hyxmp@def@InstanceID}{}%
1327   \hyxmp@add@simple{xmpMM:DocumentID}{\hyxmp@DocumentID}%
1328   \hyxmp@add@simple{xmpMM:InstanceID}{\hyxmp@InstanceID}%
1329   \hyxmp@add@simple{xmpMM:VersionID}{\@pdfversionid}%
1330   \hyxmp@add@simple{xmpMM:RenditionClass}{\@pdfrendition}%
1331 }

```

### 3.5.6 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`.

```

1332 \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.
1333   \ifmtargexp{\@pdfcreationdate}{%
1334     \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@today@xmp}%
1335   }{%
1336     \hyxmp@add@simple{xmp:CreateDate}{%
1337       \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfcreationdate}}%
1338   }%
    For the document's modification date, use the user-specified \@pdfmoddate if
    defined and non-empty. Otherwise use our fabricated \hyxmp@today@xmp.
1339   \ifmtargexp{\@pdfmoddate}{%
1340     \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@today@xmp}%
1341   }{%
1342     \hyxmp@add@simple{xmp:ModifyDate}{%
1343       \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfmoddate}}%
1344   }%

```



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

```
1345 \ifmtargexp{\@pdfmetadatettime}{%
1346   \hyxmp@add@simple{xmp:MetadataDate}{\hyxmp@today@xmp}%
1347 }{%
1348   \hyxmp@add@simple{xmp:MetadataDate}{\@pdfmetadatettime}%
1349 }%
```

Define the creation tool and the base URL.

```
1350 \hyxmp@add@simple{xmp:CreatorTool}{\@pdfcreator}%
1351 \hyxmp@add@simple{xmp:BaseURL}{\@baseurl}%
1352 }
```

### 3.5.7 The Photoshop schema

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

```
1353 \gdef\hyxmp@photoshop@schema{%
1354   \edef\hyxmp@photoshop@data{\@pdfauthortitle\@pdfcaptionwriter}%
1355   \hyxmp@add@simple{photoshop:AuthorsPosition}{\@pdfauthortitle}%
1356   \hyxmp@add@simple{photoshop:CaptionWriter}{\@pdfcaptionwriter}%
1357 }
```

### 3.5.8 PDF/\* Identification schemata

`\hyxmp@pdfa@id@schema` Add properties defined by the PDF/A Identification schema [12] to the `\hyxmp+xml` macro. These properties identify a document as conforming to a particular PDF/A standard. We default to PDF/A-1b if any PDF/A compliance is detected but let the author override the “1” with `pdfapart` and the “b” with `pdfaconformance`.

```
1358 \newcommand*{\hyxmp@pdfa@id@schema}{%
1359   \ifHy@pdfa
1360     \hyxmp@add@simple{pdfaid:part}{\@pdfapart}%
1361     \hyxmp@add@simple{pdfaid:conformance}{\@pdfaconformance}%
1362   \fi
1363 }
```

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

```
1364 \newcommand*{\hyxmp@pdfua@id@schema}{%
1365   \hyxmp@add@simple{pdfuaid:part}{\@pdfuapart}%
1366 }
```

`\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.

```
1367 \newcommand*{\hyxmp@pdfx@id@schema}{%
1368   \@tempcnta=0\hyxmp@pdfx@major\relax
```

```

1369 \ifnum\@tempcnta=0
1370 \else
1371 \ifnum\@tempcnta=1
1372 \hyxmp@add@simple{pdfx:GTS_PDFXVersion}{PDF/X-1:2001}%
1373 \hyxmp@add@simple{pdfx:GTS_PDFXConformance}{\@pdfxstandard}%
1374 \else
1375 \ifnum\@tempcnta<4
1376 \hyxmp@add@simple{pdfx:GTS_PDFXVersion}{\@pdfxstandard}%
1377 \else
1378 \hyxmp@add@simple{pdfxid:GTS_PDFXVersion}{\@pdfxstandard}%
1379 \fi
1380 \fi
1381 \fi
1382 }

```

### 3.5.9 The IPTC Photo Metadata schema

`\xmplinesep` Lines in multiline fields are separated by `\xmplinesep` in the generated XML. This defaults to an LF (`^^J`) character but written as an XML character entity for consistency across operating systems.

```

1383 \begingroup
1384 \catcode'\&=12
1385 \catcode'\#=12
1386 \gdef\xmplinesep{&#xA;}
1387 \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.

```

1388 \newcommand*{\hyxmp@list@to@lines}[2]{%
1389 \ifnotmtargexp{#2}{%
1390 \bgroup
1391 \hyxmp@add@to@xml{%
1392 \hyxmp@extra@indent_____<#1>%
1393 }%

```

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

```

1394 \def\@elt@first##1{%
1395 \hyxmp@add@to@xml{##1}%
1396 \let\@elt=\@elt@rest
1397 }%

```

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

```

1398 \def\@elt@rest##1{%
1399 \hyxmp@add@to@xml{\xmplinesep##1}%
1400 }%

```

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

```

1401     \let\@elt=\@elt@first
1402     \hyxmp@xmllify{#2}%
1403     \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmllified}%
1404     \hyxmp@list
1405     \hyxmp@add@to@xml{</#1>^^J}%
1406   \egroup
1407 }%
1408 }

```

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

```

1409 \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.

```

1410   \ifx\hyxmp@iptc@data\@empty
1411   \else
1412     \hyxmp@add@to@xml{%
1413     -----<Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">^^J%
1414     }%

```

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

```

1415   \bgroup
1416   \edef\hyxmp@extra@indent{\hyxmp@extra@indent\space\space}%
1417   \hyxmp@list@to@lines{Iptc4xmpCore:CiAdrExtadr}{\@pdfcontactaddress}%
1418   \hyxmp@add@simple{Iptc4xmpCore:CiAdrCity}{\@pdfcontactcity}%
1419   \hyxmp@add@simple{Iptc4xmpCore:CiAdrRegion}{\@pdfcontactregion}%
1420   \hyxmp@add@simple{Iptc4xmpCore:CiAdrPcode}{\@pdfcontactpostcode}%
1421   \hyxmp@add@simple{Iptc4xmpCore:CiAdrCtry}{\@pdfcontactcountry}%

```

`\xmplinesep` The IPTC standard states that sets of telephone numbers, email addresses, and URLs for the contact person or institution, “[m]ay have to be separated by a comma in the user interface” [9]. This is rather ambiguous: Does the comma appear *only* in the user interface or also in the generated XML? Here we assume the latter interpretation and temporarily redefine `\xmplinesep` as a comma and use `\hyxmp@list@to@lines` to insert the data. Unlike `\hyxmp@add@simple`, this approach trims all spaces surrounding commas.

```

1422   \def\xmplinesep{,%
1423   \hyxmp@list@to@lines{Iptc4xmpCore:CiTelWork}{\@pdfcontactphone}%
1424   \hyxmp@list@to@lines{Iptc4xmpCore:CiEmailWork}{\@pdfcontactemail}%
1425   \hyxmp@list@to@lines{Iptc4xmpCore:CiUrlWork}{\@pdfcontacturl}%
1426   \egroup
1427   \hyxmp@add@to@xml{%
1428   -----</Iptc4xmpCore:CreatorContactInfo>^^J%
1429   }%
1430   \fi
1431 }

```

### 3.5.10 The PRISM Basic Metadata schema

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

```

1432 \newcommand*{\hyxmp@prism@schema}{%
1433   \ifx\hyxmp@prism@data\@empty
1434   \else
1435     \hyxmp@add@simple{prism:complianceProfile}{three}%
1436   \fi
1437   \hyxmp@add@simple@lang{prism:subtitle}{\@pdfsubtitle}%
1438   \hyxmp@add@simple@lang{prism:publicationName}{\@pdfpublication}%
1439   \hyxmp@add@simple{prism:aggregationType}{\@pdfpubtype}%
1440   \hyxmp@add@simple@lang{prism:bookEdition}{\@pdfbookedition}%
1441   \hyxmp@add@simple{prism:volume}{\@pdfvolumenum}%
1442   \hyxmp@add@simple{prism:number}{\@pdfissuenum}%
1443   \hyxmp@add@simple{prism:pageRange}{\@pdfpagerange}%
1444   \hyxmp@add@simple{prism:isbn}{\@pdfisbn}%
1445   \hyxmp@add@simple{prism:issn}{\@pdfissn}%
1446   \hyxmp@add@simple{prism:eIssn}{\@pdfeissn}%
1447   \hyxmp@add@simple{prism:doi}{\@pdfdoi}%
1448   \hyxmp@add@simple{prism:url}{\@pdfurl}%
1449   \hyxmp@add@simple{prism:byteCount}{\@pdfbytes}%
1450   \hyxmp@add@simple{prism:pageCount}{\@pdfnumpages}%
1451 }
```

### 3.5.11 XMP extension schemata

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

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

```

1452 \newcommand*{\hyxmp@check@iptc@data}{%

\hyxmp@iptc@data
1453   \edef\hyxmp@iptc@data{%
1454     \@pdfcontactaddress
1455     \@pdfcontactcity
1456     \@pdfcontactregion
1457     \@pdfcontactpostcode
1458     \@pdfcontactcountry
1459     \@pdfcontactphone
1460     \@pdfcontactemail
1461     \@pdfcontacturl
1462   }%
1463 }
```

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

```
1464 \newcommand*{\hyxmp@check@prism@data}{%
```

```
\hyxmp@prism@data
```

```
1465 \edef\hyxmp@prism@data{%
1466   \@pdfbookedition
1467   \@pdfbytes
1468   \@pdfdoi
1469   \@pdfeissn
1470   \@pdfisbn
1471   \@pdfissn
1472   \@pdfissuenum
1473   \@pdfnumpages
1474   \@pdfpagerange
1475   \@pdfpublication
1476   \@pdfpubtype
1477   \@pdfsubtitle
1478   \@pdfurl
1479   \@pdfvolumenum
1480 }%
1481 }%
```

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

```
1482 \newcommand*{\hyxmp@begin@extension@decls}{%
1483   \hyxmp@add@to@xml{%
1484     <pdfaExtension:schemas>^^J%
1485     <rdf:Bag>^^J%
1486   }%
1487 }
```

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

```
1488 \newcommand*{\hyxmp@end@extension@decls}{%
1489   \hyxmp@add@to@xml{%
1490     </rdf:Bag>^^J%
1491     </pdfaExtension:schemas>^^J%
1492   }%
1493 }
```

`\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.

```
1494 \newcommand*{\hyxmp@begin@ext@decl}[3]{%
1495   \hyxmp@add@to@xml{%
1496     <rdf:li rdf:parseType="Resource">^^J%
1497     <pdfaSchema:schema>#1</pdfaSchema:schema>^^J%
1498     <pdfaSchema:prefix>#2</pdfaSchema:prefix>^^J%
1499     <pdfaSchema:namespaceURI>#3</pdfaSchema:namespaceURI>^^J%
1500     <pdfaSchema:property>^^J%
```

```

1501 -----<rdf:Seq>^^J%
1502  }%
1503 }%

```

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

```

1504 \newcommand*{\hyxmp@end@ext@decl}{%
1505   \hyxmp@add@to@xml{%
1506     -----</rdf:Seq>^^J%
1507     -----</pdfaSchema:property>^^J%
1508     -----</rdf:li>^^J%
1509   }%
1510 }%

```

`\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.

```

1511 \newcommand{\hyxmp@declare@property}[4][Text]{%
1512   \hyxmp@add@to@xml{%
1513     -----<rdf:li rdf:parseType="Resource">^^J%
1514     -----<pdfaProperty:name>%
1515     \xdef\hyxmp@xml{\hyxmp@xml#2}%
1516     \hyxmp@add@to@xml{</pdfaProperty:name>^^J%
1517     -----<pdfaProperty:valueType>#1</pdfaProperty:valueType>^^J%
1518     -----<pdfaProperty:category>#3</pdfaProperty:category>^^J%
1519     -----<pdfaProperty:description>#4</pdfaProperty:description>^^J%
1520     -----</rdf:li>^^J%
1521   }%
1522 }%

```

`\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.

```

1523 \newcommand{\hyxmp@declare@field}[3][Text]{%
1524   \hyxmp@add@to@xml{%
1525     -----<rdf:li rdf:parseType="Resource">^^J%
1526     -----<pdfaField:name>#2</pdfaField:name>^^J%
1527     -----<pdfaField:valueType>#1</pdfaField:valueType>^^J%
1528     -----<pdfaField:description>#3</pdfaField:description>^^J%
1529     -----</rdf:li>^^J%
1530   }%
1531 }

```

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

```

1532 \newcommand*{\hyxmp@pdf@extensions}{%
1533   \hyxmp@begin@ext@decl
1534     {Adobe PDF Schema}%
1535     {pdf}%
1536     {http://ns.adobe.com/pdf/1.3/}%
1537   \hyxmp@declare@property

```

```

1538     {Trapped}%
1539     {internal}%
1540     {Indication if the document has been modified to include trapping information}%
1541     \hyxmp@end@ext@decl
1542 }%

```

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

```

1543 \newcommand*{\hyxmp@mm@extensions}{%
1544     \hyxmp@begin@ext@decl
1545         {XMP Media Management Schema}%
1546         {xmpMM}%
1547         {http://ns.adobe.com/xap/1.0/mm/}%
1548     \hyxmp@declare@property
1549         [URI]
1550         {DocumentID}%
1551         {internal}%
1552         {UUID based identifier for all versions and renditions of a document}%
1553     \hyxmp@declare@property
1554         [URI]
1555         {InstanceID}%
1556         {internal}%
1557         {UUID based identifier for specific incarnation of a document}%
1558     \hyxmp@declare@property
1559         {VersionID}%
1560         {internal}%
1561         {Document version identifier}%
1562     \hyxmp@declare@property
1563         {RenditionClass}%
1564         {internal}%
1565         {The manner in which a document is rendered}%
1566     \hyxmp@end@ext@decl
1567 }%

```

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

```

1568 \newcommand*{\hyxmp@pdfa@id@extensions}{%
1569     \hyxmp@begin@ext@decl
1570         {PDF/A Identification Schema}%
1571         {pdfaid}%
1572         {http://www.aiim.org/pdfa/ns/id/}%
1573     \hyxmp@declare@property
1574         [Integer]%
1575         {part}%
1576         {internal}%
1577         {Part of PDF/A standard}%
1578     \hyxmp@declare@property
1579         {conformance}%
1580         {internal}%
1581         {Conformance level of PDF/A standard}%
1582     \hyxmp@end@ext@decl
1583 }%

```

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

```
1584 \newcommand*{\hyxmp@pdfua@id@extensions}{%
1585   \hyxmp@begin@ext@decl
1586     {PDF/UA Universal Accessibility Schema}%
1587     {pdfuaid}%
1588     {http://www.aiim.org/pdfua/ns/id/}%
1589   \hyxmp@declare@property
1590     [Integer]%
1591     {part}%
1592     {internal}%
1593     {Part of ISO 14289 standard}%
1594   \hyxmp@end@ext@decl
1595 }%
```

`\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.

```
1596 \newcommand*{\hyxmp@pdfx@id@extensions}{%
1597   \ifx\hyxmp@pdfx@major\empty
1598   \else
1599     \hyxmp@begin@ext@decl
1600       {Adobe Document Info PDF/X eXtension Schema}%
1601       {pdfx}%
1602       {http://ns.adobe.com/pdfx/1.3/}%
1603     \hyxmp@declare@property
1604       {GTS_PDFXVersion}%
1605       {internal}%
1606       {ID of PDF/X standard}%
1607     \hyxmp@declare@property
1608       {GTS_PDFXConformance}%
1609       {internal}%
1610       {Conformance level of PDF/X standard}%
1611     \hyxmp@end@ext@decl
1612   \fi
```

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

```
1613 \@tempcnta=0\hyxmp@pdfx@major\relax
1614 \ifnum\@tempcnta>3
1615   \hyxmp@begin@ext@decl
1616     {PDF/X ID Schema}%
1617     {pdfxid}%
1618     {http://www.npes.org/pdfx/ns/id/}%
1619   \hyxmp@declare@property
1620     {GTS_PDFXVersion}%
1621     {internal}%
1622     {ID of PDF/X standard}%
1623   \hyxmp@end@ext@decl
1624 \fi
1625 }%
```



`\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.

```

1626 \newcommand*{\hyxmp@iptc@extensions}{%
1627   \hyxmp@begin@ext@decl
1628     {IPTC Core Schema}%
1629     {Iptc4xmpCore}%
1630     {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}%
1631   \hyxmp@declare@property
1632     [ContactInfo]
1633     {CreatorContactInfo}
1634     {external}
1635     {Document creator's contact information}

```

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

```

1636   \hyxmp@add@to+xml{%
1637     _____</rdf:Seq>^^J%
1638     _____</pdfaSchema:property>^^J%
1639     _____<pdfaSchema:valueType>^^J%
1640     _____<rdf:Seq>^^J%
1641     _____<rdf:li rdf:parseType="Resource">^^J%
1642     _____<pdfaType:type>ContactInfo</pdfaType:type>^^J%
1643     _____<pdfaType:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaType:
1644     _____<pdfaType:prefix>Iptc4xmpCore</pdfaType:prefix>^^J%
1645     _____<pdfaType:description>%
1646         Basic set of information to get in contact with a person%
1647     _____</pdfaType:description>^^J%
1648     _____<pdfaType:field>^^J%
1649     _____<rdf:Seq>^^J%
1650   }%
1651   \hyxmp@declare@field
1652     {CiAdrCity}%
1653     {Contact information city}%
1654   \hyxmp@declare@field
1655     {CiAdrCtry}%
1656     {Contact information country}%
1657   \hyxmp@declare@field
1658     {CiAdrExtadr}%
1659     {Contact information address}%
1660   \hyxmp@declare@field
1661     {CiAdrPcode}%
1662     {Contact information local postal code}%
1663   \hyxmp@declare@field
1664     {CiAdrRegion}%
1665     {Contact information regional information such as state or province}%
1666   \hyxmp@declare@field
1667     {CiEmailWork}%
1668     {Contact information email address(es)}%

```

```

1669 \hyxmp@declare@field
1670     {CiTelWork}%
1671     {Contact information telephone number(s)}%
1672 \hyxmp@declare@field
1673     {CiUrlWork}%
1674     {Contact information Web URL(s)}%
1675 \hyxmp@add@to+xml{%
1676 -----</rdf:Seq>^^J%
1677 -----</pdfaType:field>^^J%
1678 -----</rdf:li>^^J%
1679 -----</rdf:Seq>^^J%
1680 -----</pdfaSchema:valueType>^^J%
1681 -----</rdf:li>^^J%
1682 }%
1683 }

```

\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.

```

1684 \newcommand*{\hyxmp@prism@extensions}{%
1685 \hyxmp@begin@ext@decl
1686     {PRISM Basic Metadata}%
1687     {prism}%
1688     {http://prismstandard.org/namespaces/basic/2.1/}%
1689 \hyxmp@declare@property
1690     {complianceProfile}%
1691     {internal}%
1692     {PRISM specification compliance profile to which this document adheres}%
1693 \hyxmp@declare@property
1694     {publicationName}%
1695     {external}%
1696     {Publication name}%
1697 \hyxmp@declare@property
1698     {aggregationType}%
1699     {external}%
1700     {Publication type}%
1701 \hyxmp@declare@property
1702     {bookEdition}%
1703     {external}%
1704     {Edition of the book in which the document was published}%
1705 \hyxmp@declare@property
1706     {volume}%
1707     {external}%
1708     {Publication volume number}%
1709 \hyxmp@declare@property
1710     {number}%
1711     {external}%
1712     {Publication issue number within a volume}%
1713 \hyxmp@declare@property

```

```

1714         {pageRange}%
1715         {external}%
1716         {Page range for the document within the print version of its publication}%
1717 \hyxmp@declare@property
1718         {issn}%
1719         {external}%
1720         {ISSN for the printed publication in which the document was published}%
1721 \hyxmp@declare@property
1722         {eIssn}%
1723         {external}%
1724         {ISSN for the electronic publication in which the document was published}%
1725 \hyxmp@declare@property
1726         {isbn}%
1727         {external}%
1728         {ISBN for the publication in which the document was published}%
1729 \hyxmp@declare@property
1730         {doi}%
1731         {external}%
1732         {Digital Object Identifier for the document}%
1733 \hyxmp@declare@property
1734         [URL]
1735         {url}%
1736         {external}%
1737         {URL at which the document can be found}%
1738 \hyxmp@declare@property
1739         [Integer]
1740         {byteCount}%
1741         {internal}%
1742         {Approximate file size in octets}%
1743 \hyxmp@declare@property
1744         [Integer]
1745         {pageCount}%
1746         {internal}%
1747         {Number of pages in the print version of the document}%
1748 \hyxmp@declare@property
1749         {subtitle}%
1750         {external}%
1751         {Document's subtitle}%
1752 \hyxmp@end@ext@decl
1753 }%

```

`\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.

```

1754 \newcommand*{\hyxmp@declare@extensions}{%

```

```

1755 \hyxmp@begin@extension@decls

```

Declare the Adobe PDF schema (always present).

```

1756 \hyxmp@pdf@extensions

```

Declare the XMP Media Management extensions (always present).

```
1757 \hyxmp@mm@extensions
```

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

```
1758 \ifHy@pdfa
```

```
1759 \hyxmp@pdfa@id@extensions
```

```
1760 \fi
```

Conditionally declare the PDF/UA Universal Accessibility extensions.

```
1761 \ifx\@pdfuapart\@empty
```

```
1762 \else
```

```
1763 \hyxmp@pdfua@id@extensions
```

```
1764 \fi
```

\next Conditionally declare the PDF/X extensions.

```
1765 \ifx\@pdfxversion\@empty
```

```
1766 \else
```

```
1767 \hyxmp@pdfx@id@extensions
```

```
1768 \fi
```

Conditionally declare IPTC photo metadata extensions.

```
1769 \ifx\hyxmp@iptc@data\@empty
```

```
1770 \else
```

```
1771 \hyxmp@iptc@extensions
```

```
1772 \fi
```

Conditionally declare PRISM basic metadata extensions.

```
1773 \ifx\hyxmp@prism@data\@empty
```

```
1774 \else
```

```
1775 \hyxmp@prism@extensions
```

```
1776 \fi
```

```
1777 \hyxmp@end@extension@decls
```

```
1778 }
```

### 3.5.12 Combining schemata into an XMP packet

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

```
1779 \begingroup
```

```
1780 \ifhyxmp@unicodetex
```

```
1781 \lccode'\!="FEFF %
```

```
1782 \lowercase{%
```

```
1783 \gdef\hyxmp@bom{!}
```

```
1784 }%
```

```
1785 \else
```

```
1786 \catcode'\^ef=12
```

```
1787 \catcode'\^bb=12
```

```
1788 \catcode'\^bf=12
```

```
1789 \gdef\hyxmp@bom{^ef^bb^bf}%
```

```
1790 \fi
```

```
1791 \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.

```
1792 \def\hyxmp@construct@packet{%
1793   \gdef\hyxmp+xml{%
1794     \hyxmp@add@to+xml{<?xpacket begin="\hyxmp@bom" %
1795       id="W5M0MPCehiHzreSzNTczkc9d"?>^^J%
1796     <x:xmpmeta xmlns:x="adobe:ns:meta/">^^J%
1797     __<rdf:RDF %
1798       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\hyxmp@hash">^^J%
1799     ____<rdf:Description rdf:about=""^^J%
```

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

```
1800 _____xmlns:pdf="http://ns.adobe.com/pdf/1.3/"^^J%
1801 _____xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/"^^J%
1802 _____xmlns:dc="http://purl.org/dc/elements/1.1/"^^J%
1803 _____xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/"^^J%
1804 _____xmlns:xmp="http://ns.adobe.com/xap/1.0/"^^J%
1805 _____xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"^^J%
1806 _____xmlns:stEvt="http://ns.adobe.com/xap/1.0/sType/ResourceEvent\hyxmp@hash"^^J%
1807 _____xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/"^^J%
1808 _____xmlns:pdfuaid="http://www.aiim.org/pdfua/ns/id/"^^J%
1809 _____xmlns:pdfx="http://ns.adobe.com/pdfx/1.3/"^^J%
1810 _____xmlns:pdfxid="http://www.npes.org/pdfx/ns/id/"^^J%
1811 _____xmlns:prism="http://prismstandard.org/namespaces/basic/2.1/"^^J%
1812 _____xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"^^J%
1813 _____xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"^^J%
1814 _____xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema\hyxmp@hash"^^J%
1815 _____xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property\hyxmp@hash"^^J%
1816 _____xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type\hyxmp@hash"^^J%
1817 _____xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field\hyxmp@hash">^^J%
1818   }%
```

Declare non-standard schemata.

```
1819   \hyxmp@check@iptc@data
1820   \hyxmp@check@prism@data
1821   \hyxmp@declare@extensions
```

Insert all the metadata we know how to insert.

```
1822   \hyxmp@pdf@schema
1823   \hyxmp@xmpRights@schema
1824   \hyxmp@dc@schema
1825   \hyxmp@photoshop@schema
1826   \hyxmp@xmp@basic@schema
1827   \hyxmp@pdfa@id@schema
1828   \hyxmp@pdfua@id@schema
1829   \hyxmp@pdfx@id@schema
1830   \hyxmp@mm@schema
1831   \hyxmp@iptc@schema
1832   \hyxmp@prism@schema
1833   \hyxmp@add@to+xml{%
```

```

1834 ____</rdf:Description>^^J%
1835 __</rdf:RDF>^^J%
1836 </x:xmpmeta>^^J%
1837 \hyxmp@padding
1838 <?xpacket end="w"?>^^J%
1839 }%
1840 }

```

### 3.6 Embedding the XMP packet

The PDF specification says that “a metadata stream may be attached to a document through the Metadata entry in the document catalogue” [3] so that’s what we do here.

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

```

1841 \newcommand*{\hyxmp@embed@packet}{%
1842   \hyxmp@construct@packet
1843   \def\hyxmp@driver{hpdfTeX}%
1844   \ifx\hyxmp@driver\Hy@driver
1845     \hyxmp@embed@packet@pdfTeX
1846   \else
1847     \def\hyxmp@driver{hLaTeX}%
1848     \ifx\hyxmp@driver\Hy@driver
1849       \hyxmp@embed@packet@LaTeX
1850     \else
1851       \def\hyxmp@driver{hdvipdfm}%
1852       \ifx\hyxmp@driver\Hy@driver
1853         \hyxmp@embed@packet@dviPDFM
1854       \else
1855         \def\hyxmp@driver{hXeTeX}%
1856         \ifx\hyxmp@driver\Hy@driver
1857           \hyxmp@embed@packet@XeTeX
1858         \else
1859           \@ifundefined{pdfmark}{%
1860             \PackageWarningNoLine{hyperxmp}{%
1861               Unrecognized hyperref driver ‘\Hy@driver’.\MessageBreak
1862               \jobname.tex’s XMP metadata will *not* be.\MessageBreak
1863               embedded in the resulting file}%
1864           }{%
1865             \hyxmp@embed@packet@pdfmark
1866           }%
1867         \fi
1868       \fi
1869     \fi
1870   \fi
1871 }

```

### 3.6.1 Embedding using pdfTeX

Up to version 0.85, LuaTeX supported the pdfTeX primitives, and `hyperref` didn't distinguish the two backends. However, from `hyperxmp`'s perspective there is one key difference: the effect of `\pdfcompresslevel` is local to a group in pdfTeX but is global in LuaTeX.

The PDF object representing the XMP packet is supposed to include an uncompressed stream so it can be read by non-PDF-aware tools. However, we don't want to unnecessarily uncompress *every* PDF stream. The solution, provided by Hans Hagen on the `luatex` mailing list (thread: "Leaving a single PDF object uncompressed", 6 JUL 2016), is to provide the `uncompressed` flag to `\pdfobj`. Our definition of `\hyxmp@embed@packet@pdftex` uses the `ifluatex` package to distinguish the pdfTeX case from the pre-0.85 LuaTeX case.

```
1872 \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.

1873 \newcommand*{\hyxmp@embed@packet@pdftex}{%
1874   \bgroup
1875     \ifluatex
1876     \else
1877       \pdfcompresslevel=0
1878     \fi
1879     \immediate\pdfobj \ifluatex uncompressed\fi stream attr {%
1880       /Type /Metadata
1881       /Subtype /XML
1882     }{\hyxmp@xml}%
1883     \pdfcatalog {/Metadata \the\pdflastobj\space 0 R}%
1884   \egroup
1885 }
```

### 3.6.2 Embedding using LuaTeX 0.85+

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

1886 \newcommand*{\hyxmp@embed@packet@luatex}{%
1887   \immediate\pdfextension obj uncompressed stream attr {%
1888     /Type /Metadata
1889     /Subtype /XML
1890   }{\hyxmp@xml}%
1891   \pdfextension catalog {/Metadata \the\numexpr\pdffeedback lastobj\relax\space 0 R}%
1892 }
```

### 3.6.3 Embedding using any pdfmark-based backend

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

```

1893 \newcommand*{\hyxmp@embed@packet@pdfmark}{%
1894   \pdfmark{%
1895     pdfmark=/NamespacePush
1896   }%
1897   \pdfmark{%
1898     pdfmark=/OBJ,
1899     Raw={/_objdef \string{hyxmp@Metadata\string} /type /stream}%
1900   }%
1901   \pdfmark{%
1902     pdfmark=/PUT,
1903     Raw={\string{hyxmp@Metadata\string}
1904       2 dict begin
1905         /Type /Metadata def
1906         /Subtype /XML def
1907         currentdict
1908       end
1909     }%
1910   }%
1911   \pdfmark{%
1912     pdfmark=/PUT,
1913     Raw={\string{hyxmp@Metadata\string} (\hyxmp@xml)}%
1914   }%
1915   \pdfmark{%
1916     pdfmark=/Metadata,
1917     Raw={\string{Catalog\string} \string{hyxmp@Metadata\string}}%
1918   }%
1919   \pdfmark{%
1920     pdfmark=/NamespacePop
1921   }%
1922 }
```

### 3.6.4 Embedding using dvipdfm

`\hyxmp@embed@packet@dvipdfm` Embed the XMP packet using `dvipdfm`-specific `\special` commands. Note that `dvipdfm` rather irritatingly requires us to count the number of characters in the `\hyxmp@xml` stream ourselves.

```

1923 \newcommand*{\hyxmp@embed@packet@dvipdfm}{%
1924   \hyxmp@string@len{\hyxmp@xml}%
1925   \special{pdf: object @hyxmp@Metadata
1926     <<
1927       /Type /Metadata
1928       /Subtype /XML
1929       /Length \the\@tempcnta
1930     >>
```



```

1931     stream^^J\hyxmp@xml endstream%
1932 }%
1933 \special{pdf: docview
1934     <<
1935     /Metadata @hyxmp@Metadata
1936     >>
1937 }%
1938 }

```

`\hyxmp@string@len` Set `\@tempcnta` to the number of characters in a given string (`#1`). The approach is first to tally the number of space characters then to tally the number of non-space characters. While this is rather sloppy I haven't found a better way to achieve the same effect, especially given that all of the characters in `#1` have already been assigned their category codes.

```

1939 \newcommand*{\hyxmp@string@len}[1]{%
1940     \@tempcnta=0
1941     \expandafter\hyxmp@count@spaces#1 {} %
1942     \expandafter\hyxmp@count@non@spaces#1{}%
1943 }

```

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

```

1944 \def\hyxmp@count@spaces#1 {%
1945     \def\hyxmp@one@token{#1}%
1946     \ifx\hyxmp@one@token\empty
1947         \advance\@tempcnta by -1
1948     \else
1949         \advance\@tempcnta by 1
1950     \expandafter\hyxmp@count@spaces
1951     \fi
1952 }

```

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

```

1953 \newcommand*{\hyxmp@count@non@spaces}[1]{%
1954     \def\hyxmp@one@token{#1}%
1955     \ifx\hyxmp@one@token\empty
1956     \else
1957         \advance\@tempcnta by 1
1958     \expandafter\hyxmp@count@non@spaces
1959     \fi
1960 }

```

### 3.6.5 Embedding using $\text{\XeTeX}$

`\hyxmp@embed@packet@xetex` Embed the XMP packet using `xdvipdfmx`-specific `\special` commands. I don't know how to tell `xdvipdfmx` always to leave the Metadata stream uncompressed,

so the XMP metadata is likely to be missed by non-PDF-aware XMP viewers.

```

1961 \newcommand*{\hyxmp@embed@packet@xetex}{%
1962   \special{pdf:stream @hyxmp@Metadata (\hyxmp@xml)
1963     <<
1964       /Type /Metadata
1965       /Subtype /XML
1966     >>
1967   }%
1968   \special{pdf:put @catalog
1969     <<
1970       /Metadata @hyxmp@Metadata
1971     >>
1972   }%
1973 }
```

### 3.7 Final clean-up

Having saved the category code of “ ” at the start of the package code (Section 3.1), we now restore that character’s original category code.

```

1974 \catcode'\="\hyxmp@dq@code
```

## 4 Help Wanted

**Comma handling** Ideally, `\xmpquote` should automatically replace all commas with `\xmpcomma`. Unfortunately, my  $\text{\TeX}$  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  $\text{\TeX}$  engines (`pdf $\text{\TeX}$` , `Lua $\text{\TeX}$` , `X $\text{\TeX}$` , etc.), please send me a code patch.

## A Sample XMP Packet

The following is an example of a complete XMP packet as may be produced by `hyperxmp`. This packet corresponds to the metadata included in the sample  $\text{\LaTeX}$  document presented on pages 9–10. For clarity, metadata values, either specified explicitly by the document or introduced automatically by `hyperxmp`, are colored blue.

```

<?xpacket begin="\357\273\277" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/">
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about=""
      xmlns:pdf="http://ns.adobe.com/pdf/1.3/"
      xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/"
```

```

xmlns:xmp="http://ns.adobe.com/xap/1.0/"
xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"
xmlns:stEvt="http://ns.adobe.com/xap/1.0/sType/ResourceEvent#"
xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/"
xmlns:pdfuaid="http://www.aiim.org/pdfua/ns/id/"
xmlns:pdfx="http://ns.adobe.com/pdfx/1.3/"
xmlns:pdfxid="http://www.npes.org/pdfx/ns/id/"
xmlns:prism="http://prismstandard.org/namespaces/basic/2.1/"
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>
  pdfTeX, Version 3.14159265-2.6-1.40.20 (TeX Live 2019/Debian)
</pdf:Producer>
<pdf:PDFVersion>1.5</pdf:PDFVersion>
<xmpRights:Marked>True</xmpRights:Marked>
<xmpRights:WebStatement>
  http://creativecommons.org/licenses/by-nc-nd/3.0/
</xmpRights:WebStatement>
<dc:format>application/pdf</dc:format>
<dc:title>
  <rdf:Alt>
    <rdf:li xml:lang="en">
      On a heuristic viewpoint concerning the production
      and transformation of light
    </rdf:li>
    <rdf:li xml:lang="x-default">
      On a heuristic viewpoint concerning the production
      and transformation of light
    </rdf:li>
  </rdf:Alt>
</dc:title>

```

```

        <rdf:li xml:lang="de">
            Über einen die Erzeugung und Verwandlung des Lichtes
            betreffenden heuristischen Gesichtspunkt
        </rdf:li>
    </rdf:Alt>
</dc:title>
<dc:description>
    <rdf:Alt>
        <rdf:li xml:lang="en">photoelectric effect</rdf:li>
        <rdf:li xml:lang="x-default">photoelectric effect</rdf:li>
    </rdf:Alt>
</dc:description>
<dc:rights>
    <rdf:Alt>
        <rdf:li xml:lang="en">
            Copyright (C) 1905, Albert Einstein
        </rdf:li>
        <rdf:li xml:lang="x-default">
            Copyright (C) 1905, Albert Einstein
        </rdf:li>
    </rdf:Alt>
</dc:rights>
<dc: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>2019-03-16T23:07:38-06:00</xmp:CreateDate>
<xmp:ModifyDate>2019-03-16T23:07:38-06:00</xmp:ModifyDate>
<xmp:MetadataDate>2019-03-16T23:07:38-06:00</xmp:MetadataDate>
<xmp:CreatorTool>LaTeX with hyperref package</xmp:CreatorTool>
<xmpMM:DocumentID>
    uuid:6d1ac9ec-4ff2-515a-954b-648eeb4853b0
</xmpMM:DocumentID>
<xmpMM:InstanceID>
    uuid:3e4c4182-b182-46c9-995f-754c41d13390
</xmpMM:InstanceID>
<xmpMM:VersionID>2.998e8</xmpMM:VersionID>
<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-
    </prism:url>
    <prism:byteCount>59846</prism:byteCount>
    <prism:pageCount>17</prism:pageCount>
  </rdf:Description>
</rdf:RDF>
</x:xmpmeta>
<?xpacket end="w"?>

```

## References

- [1] Adobe Systems, Inc., San Jose, California. *Adobe Acrobat X SDK Help, pdfmark Reference*. Available from <http://www.adobe.com/devnet/acrobat/documentation.html>.
- [2] Adobe Systems, Inc. *PostScript Language Reference Manual*. Addison-Wesley, 2nd edition, January 1996, ISBN: 0-201-18127-4.
- [3] Adobe Systems, Inc., San Jose, California. *Document Management—Portable Document Format—Part 1: PDF 1.7*, July 2008. ISO 32000-1 standard document. Available from [http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000\\_2008.pdf](http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000_2008.pdf).
- [4] Adobe Systems, Inc., San Jose, California. *XMP Specification Part 1: Data model, Serialization, and Core Properties*, April 2012. Available from <http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/xmp/pdfs/cc-201306/XMPSpecificationPart1.pdf>.
- [5] DCMI Usage Board *DCMI Metadata Terms*, June 14, 2012. Available from <http://dublincore.org/documents/dcmi-terms/>.
- [6] Michael Downes. Around the bend #15, answers, 4th (last) installment. `comp.text.tex` newsgroup posting, January 3, 1994. Archived by Google at <http://groups.google.com/group/comp.text.tex/msg/7da7643b9e8f3b48>.
- [7] International Digital Enterprise Alliance, Inc. *Publishing Requirements for Industry Standard Metadata, Version 3.0: PRISM Basic Metadata Specification*, October 12, 2012. Available from [http://www.prismstandard.org/specifications/3.0/PRISM\\_Basic\\_Metadata\\_3.0.htm](http://www.prismstandard.org/specifications/3.0/PRISM_Basic_Metadata_3.0.htm).

- [8] International Digital Enterprise Alliance, Inc. *Publishing Requirements for Industry Standard Metadata, Version 3.0: PRISM Controlled Vocabularies Specification*, October 4, 2012. Available from [http://www.prismstandard.org/specifications/3.0/PRISM\\_CV\\_Spec\\_3.0.pdf](http://www.prismstandard.org/specifications/3.0/PRISM_CV_Spec_3.0.pdf).
- [9] International Press Telecommunications Council. *IPTC Photo Metadata: Core 1.1/Extension 1.1*, July 2010. Revision 1. Available from [http://www.iptc.org/std/photometadata/specification/IPTC-PhotoMetadata-201007\\_1.pdf](http://www.iptc.org/std/photometadata/specification/IPTC-PhotoMetadata-201007_1.pdf).
- [10] Internet Assigned Numbers Authority. Language subtag registry, January 11, 2011. Available from <http://www.iana.org/assignments/language-subtag-registry>.
- [11] Paul J. Leach, Michael Mealling, and Rich Salz. A Universally Unique IDentifier (UUID) URN namespace. Request for Comments 4122, Internet Engineering Task Force, Network Working Group, July 2005. Category: Standards Track. Available from <http://www.ietf.org/rfc/rfc4122.txt>.
- [12] PDF/A Competence Center, Berlin, Germany. *TechNote 0008: Predefined XMP Properties in PDF/A-1*, March 20, 2008. Available from [http://www.pdfa.org/wp-content/uploads/2011/08/tn0008\\_predefined\\_xmp\\_properties\\_in\\_pdfa-1\\_2008-03-20.pdf](http://www.pdfa.org/wp-content/uploads/2011/08/tn0008_predefined_xmp_properties_in_pdfa-1_2008-03-20.pdf).
- [13] PDF/A Competence Center, Berlin, Germany. *TechNote 0009: XMP Extension Schemas in PDF/A-1*, March 20, 2008. Available from [http://www.pdfa.org/wp-content/uploads/2011/08/tn0009\\_xmp\\_extension\\_schemas\\_in\\_pdfa-1\\_2008-03-20.pdf](http://www.pdfa.org/wp-content/uploads/2011/08/tn0009_xmp_extension_schemas_in_pdfa-1_2008-03-20.pdf).
- [14] Misha Wolf and Charles Wicksteed. Date and time formats. Note NOTE-datetime, World Wide Web Consortium (W3C), September 15, 1997. Available from <http://www.w3.org/TR/NOTE-datetime>.

## Change History

v1.0	X <sub>Y</sub> T <sub>E</sub> X backend (xdvipdfmx) ..	1
General: Initial version .....	Added support for the	1
v1.1	Photoshop schema .....	1
\hyxmp@construct@packet:	Made the package compatible	
Explicitly set the category	with <code>ngerman</code> . Thanks to	
codes of characters $\langle EF \rangle$ , $\langle BB \rangle$ ,	Tobias Mueller for the bug	
and $\langle BF \rangle$ to “letter”. Thanks to	report. ....	17
Daniel Schömer for the bug		
report .....	v1.3	
77	\hyxmp@auto@assign@data:	
v1.2	Introduced the <code>pdfmetalang</code>	
General: Added support for the	package option, which enables	

an author to specify the language in which he wrote the document's metadata . . . . .	31	\hyxmp@is@unicode: Added by Heiko Oberdiek . . . . .	46
v1.4		\hyxmp@list@to@xml: Modified by Heiko Oberdiek to use the new Unicode-processing macros . .	61
\hyxmp@mm@schema: Renamed the xapMM namespace prefix to xmpMM . . . . .	64	\hyxmp@photoshop@schema: Simplified using \hyxmp@add@simple . . . . .	65
\hyxmp@rdf@dc: Included metadata in the x-default language regardless of the specified metadata language . . . . .	60	\hyxmp@skiptorelax: Added by Heiko Oberdiek . . . . .	49
\hyxmp@xmpRights@schema: Renamed the xapRights namespace prefix to xmpRights	63	\hyxmp@skipzeros: Added by Heiko Oberdiek . . . . .	48
v1.5		\hyxmp@toxml: Added by Heiko Oberdiek . . . . .	47
General: Made the XMP inclusion more robust. Thanks to Heiko Oberdiek for the bug report and suggested modifications. .	17	Escaped parentheses written with pdfmarks to prevent dvips from line-wrapping the XMP packet . . . . .	47
v2.0		\hyxmp@toxml@unicodetex: Added by Heiko Oberdiek . . . . .	48
\ProcessKeyvalOptions: Added this macro . . . . .	28	\hyxmp@xetex@crap: Added by Heiko Oberdiek . . . . .	48
\XMPTruncateList: Added this macro . . . . .	38	\hyxmp@xmlify: Completely rewritten by Heiko Oberdiek to better support Unicode-enabled T <sub>E</sub> X programs . . . . .	45
\hyxmp@ProcessKeyvalOptions: Added this macro . . . . .	28	\hyxmp@xmp@basic@schema: Added this macro . . . . .	64
\hyxmp@SpaceOther: Added by Heiko Oberdiek . . . . .	49	\hyxmp@xmpRights@schema: Modified to include xmpRights:Marked only when pdfcopyright is specified and xmpRights:WebStatement only when pdflicenseurl is specified .	63
\hyxmp@add@simple: Added this macro . . . . .	51	\hyxmp@zero: Added by Heiko Oberdiek . . . . .	50
\hyxmp@add@to@xml: Updated also to replace commas . . . . .	57	\ifhyxmp@unicodetex: Added by Heiko Oberdiek . . . . .	45
\hyxmp@auto@assign@data: New \AtBeginDocument code from Heiko Oberdiek to properly encode \pdfmetalang . . . . .	31	\xmpcomma: Added this macro . . .	38
\hyxmp@bom: Added by Heiko Oberdiek . . . . .	76	\xmpquote: Added this macro . . .	38
\hyxmp@comma: Added this macro	38	General: Added support for the XMP Basic schema and miscellaneous other bits of metadata . . . . .	1
\hyxmp@construct@packet: Modified by Heiko Oberdiek to use an appropriate BOM representation via \hyxmp@bom	77	Heiko Oberdiek's major rewrite of the code to better support native-Unicode T <sub>E</sub> X implementations (X <sub>Y</sub> T <sub>E</sub> X and LuaT <sub>E</sub> X) . . . . .	1
\hyxmp@crap@convert: Added by Heiko Oberdiek . . . . .	49		
\hyxmp@crap@test: Added by Heiko Oberdiek . . . . .	49		
\hyxmp@dc@schema: Added support for dc:language and dc:source .	62		



v2.1		individual item so as to conform to the latest XMP specifications, a detail identified by Florian Breitwieser . . . . .	62
	\hypersetup: Added this macro . . . . .		28
	\hyxmp@hypersetup: Added this macro . . . . .		28
	\hyxmp@redefine@Hyp: Added this macro . . . . .	\hyxmp@parse@time: Added this macro . . . . .	40
	General: Enabled hyperxmp and hyperref to be loaded in either order. This addresses a bug report by Yury Donskoy . . . . .	\hyxmp@parse@tz: Added this macro . . . . .	40
		\hyxmp@parse@tz@char: Added this macro . . . . .	40
v2.2		\hyxmp@pdf@schema: Made \hyxmp@pdf@schema <i>always</i> include the Adobe PDF schema, even when empty. Florian Breitwieser noted that this is necessary for PDF/A-1b compliance . . . . .	60
	\hyxmp@iptc@extensions: Added this macro to support PDF/A generation . . . . .	\hyxmp@pdf@to@xmp@date: Added this macro . . . . .	39
	\hyxmp@iptc@schema: Added this macro . . . . .	\hyxmp@pdfa@id@schema: Added this macro . . . . .	65
	\hyxmp@list@to@lines: Added this macro . . . . .	\hyxmp@today@xmp: Modified the code to parse the time and timezone from \pdfcreationdate, as proposed by Florian Breitwieser . . . . .	43
	\xmpcomma: Changed the default from \relax to an ordinary comma . . . . .	\hyxmp@today@xmp@define: Added this macro . . . . .	42
	\xmplinesep: Added this macro . . . . .	\hyxmp@xmp@to@pdf@date: Added this macro . . . . .	40
	General: Added support for the IPTC Photo Metadata schema . . . . .	\xmptilde: Added this macro . . . . .	38
v2.3		General: Added support for the PDF/A Identification schema, as requested by Florian Breitwieser . . . . .	1
	\hyxmp@iptc@extensions: Gave the lptc4xmpCore:CreatorContactInfo fields a unique pdfaType:prefix to better support conversion of the document to PDF/A . . . . .		
v2.3a			
	General: Bug fix: Redefine \@pdflang as \@empty when hyperref has set it to \relax . . . . .		
v2.3b		v2.5	
	\XMPTruncateList: Made all definitions local to avoid spurious Too many unprocessed floats errors when running with memoir . . . . .	\hyxmp@add@to@xml: Updated also to replace underscores and to modify only the text being added, not the already-modified text . . . . .	57
v2.4		\hyxmp@textunderscore: Added this macro . . . . .	19
	\hyxmp@add@simple@var: Added this macro . . . . .	\hyxmp@uscore: Added this macro . . . . .	38
	\hyxmp@create@uuid: Modified this macro to produce a proper version 4 (random or pseudorandom) UUID . . . . .	General: Enabled “\_” to work within email addresses, as requested by Leonid Sinev . . . . .	1
	\hyxmp@dc@schema: Made dc:language a Bag instead of an	v2.6	
		General: Added support for a new pdfdate key to explicitly specify	

the document date (and optionally time) . . . . .	1	Robert Schlicht, Leonid Sinev, and David Carlisle for bug reports and to Leonid Sinev for helping test the new hyperxmp code . . . . .	1
v2.7		v3.1	
\hyxmp@auto@assign@data:		\hyxmp@embed@packet@luatex:	
Automatically use \title and \author if pdftitle and pdfauthor are left unspecified. Thanks to Maciej Radziejewski for the suggestion . . . . .	31	Updated to use \pdfextension obj uncompressed as suggested by Hans Hagen . . . . .	79
v2.8		\hyxmp@embed@packet@pdftex:	
\hyxmp@add@to+xml: Corrected inadvertent lowercasing of non-Latin characters when run under X <sub>q</sub> L <sup>A</sup> T <sub>E</sub> X or Lua <sup>A</sup> T <sub>E</sub> X (bug reported by Leonid Sinev) . . . . .	57	Leave the XMP packet—and only the XMP packet—uncompressed in both pdfT <sub>E</sub> X and pre-0.85 LuaT <sub>E</sub> X . . . . .	79
v2.9		v3.2	
\hyxmp@iptc@schema: Use lptc4xmpCore instead of lptc4ContInfo as the contact-information metadata prefix. Leonid Sinev reports that Acrobat's PDF/A validator seems to prefer lptc4xmpCore . . . . .	67	\hyxmp@as@pdf@date: Added this macro . . . . .	40
\hyxmp@pdfa@id@schema: Let the author specify the PDF/A part and conformance IDs, as requested by Leonid Sinev . . . . .	65	\hyxmp@as@xmp@date: Added this macro . . . . .	39
General: Force inclusion of dc:creator, dc:title, and dc:description—even if empty—when hyperref is loaded with the pdfa option (suggested by Leonid Sinev) . . . . .	1	\hyxmp@today@xmp@define:	
Introduced the pdftype package option, which enables an author to specify the type of document being produced . . . . .	1	Modified to include hours and minutes . . . . .	42
v3.0		\hyxmp@xmp@basic@schema: Honor hyperref's pdfcreationdate and pdfmoddate options plus a new pdfmetadate option. Leonid Sinev requested this additional control and helped test the resulting hyperxmp code . . . . .	64
\hyxmp@embed@packet@luatex:		v3.3	
Added this macro . . . . .	79	\@pdfsource: Added this macro and the corresponding pdfsource option, at Niklas Beisert's request . . . . .	22
\hyxmp@today@xmp@define:		\XMPLangAlt: Added this macro based on a request—and some code—by Niklas Beisert to support metadata expressed in multiple languages . . . . .	53
Modified to accept the name of a macro to define . . . . .	42	\hyxmp@auto@assign@data: Don't overwrite an existing pdfmetalang with pdflang or x-default. This addresses a bug report by Niklas Beisert . . . . .	31
\hyxmp@xmp@basic@schema: Made the XMP xmp:CreateDate, xmp:ModifyDate, and xmp:MetadataDate match the PDF CreationDate . . . . .	64	\hyxmp@rdf@dc: Bug fix: Output the metadata language as correct XML even when hyperref is loaded with the unicode option . . . . .	60
General: Made the code compatible with LuaT <sub>E</sub> X 0.85. Thanks to			

v3.4			
	<code>\hyxmp@seed@string</code> : Correctly handle an author field of all spaces. Bug reported by Gaëtan Leurent	56	
	General: Use <code>ifmtarg</code> to test for empty arguments, including non-empty but all spaces	1	
v3.5			
	<code>\hyxmp@DocumentID</code> : Added the <code>pdfdocumentid</code> option, at Michael Osipov's request	22	
	<code>\hyxmp@InstanceID</code> : Added the <code>pdfinstanceid</code> option, at Michael Osipov's request	22	
	<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	64	
	<code>\hyxmp@seed@string</code> : Seed with the T <sub>E</sub> X timestamp in addition to the document-specified timestamp	56	
v4.0			
	<code>\XMPTruncateList</code> : Deprecated this macro	38	
	<code>\hyxmp@add@simple@lang</code> : Added this macro	51	
	<code>\hyxmp@begin@ext@decl</code> : Added this macro	69	
	<code>\hyxmp@declare@field</code> : Replaced <code>\hyxmp@declare@resource</code> with this macro	70	
	<code>\hyxmp@declare@property</code> : Added this macro	70	
	<code>\hyxmp@end@ext@decl</code> : Added this macro	70	
	<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>	72	
	Rewrote to more closely honor the XMP specification	72	
	<code>\hyxmp@iptc@schema</code> : Moved the definition of <code>\hyxmp@iptc@data</code>		
	from here into <code>\hyxmp@check@iptc@data</code>	67	
	Renamed this macro to <code>\hyxmp@iptc@schema</code> from <code>\hyxmp@photometa@schema</code>	67	
	Rewrote this macro entirely to correct the use of fields within a structure	67	
	<code>\hyxmp@mm@extensions</code> : Added this macro	71	
	<code>\hyxmp@mm@schema</code> : Include <code>xmpMM:VersionID</code> in the XMP packet	64	
	<code>\hyxmp@no@info@lists</code> : Added this macro	25	
	<code>\hyxmp@pdfa@id@extensions</code> : Added this macro	71	
	<code>\hyxmp@prism@extensions</code> : Added this macro	74	
	<code>\hyxmp@prism@schema</code> : Added this macro	68	
	General: Include all metadata within a single <code>rdf:Description</code> block	1	
v4.1			
	<code>\hyxmp@singleton@dc</code> : Added this macro	62	
	General: Invoke <code>\hyxmp@no@info@lists</code> at the beginning of the document, for compatibility with both newer and older versions of <code>hyperref</code>	36	
	Updated the documentation to refer to <code>\pdfnumpages</code> by its correct name. Thanks to Volker RW Schaa for catching the discrepancy	1	
v5.0			
	<code>\@pdfrendition</code> : Added the <code>pdfrendition</code> option	22	
	<code>\@pdfxstandard</code> : Added this macro	21	
	<code>\hyxmp@add@simple</code> : Insert the tag name (#1) verbatim	51	
	<code>\hyxmp@check@standards</code> : Added this macro	29	
	<code>\hyxmp@check@std</code> : Added this macro	21	
	<code>\hyxmp@declare@property</code> : Insert the property name (#2) verbatim	70	

\hyxmp@define@pdfproducer:	Robin Schwab for the bug
Added this macro . . . . . 58	report . . . . . 58
\hyxmp@no@info@lists: Renamed	\hyxmp@timestamp: Don't rely on
this macros from	\jobname.aux existing to query
\hyxmp@suppress@pdf@metadata	the current time under Xe <sub>La</sub> TeX.
and rewrote it to replace, if	Instead, use \jobname.log.
possible, only Author and	Thanks to Ulrike Fischer for
Keywords . . . . . 25	the bug report and for her
\hyxmp@pdf@extensions: Added	suggestion to use the log file. . 43
this macro . . . . . 70	v5.2
\hyxmp@pdf@schema: Honor	\hyxmp@add@simple@pfx: Added
pdftrapped . . . . . 60	this macro . . . . . 52
\hyxmp@pdfua@id@extensions:	\hyxmp@assign@major@minor:
Added this macro . . . . . 72	Added this macro. hyperxmp
\hyxmp@pdfua@id@schema: Added	now correctly specifies
this macro . . . . . 65	pdf:PDFVersion when
\hyxmp@pdfx@id@extensions:	generating PDF 2.0+. Thanks
Added this macro . . . . . 72	to Ulrike Fischer for alerting
\hyxmp@pdfx@id@schema: Added	me to PDF 2.0's availability in
this macro . . . . . 65	the T <sub>E</sub> X ecosystem and
\hyxmp@today@pdf: Added this	informing me how to activate it 59
macro . . . . . 44	\hyxmp@cond@dc@identifier:
\hyxmp@today@xmp: Support	Added this macro . . . . . 62
Xe <sub>La</sub> TeX's \filemoddate . . . . 43	\ifdraft: Define \ifdraft only
\hyxmp@today@xmp@define:	locally, at Niklas Beisert's
Modified to specify UTC . . . . 42	request . . . . . 22
General: Added support for	General: Introduced the
PDF/UA standards, as requested	pdfidentifier package option,
by Robin Schwab . . . . . 1	which enables an author to
Added support for PDF/X	specify a unique identifier for
standards, as requested by	the document . . . . . 1
Robin Schwab . . . . . 1	v5.3
Define a default producer . . . . 59	\@if@def@and@nonempty: Added
Don't set any document dates	this macro . . . . . 18
(creation, modification, or	\hyxmp@at@end: Use
metadata) from pdfdate . . . . . 1	\AtEndDocument in all T <sub>E</sub> X
v5.1	back ends that provide it.
\hyxmp@banner@to@producer:	Thanks to Nelson Posse Lago
Prevent the category code of	for pointing out why atenddvi is
"@" from propagating past the	best avoided if possible . . . . 18
\begin{document}. Thanks to	\hyxmp@auto@assign@data:
Robert Schlicht for noticing this	Consider other author-provided
catcode "leak" and providing a	sources of metadata. Thanks to
correction . . . . . 58	Robin Schwab for proposing
\hyxmp@define@pdfproducer:	that hyperxmp use the KOMA
Check for Lua <sub>TeX</sub> before	letter classes's metadata . . . . 31
checking for pdf <sub>TeX</sub> to work	\hyxmp@dc@schema: Include all
around luatex85's confusing	languages used in the document
iftex by defining	in dc:language . . . . . 63
\pdfTeXversion. Thanks to	

\hyxmp@parse@acmart: Added this macro .....	32	General: Acquire the default language from the polyglossia package, if loaded. Thanks to Robin Schwab for bringing that package to my attention ....	35
\hyxmp@set@koma@phones: Added this macro .....	30		
\hyxmp@use@first@valid: Added this macro .....	30		

## 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		
\# .....	1134, 1385	\@pdfaformance .. ..... <u>74</u> , 324, 1361
\& .....	810, 842, 1384	\@pdfapart ..... <u>69</u> , 317, 323, 329, 1360
\@acmBooktitle ....	486	\@pdfauthor 198, <u>222</u> , 268, 391, 1099, 1107
\@acmConference ...	487	\@pdfauthortitle .. 58, 269, 1354, 1355
\@acmDOI ..	459, 460, 462	\@pdfbookedition .. <u>147</u> , 270, 1440, 1466
\@acmISBN ..	468, 469, 472	\@pdfbytes ..... <u>137</u> , 271, 1449, 1467
\@acmNumber .....	501	\@pdfcaptionwriter .. <u>60</u> , 272, 1354, 1356
\@acmVolume .....	498	\@pdfcontactaddress ..... <u>165</u> , 273, 424, 1417, 1454
\@author .....	393, <u>415</u>	\@pdfcontactcity <u>173</u> , 274, 430, 1418, 1455
\@baseurl ...	267, 1351	\@pdfcontactcountry ..... <u>179</u> , 275, 442, 1421, 1458
\@elt <u>572</u> , <u>1245</u> , 1396, <u>1401</u>		\@pdfcontactemail .. ..... <u>183</u> , 276, 395, 418, 1424, 1460
\@elt@first .....	<u>1394</u>	\@pdfcontactphone .. ..... <u>181</u> , 277, 399, 1423, 1459
\@elt@rest ..	1396, <u>1398</u>	\@pdfcontactpostcode ..... <u>177</u> , 278, 448, 1420, 1457
\@if@def@and@nonempty ..... <u>19</u> , 344, 345, 351, 458, 467		\@pdfcontactregion .. ..... <u>175</u> , 279, 436, 1419, 1456
\@ifclassloaded ...	504	\@pdfcontacturl <u>185</u> , 280, 402, 1425, 1461
\@ifmtarg .....	17, 975	\@pdfcopyright .... ..... <u>52</u> , 281, 1283, 1311, 1317
\@ifmtargexp .....		\@pdfcreationdate .. ..... <u>282</u> , 522, 523, 1333, 1337
... <u>17</u> , 21, 317, 334, 358, 522, 711, 1200, 1235, 1295, 1325, 1326, 1333, 1339, 1345		\@pdfcreator ..... 1350
\@ifnextchar .....	970	\@pdfdatetime .. <u>30</u> , 283
\@ifnotmtarg .....		\@pdfdoi <u>157</u> , 284, 461, 1297, 1447, 1468
.. 18, 65, 964, 973		\@pdffeissn . <u>143</u> , 285, 1298, 1446, 1469
\@ifnotmtargexp ...		\@pdfidentifier ... <u>161</u> , 286, 1295, 1302
..... <u>17</u> , 333, 370, 946, 987, 1261, 1274, 1389		\@pdfisbn ..... . <u>145</u> , 287, 474, 1300, 1444, 1470
\@journalName .....	485	\@pdfissn . <u>141</u> , 288, 1299, 1445, 1471
		\@pdfissuenum .. <u>153</u> , 289, 500, 1442, 1472
		\@pdfkeywords ..... .... 204, <u>243</u> , 290
		\@pdflang ..... . 291, 380, 383, 506, 510, 512, 514
		\@pdflicenseurl ... <u>56</u> , 292, 1307, 1321
		\@pdfmetadatetitle .. <u>41</u> , 293, 1345, 1348
		\@pdfmetalang ..... .. <u>62</u> , 379, 381,

383, 386, 1211, 1213		
\@pdfmoddate . . . . .	ACM . . . . . 15, 34	DCMI . . . . . 8
... 294, 1339, 1343	acmart . . . . . 15, 32–35	\define@key . . . . . 31,
\@pdfnumpages . . . . .	\addresses . . . . . 456	42, 53, 55, 57, 59,
139, 295, 1450, 1473	\affiliation . . . . . 452	61, 63, 70, 75,
\@pdfpagerange . . . . .	\and . . . . . 222	79, 98, 116, 118,
155, 296, 1443, 1474	ASCII . . . . . 19, 46	120, 122, 132,
\@pdfproducer . 1150,	\AtBeginDocument . 1172	134, 136, 138,
1164, 1171, 1173	\AtEndDocument . . . . . 7	140, 142, 144,
\@pdfpublication 133,	\AtEndDvi . . . . . 5	146, 148, 150,
297, 484, 1438, 1475	atendddvi . . . . . 18, 92	152, 154, 156,
\@pdfpublisher . . . . .	\AtEndPreamble . . . . 508	158, 160, 162,
149, 408, 481, 1284	Author . . . . . 11, 25, 92	164, 166, 174,
\@pdfpubtype . . . 135,		176, 178, 180,
298, 494, 1439, 1476		182, 184, 186,
\@pdfrendition 126, 1330	<b>B</b>	210, 222, 243, 1002
\@pdfsource . . . . .	Bag . . . . . 89	\do . . . 1004, 1011, 1221
... 115, 1289, 1291	baseurl (option) . . . . .	DOI . . . . . 2, 6, 24, 33, 34
\@pdfsubject . 299, 1282	4, 6, 14, 18, 24, 64	draft (option) . . . . . 8
\@pdfsubtitle . . 163,	BOM . . . . . 76, 88	dvipdf (option) . . . . . 80
300, 405, 1437, 1477		dvipdfm . . . . . 80
\@pdftitle . . . . .	<b>C</b>	dvips (option) . . . . . 80
. 301, 334, 387,	CiAdrCity . . . . . 2	dvips . . . . . 10, 47, 88
1099, 1107, 1281	CiAdrCtry . . . . . 2	dvipsone (option) . . . . 80
\@pdftrapped . . . . 1195	CiAdrExtadr . . . . . 2	dviwindo (option) . . . 80
\@pdftype . . . . 54, 1286	CiAdrPcode . . . . . 2	
\@pdfuapart . . . . 78,	CiAdrRegion . . . . . 2	<b>E</b>
302, 331, 1365, 1761	CiEmailWork . . . . . 2	$\epsilon$ -TeX . . . . . 58
\@pdfurl . . . . .	CiTelWork . . . . . 3	\EdefEscapeHex 769, 782
159, 303, 1448, 1478	\city . . . . . 428	\EdefUnescapeHex . . 786
\@pdfversionid 121, 1329	CiUrlWork . . . . . 3	\EdefUnescapeString 756
\@pdfvolumenum . 151,	\country . . . . . 440	\email . . . . . 416
304, 497, 1441, 1479	CreationDate . . . . . 36, 90	\empty . . . . . 1597
\@pdfxstandard . . . . .	<b>D</b>	\equal . . . . . 92
... 96, 305, 330,	Date . . . . . 43	etoolbox . . . . . 18
1373, 1376, 1378	\day . . . . . 689, 690, 692	ETX . . . . . 38, 45
\@pdfxversion . . . . 1765	dc:creator . . . 2, 12, 62, 90	
\@publishers . . . . . 409	dc:date . . . . . 2, 62	<b>F</b>
\@subtitle . . . . . 406	dc:description . . . . .	\filemoddate . . . . . 724
\@tempswafalse 1200, 1235	. . . 3, 12, 52, 62, 90	
\@tempswatrue . 1200,	dc:format . . . . . 2	<b>G</b>
1202, 1235, 1237	dc:identifier . . . . . 3, 62	Ghostscript . . . . . 10, 11
\@title . . . . . 389	dc:language . . . . . 3,	gitver . . . . . 6
\^ 560, 564, 747, 1127,	15, 35, 62, 88, 89	
1128, 1786–1788	dc:publisher . . . . . 3	<b>H</b>
\_ . . . . . 1126, 1128	dc:rights . . . 2, 16, 52, 62	\hbox . . . . . 456
\~ . . . . . 569, 880, 881	dc:source . . . . . 2, 62, 88	\Hy@driver 1844, 1848,
	dc:subject . . . . . 2, 62	1852, 1856, 1861
	dc:title . . 3, 12, 52, 62, 90	\Hy@unicodefalse 33, 44
\_ . . . . . 846, 881, 1126	dc:type . . . . . 3, 62	hyperref . . . 1, 4–9, 12,
		15, 18–20, 25, 26,

28, 29, 31, 35–37,	\hyxmp@address@val .	\hyxmp@check@std ..
58, 59, 78–80, 89–91	..... <a href="#">416</a> , <a href="#">422</a> ,	..... <a href="#">91</a> , 103–111
\hypersetup ... <a href="#">260</a> , 371	<a href="#">428</a> , <a href="#">434</a> , <a href="#">440</a> , <a href="#">446</a>	\hyxmp@comma .....
hyperxmp 1, 2, 4–9, 12–	\hyxmp@alt@description	. 167, 223, 244, <a href="#">559</a>
21, 24, 26, 30, 35–	..... <a href="#">998</a> , 1008	\hyxmp@commas@to@list
39, 45, 54, 58–60,	\hyxmp@alt@rights ..	<a href="#">543</a> , 579, 1246, 1403
68, 79, 82, 89, 90, 92	..... <a href="#">998</a> , 1009	\hyxmp@commas@to@list@i
\hyxmp@is@unicode . <a href="#">790</a>	\hyxmp@alt@title ..	..... <a href="#">545</a> , <a href="#">547</a>
\hyxmp@acm@isbn ... <a href="#">467</a>	..... <a href="#">998</a> , 1007	\hyxmp@concat@metadata
\hyxmp@acm@publisher	\hyxmp@and ..... <a href="#">222</a>	..... <a href="#">265</a>
..... <a href="#">480</a>	\hyxmp@append@hex ..	\hyxmp@cond@dc@identifier
\hyxmp@acm@pubtype . <a href="#">489</a>	..... <a href="#">1047</a> ,	.. <a href="#">1272</a> , 1297–1300
\hyxmp@add@simple ..	1066–1068, 1072	\hyxmp@construct@packet
.... <a href="#">945</a> , 1195,	\hyxmp@append@hex@iii	..... <a href="#">1792</a> , 1842
1280, 1291, 1302,	..... <a href="#">1065</a> ,	\hyxmp@count@non@spaces
1319, 1321, 1327–	1071, 1081, 1092	..... <a href="#">1942</a> , <a href="#">1953</a>
1330, 1334, 1336,	\hyxmp@append@hex@iv	\hyxmp@count@spaces
1340, 1342, 1346,	..... <a href="#">1070</a> ,	..... <a href="#">1941</a> , <a href="#">1944</a>
1348, 1350, 1351,	1076, 1077,	\hyxmp@crap@convert
1355, 1356, 1360,	1079, 1094–1096	..... <a href="#">872</a> , <a href="#">906</a>
1361, 1365, 1372,	\hyxmp@as@pdf@date . <a href="#">616</a>	\hyxmp@crap@result .
1373, 1376, 1378,	\hyxmp@as@xmp@date .	..... <a href="#">862</a> , 898
1418–1421, 1435,	..... <a href="#">36</a> , <a href="#">47</a> ,	\hyxmp@crap@test <a href="#">869</a> , <a href="#">894</a>
1439, 1441–1450	<a href="#">588</a> , 726, 1337, 1343	\hyxmp@create@uuid .
\hyxmp@add@simple@lang	\hyxmp@assign@major@minor	.. <a href="#">1074</a> , 1102, 1111
..... <a href="#">963</a> ,	..... <a href="#">1177</a> , 1196	\hyxmp@cur@lang ...
1437, 1438, 1440	\hyxmp@at@end <a href="#">3</a> , 529, 542	..... <a href="#">1004</a> , <a href="#">1012</a>
\hyxmp@add@simple@lang@i	\hyxmp@auto@assign@data	\hyxmp@dc@lang ....
..... <a href="#">966</a> , <a href="#">969</a>	.... <a href="#">378</a> , 520, 542	... <a href="#">505</a> , 516, 1294
\hyxmp@add@simple@lang@ii	\hyxmp@banner@to@producer	\hyxmp@dc@schema ..
..... <a href="#">970</a> , <a href="#">972</a>	.. <a href="#">1153</a> , 1156, <a href="#">1164</a>	..... <a href="#">1279</a> , 1824
\hyxmp@add@simple@pfx	\hyxmp@begin@ext@decl	\hyxmp@declare@extensions
..... <a href="#">986</a> , 1275	..... <a href="#">1494</a> ,	..... <a href="#">1754</a> , 1821
\hyxmp@add@simple@var	1533, 1544, 1569,	\hyxmp@declare@field
..... <a href="#">954</a> ,	1585, 1599,	..... <a href="#">1523</a> ,
1193, 1194, 1197	1615, 1627, 1685	1651, 1654, 1657,
\hyxmp@add@to@xml ..	\hyxmp@begin@extension@decls	1660, 1663,
. 948, 950, 958,	..... <a href="#">1482</a> , 1755	1666, 1669, 1672
976, 980, 988,	\hyxmp@big@prime ..	\hyxmp@declare@property
992, 994, <a href="#">1113</a> ,	..... <a href="#">1021</a> ,	..... <a href="#">1511</a> ,
1139, 1207, 1214,	1024, 1034, 1044	1537, 1548, 1553,
1218, 1223, 1228,	\hyxmp@big@prime@ii	1558, 1562, 1573,
1240, 1248, 1254,	..... <a href="#">1021</a> , 1043	1578, 1589, 1603,
1263, 1391, 1395,	\hyxmp@bom .. <a href="#">1779</a> , 1794	1607, 1619, 1631,
1399, 1405, 1412,	\hyxmp@check@iptc@data	1689, 1693, 1697,
1427, 1483, 1489,	..... <a href="#">1452</a> , 1819	1701, 1705, 1709,
1495, 1505, 1512,	\hyxmp@check@prism@data	1713, 1717, 1721,
1516, 1524, 1636,	..... <a href="#">1464</a> , 1820	1725, 1729, 1733,
1675, 1794, 1833	\hyxmp@check@standards	1738, 1743, 1748
	..... <a href="#">315</a> , 527	



\hyxmp@def@DocumentID	\hyxmp@iptc@data ..	\hyxmp@pdf@schema ..
..... 1098, 1325	.. 1410, 1453, 1769	..... 1192, 1822
\hyxmp@def@InstanceID	\hyxmp@iptc@extensions	\hyxmp@pdf@to@xmp@date
..... 1104, 1326	..... 1626, 1771	.. 590, 595, 718, 721
\hyxmp@define@pdfproducer	\hyxmp@iptc@schema ..	\hyxmp@pdfa@id@extensions
..... 1150, 1174	..... 1409, 1831	..... 1568, 1759
\hyxmp@DocumentID ..	\hyxmp@is@unicode ..	\hyxmp@pdfa@id@schema
..... 117,	.... 758, 775, 790	..... 1358, 1827
1098, 1325, 1327	\hyxmp@koma@phones ..	\hyxmp@pdfauthor ..
\hyxmp@dq@code .. 1, 1974	..... 343, 400	... 213, 222, 1287
\hyxmp@driver .... 1841	\hyxmp@LA@accept ..	\hyxmp@pdfkeywords ..
\hyxmp@embed@packet	.. 1001, 1007–1009	... 213, 243, 1288
..... 531, 1841	\hyxmp@legal ..... 1306	\hyxmp@pdfstringdef
\hyxmp@embed@packet@dvi	\hyxmp@list ... 1246,	..... 24, 35,
..... 1853, 1923	1252, 1403, 1404	46, 53, 55, 57, 59,
\hyxmp@embed@packet@luatex	\hyxmp@list@to@lines	61, 63, 72, 76, 81,
..... 1849, 1886	..... 1388,	99, 116, 118, 120,
\hyxmp@embed@packet@pdfmark	1417, 1423–1425	122, 132, 134,
..... 1865, 1893	\hyxmp@list@to@xml ..	136, 138, 140,
\hyxmp@embed@packet@pdftex	..... 1234,	142, 144, 146,
..... 1845, 1873	1287, 1288, 1294	148, 150, 152,
\hyxmp@embed@packet@xetex	\hyxmp@major@minor 1177	154, 156, 158,
..... 1857, 1961	\hyxmp@mm@extensions	160, 162, 164,
\hyxmp@end@ext@decl	..... 1543, 1757	169, 174, 176,
... 1504, 1541,	\hyxmp@mm@schema ..	178, 180, 182,
1566, 1582, 1594,	..... 1324, 1830	184, 186, 990, 1003
1611, 1623, 1752	\hyxmp@modulo@a ...	\hyxmp@pdfua@id@extensions
\hyxmp@end@extension@decls	... 1015, 1034,	..... 1584, 1763
..... 1488, 1777	1044, 1050, 1085	\hyxmp@pdfua@id@schema
\hyxmp@extra@indent	\hyxmp@new@xml 1129, 1130	..... 1364, 1828
..... 944, 948,	\hyxmp@no@bad@parts	\hyxmp@pdfx@id@extensions
959, 988, 1392, 1416	..... 64, 71, 80	..... 1596, 1767
\hyxmp@find@metadata	\hyxmp@no@info@lists	\hyxmp@pdfx@id@schema
..... 265, 530	.... 187, 211, 528	..... 1367, 1829
\hyxmp@first@char .. 586	\hyxmp@num ..... 906	\hyxmp@pdfx@major ..
\hyxmp@first@char@i	\hyxmp@one@token ..	... 88, 97, 113,
.... 586, 589, 617	..... 1023,	1368, 1597, 1613
\hyxmp@gobbletwo 656, 669	1027, 1945,	\hyxmp@photoshop@data
\hyxmp@hash .....	1946, 1954, 1955	..... 1353
... 1133, 1798,	\hyxmp@padding 1137, 1837	\hyxmp@photoshop@schema
1806, 1814–1817	\hyxmp@parse@acmart	..... 1353, 1825
\hyxmp@Hyp@pdfauthor	.... 411, 413, 504	\hyxmp@prism@data ..
..... 216	\hyxmp@parse@time ..	.. 1433, 1465, 1773
\hyxmp@Hyp@pdfkeywords	..... 597, 599	\hyxmp@prism@extensions
..... 237	\hyxmp@parse@tz ...	..... 1684, 1775
\hyxmp@hypersetup .. 260	.... 606, 609, 613	\hyxmp@prism@schema
\hyxmp@InstanceID ..	\hyxmp@parse@tz@char	..... 1432, 1832
..... 119,	..... 601, 603	\hyxmp@ProcessKeyvalOptions
1104, 1326, 1328	\hyxmp@pdf@extensions	..... 255
\hyxmp@iprefix 990, 991	..... 1532, 1756	



<code>\hyxmp@rand@num</code> . . .	<code>\hyxmp@textunderscore</code>	1224, 1246, 1266,
. . . <u>1040</u> , 1049,	. . . . . <u>24</u>	1273, 1296, 1403
1084, 1101, 1110	<code>\hyxmp@timestamp</code> .. <u>723</u>	<code>\hyxmp@xmllify</code> . . . . .
<code>\hyxmp@rdf@dc</code> . . . . .	<code>\hyxmp@today@pdf</code> 523, <u>733</u>	. . . . . 386, <u>754</u> ,
. . . <u>1199</u> , 1281–1283	<code>\hyxmp@today@xmp</code> . .	947, 957, 965,
<code>\hyxmp@redefine@Hyp</code>	. . . . . 711, <u>716</u> ,	974, 991, 993,
. . . . . <u>215</u> , 257, 262	734, 1107, 1285,	1205, 1213, 1222,
<code>\hyxmp@remove@this</code> .	1334, 1340, 1346	1245, 1262, 1402
. . . . . 1168, <u>1171</u>	<code>\hyxmp@today@xmp@define</code>	<code>\hyxmp@xmp@basic@schema</code>
<code>\hyxmp@rights</code> . 1306,	. . . <u>682</u> , 731, 1105	. . . . . 1332, 1826
1309, 1313, 1315	<code>\hyxmp@toxml</code> . . 784, <u>807</u>	<code>\hyxmp@xmp@to@pdf@date</code>
<code>\hyxmp@seed@rng</code> . . .	<code>\hyxmp@toxml@unicodetex</code>	. . . . . 620, <u>623</u> , 734
. . . <u>1023</u> , 1100, 1109	. . . . . 772, 832	<code>\hyxmp@xmp@to@pdf@date@i</code>
<code>\hyxmp@seed@rng@i</code> . .	<code>\hyxmp@trimb</code> . . 740, <u>743</u>	. . . . . 624, <u>626</u>
. . . . . 1025, <u>1027</u>	<code>\hyxmp@trimc</code> . . 743, <u>744</u>	<code>\hyxmp@xmp@to@pdf@date@ii</code>
<code>\hyxmp@seed@string</code> .	<code>\hyxmp@trimspaces</code> . .	. . . . . 629, 632
. . . . . <u>1098</u> , <u>1104</u>	. . . . . 552, <u>736</u>	<code>\hyxmp@xmp@to@pdf@date@iii</code>
<code>\hyxmp@set@dc@lang</code> .	<code>\hyxmp@try</code> . . . . . <u>862</u>	. . . . . 635, <u>638</u>
. . . <u>505</u> , 515, 1293	<code>\hyxmp@try@today</code> <u>710</u> ,	<code>\hyxmp@xmp@to@pdf@date@iv</code>
<code>\hyxmp@set@koma@phones</code>	717, 720, 723, 730	. . . . . 641, <u>644</u>
. . . . . <u>343</u> , 398	<code>\hyxmp@unicodetexfalse</code>	<code>\hyxmp@xmp@to@pdf@date@v</code>
<code>\hyxmp@set@pdfx@major</code>	. . . . . <u>746</u>	. . . . . 647, <u>650</u>
. . . . . <u>83</u> , 113	<code>\hyxmp@unicodetextrue</code>	<code>\hyxmp@xmp@to@pdf@date@vi</code>
<code>\hyxmp@set@pdfx@major@i</code>	. . . . . <u>746</u>	. . . . . 653, <u>657</u>
. . . . . <u>83</u> , <u>84</u>	<code>\hyxmp@uscore</code> . . 26, <u>563</u>	<code>\hyxmp@xmp@to@pdf@date@vii</code>
<code>\hyxmp@set@pdfx@major@ii</code>	<code>\hyxmp@use@first@valid</code>	. . . . . 660, 663, <u>673</u>
. . . . . <u>85</u> , <u>88</u>	. . . . . <u>357</u> , 387,	<code>\hyxmp@xmp@to@pdf@date@viii</code>
<code>\hyxmp@set@rand@num</code>	391, 395, 399,	. . . . . 676, <u>679</u>
. . . <u>1040</u> , 1048, 1083	402, 405, 408,	<code>\hyxmp@xmpRights@schema</code>
<code>\hyxmp@singleton@dc</code>	418, 424, 430,	. . . . . <u>1305</u> , 1823
. . . <u>1260</u> , 1284–1286	436, 442, 448,	<code>\hyxmp@zero</code> . . . . 915,
<code>\hyxmp@skiptorelax</code> .	461, 474, 481,	922, 929, 935, <u>940</u>
. . . . . 899, <u>905</u>	484, 494, 497, 500	
<code>\hyxmp@skipzeros</code> . . <u>857</u>	<code>\hyxmp@use@first@valid@i</code>	
<code>\hyxmp@SpaceOther</code> . .	. . . . . 359, <u>363</u>	
. . . . . 866, <u>879</u>	<code>\hyxmp@value</code> <u>1003</u> , <u>1206</u>	
<code>\hyxmp@standards</code> . . <u>328</u>	<code>\hyxmp@x@default</code> 381,	
<code>\hyxmp@string@len</code> . .	<u>1149</u> , 1211, 1219	
. . . . . 1924, <u>1939</u>	<code>\hyxmp@xetex@crap</code> . .	
<code>\hyxmp@strip@isbn@date</code>	. . . . . 763, <u>862</u>	
. . . . . <u>467</u>	<code>\hyxmp@xml</code> . 949, 951,	
<code>\hyxmp@sublist</code> . . . .	989, 995, 1130,	
. . . 548, 549, 552, 553	<u>1137</u> , 1515, <u>1792</u> ,	
<code>\hyxmp@suppress@pdf@info</code>	1882, 1890, 1913,	
. . . . . <u>188</u>	1924, 1931, 1962	
<code>\hyxmp@temp@list</code> . . <u>572</u>	<code>\hyxmp@xmllified</code> <u>754</u> ,	
<code>\hyxmp@temp@str</code> . . . <u>572</u>	950, 959, 966,	
<code>\hyxmp@text</code> . . . . .	977, 981, 992,	
. . . <u>754</u> , <u>832</u> , <u>862</u> , <u>906</u>	994, 1206, 1215,	

# I

IETF . . . . .	5
<code>\if@ACM@journal</code> . . .	489
<code>\if@tempswa</code> . . . . .	1204, 1239
<code>ifdraft</code> . . . . .	22
<code>\ifdraft</code> . . . . .	123, 126
<code>\iffalse</code> . . . . .	1199, 1234
<code>\ifHy@pdfa</code> . . . . .	
. . . . .	316, 1281, 1282,
. . . . .	1287, 1359, 1758
<code>\ifhyxmp@unicodetex</code>	<u>746</u> , 757, 1116, 1780
<code>\ifLuaTeX</code> . . . . .	1152
<code>ifluatex</code> . . . . .	79
<code>\ifluatex</code> . . . . .	1875, 1879
<code>ifmtarg</code> . . . . .	18, 91
<code>\ifPDFTeX</code> . . . . .	1155

<code>\IfSubStr</code> . . . . .	<code>nativepdf</code> (option) . . . . . 80	<code>pdfdoi</code> . . . . . 4, 6
. 459, 460, 468, 469	<code>\newif</code> . . . . . 746	<code>pdffeissn</code> . . . . . 4, 6
<code>iftex</code> . . . . . 18, 92	<code>\next</code> . 34, 45, 93, 100,	<code>pdfidentifier</code> 4, 6, 62, 92
<code>ifthen</code> . . . . . 18	188, 364, 366,	<code>pdfinstanceid</code> . 4, 6, 91
<code>\ifthenelse</code> . . . . . 92	372, 376, 547,	<code>pdfisbn</code> . . . . . 4, 6
<code>\ifXeTeX</code> . . . . . 762, 1158	725, 728, 1027, 1765	<code>pdfissn</code> . . . . . 4, 6
<code>Info</code> . 8, 11–13, 25, 36, 58	<code>ngerman</code> . . . . . 17, 87	<code>pdfissuenum</code> . . . . . 4, 7
<code>intcalc</code> . . . . . 18	<code>\number</code> 909, 911, 913,	<code>pdfkeywords</code> . . . . . 4,
<code>\intcalcDiv</code> 911, 918, 925	918, 920, 925, 927	12, 14, 18, 26, 62
<code>\intcalcMod</code> 913, 920, 927	<code>\numexpr</code> . . . . . 1891	<code>pdflang</code> . . 4–7, 15,
<code>IPTC</code> . . . . . 13, 24,		18, 31, 35, 52, 62, 90
56, 67, 68, 73, 76, 89		<code>pdflicenseurl</code> . . . . .
<code>lptc4xmpCore:ContactInfo</code>	<b>O</b>	. . . . . 5, 14, 63, 88
. . . . . 67, 73	options	<code>pdfmark</code> . . . . . 80
<code>lptc4xmpCore:CreatorContactInfo</code>	baseurl . . . . .	<code>pdfmetadate</code> . . . . .
. . . . . 2, 3, 67, 89	4, 6, 14, 18, 24, 64	. . . . . 5, 7, 8, 19, 90
<code>ISBN</code> . . . . . 2, 6, 23, 34	draft . . . . . 8	<code>pdfmetalang</code> . . . . .
<code>ISO</code> . . . . . 5–7, 15, 20, 52	dvipdf . . . . . 80	5, 6, 15, 52, 87, 90
<code>ISSN</code> . . . . . 2, 6, 23	dvips . . . . . 80	<code>pdfmoddate</code> 4, 7, 8, 90
	dvipsone . . . . . 80	<code>pdfnumpages</code> . . . . .
<b>J</b>	dviwindo . . . . . 80	. . . . . 5, 7, 16, 17
<code>\jobname</code> . . . . . 115,	keeppdfinfo . 8, 13, 25	<code>pdfpagerange</code> . . . . .
309, 535, 724,	nativepdf . . . . . 80	. . . . . 5, 7, 16, 17
1099, 1107, 1862	pdfa . . . 8, 20, 29, 90	<code>pdfproducer</code> . 4, 18, 59
	pdfaconformance .	<code>pdfpublication</code> . . 5, 7
<b>K</b>	. . . . . 4, 8, 65	<code>pdfpublisher</code> . . . . . 5, 7
<code>keeppdfinfo</code> (option) .	pdfapart . 4, 8, 20, 65	<code>pdfpubtype</code> . . . . . 5, 7
. . . . . 8, 13, 25	pdfauthor . . . . .	<code>pdfrendition</code> . . 5, 8, 91
<code>Keywords</code> . 11, 25, 60, 92	. . . 4, 5, 12, 14,	<code>pdfsource</code> . . . . . 5, 9, 90
<code>\KV@Hyp@pdfauthor</code> . . 222	15, 18, 26, 27, 62, 90	<code>pdfsubject</code> 4, 12, 18, 62
<code>\KV@Hyp@pdfkeywords</code> 243	pdfauthortitle . 4, 5, 14	<code>pdfsubtitle</code> . . . . . 5
<code>kvoptions</code> . . . . . 18, 28	pdfbookedition . . 4, 7	<code>pdftitle</code> . . . . . 4, 5,
	pdfbytes . . . . . 4, 8	12, 15, 18, 31, 62, 90
<b>L</b>	pdfcaptionwriter . 4, 5	<code>pdftrapped</code> 4, 8, 18, 92
<code>Lang</code> . . . . . 35	pdfcontactaddress .	<code>pdftype</code> . . 5, 8, 62, 90
<code>LF</code> . . . . . 66	. . . . . 4, 5, 13	<code>pdfuapart</code> . 5, 8, 20, 65
<code>LuaL<sup>A</sup>T<sub>E</sub>X</code> . . . . . 10,	pdfcontactcity . . 4, 5	<code>pdfurl</code> . . . . . 5, 6
13, 14, 43, 59, 90	pdfcontactcountry 4, 5	<code>pdfversionid</code> . . 5, 6, 64
<code>LuaT<sub>E</sub>X</code> . . . . . 45,	pdfcontactemail . 4, 5	<code>pdfvolumenum</code> . . 5, 7
48, 79, 82, 88, 90, 92	pdfcontactphone . 4, 5	<code>pdfxstandard</code> . . . . .
<code>luatex85</code> . . . . . 92	pdfcontactpostcode	. . . . . 5, 8, 9, 21, 22, 65
<code>\luatexbanner</code> . . . . 1153	. . . . . 4, 5	<code>ps2pdf</code> . . . . . 80
	pdfcontactregion . 4, 5	<code>textures</code> . . . . . 80
<b>M</b>	pdfcontacturl . 4, 5, 14	<code>unicode</code> . . . . . 15, 90
<code>\makeatletter</code> . . . . 1168	pdfcopyright . . . . .	<code>vtexpdfmark</code> . . . . . 80
<code>memoir</code> . . . . . 89	. . . 4, 5, 62, 63, 88	
<code>Metadata</code> . . . . . 11, 78, 81	pdfcreationdate . .	<b>P</b>
<code>\month</code> . . . . . 684, 685, 687	. . . . . 4, 7, 8, 36, 90	<code>\PackageWarning</code> . . .
	pdfdate . . . . . 4,	. . . . . 66, 101, 573
<b>N</b>	7, 15, 19, 56, 89, 92	
<code>NAK</code> . . . . . 19, 38, 45	pdfdocumentid 4, 6, 91	

<code>\PackageWarningNoLine</code>	<code>pdfcontactemail</code> (option)	<code>\pdfminorversion</code>
..... 190, 308,	..... 4, 5	1181
318, 335, 534, 1860	<code>pdfcontactphone</code> (option)	<code>pdfmoddate</code> (option)
<code>\patchcmd</code> .... 196, 202	..... 4, 5	..... 4, 7, 8, 90
PDF 1–4, 7, 8, 10–14, 16,	<code>pdfcontactpostcode</code> (option)	<code>pdfnumpages</code> (option)
19, 25, 29, 35–37,	..... 4, 5	..... 5, 7, 16, 17
39, 40, 44, 45, 47,	<code>pdfcontactregion</code> (option)	<code>\pdfobj</code> ..... 1879
56, 58, 59, 70, 75,	..... 4, 5	<code>pdfpagerange</code> (option)
77–79, 82, 89, 90, 92	<code>pdfcontacturl</code> (option)	..... 5, 7, 16, 17
PDF/A ..... 3,	..... 4, 5, 14	<code>pdfproducer</code> (option)
8, 12, 13, 20, 25,	<code>pdfcopyright</code> (option)	..... 4, 18, 59
29, 58, 59, 65, 68,	... 4, 5, 62, 63, 88	<code>pdfpublication</code> (option)
71, 73, 74, 76, 89, 90	<code>pdfcreationdate</code> (option)	..... 5, 7
PDF/UA ..... 3, 8,	.... 4, 7, 8, 36, 90	<code>pdfpublisher</code> (option)
21, 29, 65, 72, 76, 92	<code>\pdfcreationdate</code> .. 718	<code>pdfpubtype</code> (option)
PDF/X ... 3, 8, 9, 21,	<code>pdfdate</code> (option) .. 4,	<code>pdfrendition</code> (option)
22, 29, 65, 72, 76, 92	7, 15, 19, 56, 89, 92	..... 5, 8, 91
<code>pdf:Keywords</code> 2, 12, 59, 60	PDFDocEncoding ...	<code>pdfsource</code> (option)
<code>pdf:PDFVersion</code> . 3, 59, 92	..... 26, 45, 46	5, 9, 90
<code>pdf:Producer</code> ... 3, 58–60	<code>pdfdocumentid</code> (option)	<code>\pdfstringdef</code> ..... 27
<code>pdf:trapped</code> ..... 3	..... 4, 6, 91	<code>pdfsubject</code> (option)
<code>\PDF@FinishDoc</code> ....	<code>pdfdoi</code> (option) .... 4, 6	..... 4, 12, 18, 62
.... 189, 197, 203	<code>pdfdoi</code> (option) .... 4, 6	<code>pdfsubtitle</code> (option) ... 5
<code>pdfa</code> (option) 8, 20, 29, 90	<code>pdfescape</code> ..... 18	<code>pdfTeX</code> 8, 47, 79, 82, 90, 92
<code>pdfa:conformance</code> (option)	<code>\pdfextension</code> 1887, 1891	<code>\pdftexbanner</code> .... 1156
... 4, 8, 65	<code>\pdffeedback</code> . 721, 1891	<code>pdftitle</code> (option) 4, 5,
<code>pdfaid:conformance</code> ... 3	<code>pdfidentifier</code> (option)	12, 15, 18, 31, 62, 90
<code>pdfaid:part</code> ..... 3	..... 4, 6, 62, 92	<code>pdftrapped</code> (option) ..
<code>pdfapart</code> (option) ....	<code>pdfinstanceid</code> (option)	..... 4, 8, 18, 92
..... 4, 8, 20, 65	..... 4, 6, 91	<code>pdftype</code> (option) 5, 8, 62, 90
<code>pdfaType:prefix</code> ..... 89	<code>pdfisbn</code> (option) ... 4, 6	<code>pdfuaid:part</code> ..... 3
<code>pdfauthor</code> (option) ...	<code>pdfissn</code> (option) .... 4, 6	<code>pdfuapart</code> (option) ...
... 4, 5, 12, 14,	<code>pdfissuenum</code> (option) 4, 7	..... 5, 8, 20, 65
15, 18, 26, 27, 62, 90	<code>pdfkeywords</code> (option) 4,	<code>pdfurl</code> (option) .... 5, 6
<code>pdfauthor:html</code> (option)	12, 14, 18, 26, 62	<code>\pdfvariable</code> ..... 1189
..... 4, 5, 14	<code>pdflang</code> (option) 4–7, 15,	<code>pdfversionid</code> (option)
<code>pdfbookedition</code> (option)	18, 31, 35, 52, 62, 90	..... 5, 6, 64
..... 4, 7	<code>\pdflastobj</code> ..... 1883	<code>pdfvolumenum</code> (option)
<code>pdfbytes</code> (option) ... 4, 8	<code>pdfL<sup>A</sup>T<sub>E</sub>X</code> 4, 10, 13, 43, 59	..... 5, 7
<code>pdfcaptionwriter</code> (option)	<code>pdflicenseurl</code> (option)	<code>pdfxid:GTS_PDFXVersion</code>
..... 4, 5	..... 5, 14, 63, 88	..... 3
<code>\pdfcatalog</code> ..... 1883	<code>\pdfmajorversion</code> . 1185	<code>pdfxstandard</code> (option)
<code>\pdfcompresslevel</code> . 1877	<code>pdfmark</code> (option) .... 80	. 5, 8, 9, 21, 22, 65
<code>pdfcontactaddress</code> (option)	<code>\pdfmark</code> ..... 1894,	<code>photoshop:AuthorsPosition</code>
... 4, 5, 13	1897, 1901,	..... 3, 65
<code>pdfcontactcity</code> (option)	1911, 1915, 1919	<code>photoshop:CaptionWriter</code>
..... 4, 5	<code>pdfmetadate</code> (option)	..... 3, 65
<code>pdfcontactcountry</code> (option)	.... 5, 7, 8, 19, 90	PI ..... 56
..... 4, 5	<code>pdfmetalang</code> (option)	<code>polyglossia</code> 15, 35, 63, 93
	5, 6, 15, 52, 87, 90	<code>\postcode</code> ..... 446
		PRISM ... 7, 68, 69, 74, 76
		<code>prism:aggregationType</code> . 3

prism:bookEdition . . . . .	2	prism:aggregationType . . . . .	3	<b>S</b>	
prism:byteCount . . . . .	2	prism:bookEdition . . . . .	2	\scantokens . . . . .	1165, 1168
prism:doi . . . . .	2	prism:byteCount . . . . .	2	\scr@fromemail@var . . . . .	396
prism:elssn . . . . .	2	prism:doi . . . . .	2	\scr@frommobilephone@var . . . . .	346, 348
prism:isbn . . . . .	2	prism:elssn . . . . .	2	\scr@fromname@var . . . . .	392
prism:issn . . . . .	2	prism:isbn . . . . .	2	\scr@fromphone@var . . . . .	346, 352
prism:number . . . . .	2	prism:issn . . . . .	2	\scr@fromurl@var . . . . .	403
prism:pageCount . . . . .	3	prism:number . . . . .	2	\scr@subject@var . . . . .	388
prism:pageRange . . . . .	3	prism:pageCount . . . . .	3	scrLtr2 . . . . .	15
prism:publicationName . . . . .	3	prism:pageRange . . . . .	3	\SE->pdfdoc@03 . . . . .	752
prism:subtitle . . . . .	3	prism:publicationName . . . . .	3	\SE->pdfdoc@15 . . . . .	753
prism:url . . . . .	3	prism:subtitle . . . . .	3	\setbox . . . . .	456
prism:volume . . . . .	3	prism:url . . . . .	3	\setkeys . . . . .	1013
\ProcessKeyvalOptions . . . . .	255	prism:volume . . . . .	3	\special . . . . .	1925,
Producer . . . . .	60	xmp:BaseURL . . . . .	2	1933, 1962, 1968	
properties, XMP		xmp:CreateDate . . . . .	2, 36, 90	\state . . . . .	434
dc:creator . . . . .	2, 12, 62, 90	xmp:CreatorTool . . . . .	3	\streetaddress . . . . .	422
dc:date . . . . .	2, 62	xmp:MetadataDate . . . . .	2, 90	stringenc . . . . .	18
dc:description . . . . .	3, 12, 52, 62, 90	xmp:ModifyDate . . . . .	2, 90	\StringEncodingConvert . . . . .	759,
dc:format . . . . .	2	xmpMM:DocumentID . . . . .	3, 53, 64, 75	765, 776, 779, 874	
dc:identifier . . . . .	3, 62	xmpMM:InstanceID . . . . .	3, 53, 64, 75	Subject . . . . .	11
dc:language . . . . .	3, 62, 88, 89	xmpMM:RenditionClass . . . . .	3	<b>T</b>	
dc:publisher . . . . .	3	xmpMM:VersionID . . . . .	3, 64, 91	TeX . . . . .	16, 18, 42–
dc:rights . . . . .	2, 16, 52, 62	xmpRights:Marked . . . . .	2, 63, 88	45, 47, 48, 53, 54,	
dc:source . . . . .	2, 62, 88	xmpRights:WebStatement . . . . .	3, 63, 88	56, 58, 64, 81, 82, 92	
dc:subject . . . . .	2, 62	ps2pdf (option) . . . . .	80	texdate . . . . .	16
dc:title . . . . .	3, 12, 52, 62, 90			Text . . . . .	70
dc:type . . . . .	3, 62			\textunderscore . . . . .	25, 26, 28
lptc4xmpCore:ContactInfo . . . . .	67, 73			textures (option) . . . . .	80
lptc4xmpCore:CreatorContactInfo . . . . .	2, 3, 67, 89			\time . . . . .	694, 702
pdf:Keywords . . . . .	2, 12, 59, 60			Title . . . . .	11
pdf:PDFVersion . . . . .	3, 59			totpages . . . . .	16, 17
pdf:Producer . . . . .	3, 58–60			<b>U</b>	
pdf:trapped . . . . .	3			\undefined . . . . .	368
pdfaid:conformance . . . . .	3			Unicode . . . . .	15, 18, 45–
pdfaid:part . . . . .	3			49, 61, 66, 76, 82, 88	
pdfaType:prefix . . . . .	89			unicode (option) . . . . .	15, 90
pdfuaid:part . . . . .	3			URI . . . . .	6
pdfxid:GTS_PDFXVersion . . . . .	3			URL . . . . .	2, 3, 5, 6, 14,
photoshop:AuthorsPosition . . . . .	3, 65			20, 24, 25, 63–65, 67	
photoshop:CaptionWriter . . . . .	3, 65			UTF-16BE . . . . .	46
				UTF-32BE . . . . .	46
				UTF-8 . . . . .	46
				UUID . . . . .	3, 6, 22, 53, 55, 56, 89

<b>V</b>		
\vfuzz . . . . .	744	
vtexpdfmark (option) . .	80	
<b>X</b>		
\x . . . . .	862	
xdvipdfmx . . . . .	13, 14, 36, 81	
X <sub>Y</sub> L <sup>A</sup> T <sub>E</sub> X . . . . .	10, 13, 14, 36, 43, 59, 90, 92	
X <sub>Y</sub> L <sub>T</sub> E <sub>X</sub> . . . . .	45, 48, 81, 82, 87, 88, 92	
\XeTeXrevision . . . .	1159	
\XeTeXversion . . . .	1159	
XML 1, 2, 13, 37, 45–48, 50–52, 56, 57, 59– 62, 66, 67, 69, 77, 90		
XMP 1, 2, 4, 6–8, 11–19, 21, 25, 31–40, 43, 44, 47, 50–53, 56– 61, 63, 64, 68, 71, 75, 76, 79–82, 88–91 properties . . . . . <i>see</i> properties, XMP		
xmp:BaseURL . . . . .	2	
xmp:CreateDate . . . .	2, 36, 90	
xmp:CreatorTool . . . .	3	
xmp:MetadataDate . . .	2, 90	
xmp:ModifyDate . . . .	2, 90	
\xmpcomma . . . . .	167, 170, <u>222</u> , <u>243</u> , <u>558</u>	
xmpincl . . . . .	4	
\XMPLangAlt . . . . .	<u>1010</u>	
\xmplinesep . . . . .	. . <u>1383</u> , 1399, <u>1422</u>	
xmpMM:DocumentID . . . . .	3, 53, 64, 75	
xmpMM:InstanceID . . . . .	3, 53, 64, 75	
xmpMM:RenditionClass . .	3	
xmpMM:VersionID . . . .	3, 64, 91	
\xmpquote . . . . .	168, 171, <u>222</u> , <u>243</u> , <u>567</u>	
xmpRights:Marked . . . .	2, 63, 88	
xmpRights:WebStatement . . . . .	3, 63, 88	
\xmptilde . . . . .	<u>568</u>	
\XMPTruncateList . . . .	<u>572</u>	
\xpg@bcp@loaded . . . .	516	
<b>Y</b>		
\year . . . . .	683	