

The hyperxmp package*

Scott Pakin
scott+hyxmp@pakin.org

June 19, 2020

Abstract

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

1 Introduction

Adobe Systems, Inc. has been promoting XMP [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 \LaTeX document with three authors—Jack Napier, Edward Nigma, and Harvey Dent—named in the \LaTeX source in the usual way: “`\author{Jack Napier \and Edward Nigma \and Harvey Dent}`”. With `hyperxmp`, the generated PDF file will contain, among other information, the following stanza of XMP code embedded within it:

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

*This document corresponds to `hyperxmp` v5.4, dated 2020/06/19.

```

    <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^AT_EX 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 5.4 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. 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. ¹ 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.

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

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 \LaTeX <code>\date</code> command, and, like <code>\date</code>, defaults to the date the document was built. It must be specified in either XMP format [4] or PDF format [3]. XMP dates are written in the form <code>YYYY-MM-DDThh:mm:ss+TT:tt</code>.² 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>

²Although 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 ^A T _E 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 _E 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 L^AT_EX source file. It defaults to `\jobname.tex` but can be replaced by any other string. If `pdfsource` is given an empty argument, no document source will be specified at all.

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 L^AT_EX document that provides values for most of the metadata options that `hyperxmp` recognizes:

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

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

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

```

pdfcontactcountry={Switzerland},
pdfcontactphone={031 312 00 91},
pdfcontactemail={aeinstein@ipi.ch},
pdfcontacturl={%
    http://einstein.biz/,
    https://www.facebook.com/AlbertEinstein
},
pdfdocumentid={uuid:6d1ac9ec-4ff2-515a-954b-648eeb4853b0},
pdfversionid={2.998e8},
pdfpublication={[de]Annalen der Physik},
pdfpublisher={Wiley-VCH},
pdfpubtype={journal},
pdfvolumenum={322},
pdfissuenum={6},
pdfpagerange={132-148},
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.

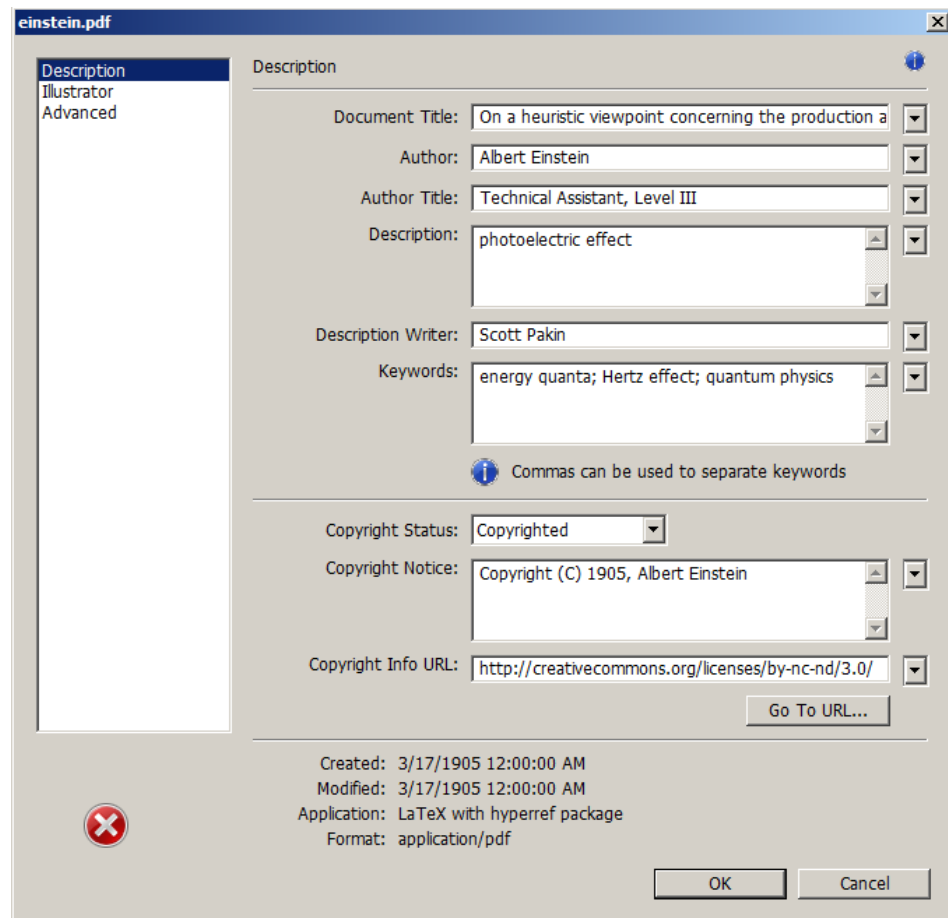


Figure 1: XMP metadata as it appears in Adobe Acrobat

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

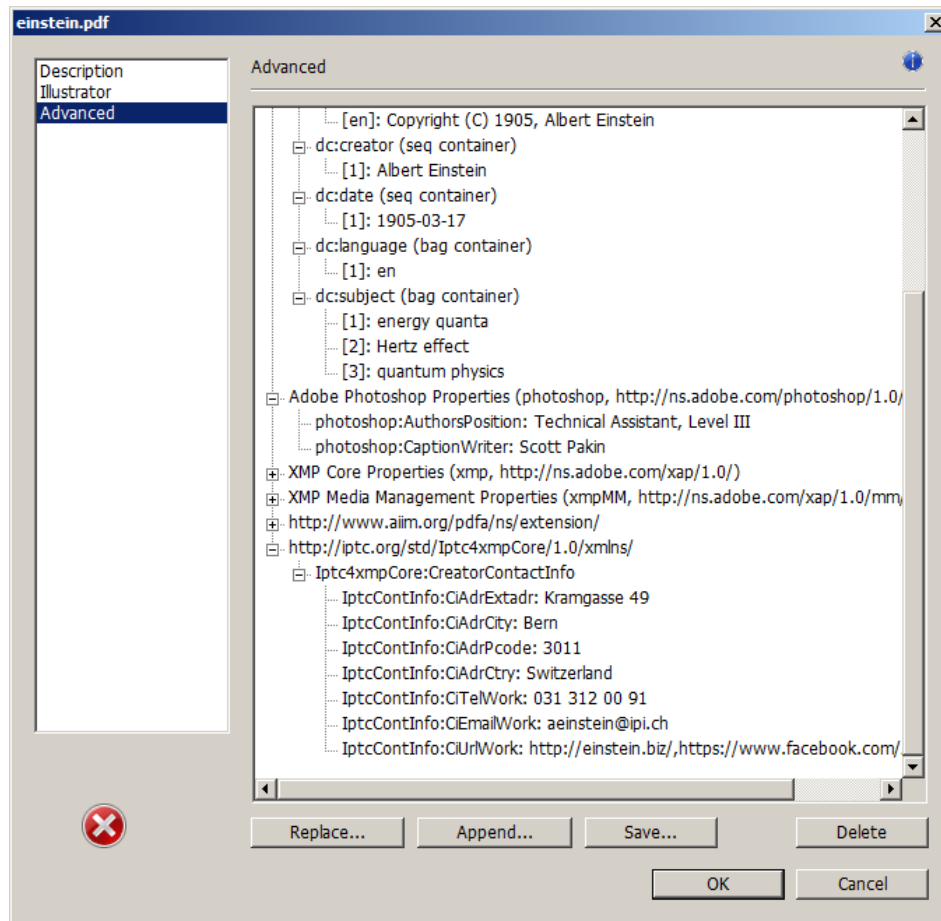


Figure 2: Additional XMP metadata as it appears in Adobe Acrobat

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 \LaTeX earlier than 0.85, `hyperxmp` leaves all PDF objects uncompressed. This is due to Lua \LaTeX treating object compression as a global parameter, unlike pdf \LaTeX , which treats it as a local parameter. Hence, when `hyperxmp` requests that the XMP packet be left uncompressed, Lua \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 \LaTeX v0.85 onwards.
2. Xe \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 \LaTeX), (2) pass the `--output-driver="xdvipdfmx -z0"` option to Xe \LaTeX 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 either `babel` or `polyglossia` package, it is recommended that it *not* explicitly set `pdflang`. `pdflang` accepts only a single language name while `hyperxmp` can automatically query `babel` and `polyglossia` for a list of all languages used in the document and include this list in an XMP `dc:language` element.

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 \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 \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^AT_EX to do that on your behalf:

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

3 Implementation

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

3.1 Initial preparation

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

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

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

```

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

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

```

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

```

`\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 }

```

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

```

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

```

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

```

45 \edef\next{%
46 \noexpand\hyxmp@pdfstringdef\noexpand\@pdfmetadatetime{%
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 68 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,
`\xmpcomma` `\xmpcomma` is defined as a placeholder character (`\hyxmp@comma`) and `\xmpquote`
`\xmpquote` as the identity function. The result is stored in `\hyxmp@pdfauthor` for use in
`\hyxmp@and` structured lists (those surrounding each entry with `<rdf:li>`). The second time,
`\and` `\xmpcomma` is defined as an ordinary comma, and `\xmpquote` is defined as a macro
`\hyxmp@pdfauthor` that puts its argument within double quotes. The result is stored in `\@pdfauthor`
`\@pdfauthor` for use in unstructured lists (those in which the entire list appears within a single
pair of tags). In case `pdfauthor` is left unspecified and we copy `\author`'s argument
to `pdfauthor`, we temporarily redefine `\and` as the list separator when producing a
structured list and as “and” when producing an unstructured list.

```

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

```

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`
`\ProcessKeyvalOptions` 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
`\hypersetup` performing its normal option processing.

```

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

```

`\hyxmp@concat@metadata` Assume that if the document loaded either `babel` or `polyglossia` it will eventually
define one or more languages that `hyperxmp` can list within a `dc:language` element.

```

265 \edef\hyxmp@concat@metadata{}
266 \AtEndPreamble{%
267 \ifpackageloaded{babel}{%
268 \edef\hyxmp@concat@metadata{babel}%
269 }{%
270 \ifpackageloaded{polyglossia}{%
271 \edef\hyxmp@concat@metadata{polyglossia}%
272 }{%
273 }%
274 }%
275 }

```

`\hyxmp@find@metadata` Issue a warning message if the author failed to specify any metadata at all. This
`\hyxmp@concat@metadata` excludes metadata that is included automatically such as the current timestamp.
Note that we don't consider `\@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.

```

276 \newcommand*{\hyxmp@find@metadata}{%
277 \edef\hyxmp@concat@metadata{%
278 \hyxmp@concat@metadata
279 \@baseurl
280 \@pdfauthor
281 \@pdfauthortitle

```

```

282 \pdfbookedition
283 \pdfbytes
284 \pdfcaptionwriter
285 \pdfcontactaddress
286 \pdfcontactcity
287 \pdfcontactcountry
288 \pdfcontactemail
289 \pdfcontactphone
290 \pdfcontactpostcode
291 \pdfcontactregion
292 \pdfcontacturl
293 \pdfcopyright
294 \pdfcreationdate
295 \pdfdatetime
296 \pdfdoi
297 \pdfeissn
298 \pdfidentifier
299 \pdfisbn
300 \pdfissn
301 \pdfissuenum
302 \pdfkeywords
303 \pdflang
304 \pdflicenseurl
305 \pdfmetadatetimestamp
306 \pdfmoddate
307 \pdfnumpages
308 \pdfpagerange
309 \pdfpublication
310 \pdfpubtype
311 \pdfsubject
312 \pdfsubtitle
313 \pdftitle
314 \pdfuapart
315 \pdfurl
316 \pdfvolumenum
317 \pdfxstandard
318 }%
319 \ifx\hyxmp@concat@metadata\@empty
320 \PackageWarningNoLine{hyperxmp}{%
321 \jobname.tex did not specify any metadata to\MessageBreak
322 include in the XMP packet.\space\space Please see the\MessageBreak
323 hyperxmp documentation for instructions on how to\MessageBreak
324 provide metadata values to hyperxmp}%
325 \fi
326 }

```

`\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.

```

327 \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.

```

328 \ifHy@pdfa
329   \@ifmtargexp{\@pdfapart}{%
330     \PackageWarningNoLine{hyperxmp}{%
331       'pdfa' was passed to hyperref, but 'pdfapart' was\MessageBreak
332       not specified.\space\space Setting pdfapart to '1' and\MessageBreak
333       pdfaconformance to 'B'%
334     }%
335     \gdef\@pdfapart{1}%
336     \gdef\@pdfaconformance{B}%
337   }%
338   {}%
339 \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.

```

340 \edef\hyxmp@standards{%
341   \@pdfapart
342   \@pdfxstandard
343   \@pdfuapart
344 }%

```

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

```

345 \@ifnotmtargexp{\hyxmp@standards}{%
346   \@ifmtargexp{\@pdftitle}{%
347     \PackageWarningNoLine{hyperxmp}{%
348       Missing pdftitle (required for PDF standards\MessageBreak
349       compliance)%
350     }%
351   }%
352   {}%
353 }%
354 }

```

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.

```

355 \AtEndPreamble{%
356   \ifpackageloaded{hyperref}{%

```

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

```

357   \@if@def@and@nonempty{\@pdflang}{%
358   }{%
359     \hyxmp@detect@langs
360   }%

```

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

```
361 \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 \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.

```
362 \@ifundefined{XeTeXversion}{\}%
363 \@ifmtargexp{\@pdfcreationdate}{\%
364 \let\@pdfcreationdate=\hyxmp@today@pdf
365 }%
366 }%
367 }
```

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

```
368 \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.

```
369 \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`.

```
370 \hyxmp@at@end{%
371 \hyxmp@find@metadata
372 \hyxmp@embed@packet
373 }%
374 }{%
375 \PackageWarningNoLine{hyperxmp}{\%
376 \jobname.tex failed to include a\MessageBreak
377 \string\usepackage\string{hyperref\string}
378 in the preamble.\MessageBreak
379 Consequently, all hyperxmp functionality will be\MessageBreak
380 disabled}%
381 }%
382 }
```

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}` and at the `\end{document}`. We additionally identify here the natural language(s) in use because these aren’t known until the end of the document when provided by `babel` or `polyglossia`. Also, `hyperref` forbids changes to `pdflang` past the `\begin{document}`.

```

383 \hyxmp@at@end{%
384   \hyxmp@set@dc@lang
385   \ifx\@pdfmetalang\@empty
386     \ifx\@pdflang\@empty
387       \let\@pdfmetalang=\hyxmp@x@default
388     \else
389       \edef\@pdfmetalang{\@pdflang}%
390     \fi
391   \fi
392   \hyxmp@xmllify\@pdfmetalang
393   \hyxmp@auto@assign@data
394 }
```

3.3 Advanced metadata detection

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

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

<pre> \hyxmp@set@koma@phones \hyxmp@koma@phones</pre>	<p>Define <code>\hyxmp@koma@phones</code> as a comma-separated list of the phone numbers provided to a Koma letter class (mobile and landline).</p> <pre> 395 \newcommand*{\hyxmp@set@koma@phones}{% 396 \@if@def@and@nonempty{scr@frommobilephone@var}{% 397 \@if@def@and@nonempty{scr@fromphone@var}{% 398 \edef\hyxmp@koma@phones{\scr@frommobilephone@var,\scr@fromphone@var}% 399 }{% 400 \edef\hyxmp@koma@phones{\scr@frommobilephone@var}% 401 }% 402 }{% 403 \@if@def@and@nonempty{scr@fromphone@var}{% 404 \edef\hyxmp@koma@phones{\scr@fromphone@var}% 405 }{%</pre>
---	--


```

406     }%
407 }%
408 }

```

`\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.

```

409 \newcommand*{\hyxmp@use@first@valid}[3]{%
410   \ifmtargexp{#2}{%
411     \hyxmp@use@first@valid@i{#1}#3,!,%
412   }%
413 }%
414 }

```

`\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`.

```

415 \def\hyxmp@use@first@valid@i#1#2,{%
416   \def\next{\hyxmp@use@first@valid@i{#1}}%
417   \ifx#2!%
418     \let\next=\relax
419   \else
420     \ifx#2\undefined
421     \else
422       \@ifnotmtargexp{#2}{%
423         \hypersetup{#1={#2}}%
424         \def\next##1!,{%
425           }%
426       \fi
427     \fi
428   \next
429 }

```

`\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).

```

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

```

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.

```

431   \hyxmp@use@first@valid{pdftitle}{\@pdftitle}{%
432     \scr@subject@var,%
433     \@title
434   }%
435   \hyxmp@use@first@valid{pdfauthor}{\@pdfauthor}{%
436     \scr@fromname@var,%
437     \@author

```

```

438 }%
439 \hyxmp@use@first@valid{pdfcontactemail}{\@pdfcontactemail}{%
440   \scr@fromemail@var
441 }%
442 \hyxmp@set@koma@phones
443 \hyxmp@use@first@valid{pdfcontactphone}{\@pdfcontactphone}{%
444   \hyxmp@koma@phones
445 }%
446 \hyxmp@use@first@valid{pdfcontacturl}{\@pdfcontacturl}{%
447   \scr@fromurl@var
448 }%
449 \hyxmp@use@first@valid{pdfsubtitle}{\@pdfsubtitle}{%
450   \@subtitle
451 }%
452 \hyxmp@use@first@valid{pdfpublisher}{\@pdfpublisher}{%
453   \@publishers
454 }%

```

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.

```

455 \hyxmp@parse@acmart
456 }

```

`\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.

```

457 \newcommand*{\hyxmp@parse@acmart}{%
458   \begin{group}

```

`\@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.

```

459   \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 460   \def\email##1##2{%
461     \def\hyxmp@address@val{##2}%
462     \hyxmp@use@first@valid{pdfcontactemail}{\@pdfcontactemail}{%
463       \hyxmp@address@val
464     }%
465   }%

```

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

```

\hyxmp@address@val 466   \def\streetaddress##1{%
467     \def\hyxmp@address@val{##1}%
468     \hyxmp@use@first@valid{pdfcontactaddress}{\@pdfcontactaddress}{%

```

```

469         \hyxmp@address@val
470     }%
471 }%

\city \city wraps the author's city name.
\hyxmp@address@val 472 \def\city##1{%
473     \def\hyxmp@address@val{##1}%
474     \hyxmp@use@first@valid{pdfcontactcity}{\@pdfcontactcity}{%
475         \hyxmp@address@val
476     }%
477 }%

\state \state wraps the author's state or region name.
\hyxmp@address@val 478 \def\state##1{%
479     \def\hyxmp@address@val{##1}%
480     \hyxmp@use@first@valid{pdfcontactregion}{\@pdfcontactregion}{%
481         \hyxmp@address@val
482     }%
483 }%

\country \country wraps the author's country name.
\hyxmp@address@val 484 \def\country##1{%
485     \def\hyxmp@address@val{##1}%
486     \hyxmp@use@first@valid{pdfcontactcountry}{\@pdfcontactcountry}{%
487         \hyxmp@address@val
488     }%
489 }%

\postcode \postcode wraps the author's postal code.
\hyxmp@address@val 490 \def\postcode##1{%
491     \def\hyxmp@address@val{##1}%
492     \hyxmp@use@first@valid{pdfcontactpostcode}{\@pdfcontactpostcode}{%
493         \hyxmp@address@val
494     }%
495 }%

\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.

496 \def\affiliation##1##2{%
497     ##2%
498     \let\affiliation=\gobbletwo
499 }%

```

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.

```
500   \setbox0=\hbox{\addresses}%
501   \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.

```
502   \if@def@and@nonempty{@acmDOI}{%
503     \IfSubStr{\@acmDOI}{10.1145/1122445.1122456}{}%
504     \IfSubStr{\@acmDOI}{10.1145/nnnnnnn.nnnnnnn}{}%
505     \hyxmp@use@first@valid{pdfdoi}{\@pdfdoi}{%
506       \@acmDOI
507     }%
508   }%
509 }%
510 }%
511 {}%
```

`\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.

```
512   \if@def@and@nonempty{@acmISBN}{%
513     \IfSubStr{\@acmISBN}{XXXX}{}%
514     \IfSubStr{\@acmISBN}{xxxx}{}%
515     \def\hyxmp@strip@isbn@date##1/##2!{##1}%
516     \edef\hyxmp@acm@isbn{%
517       \expandafter\hyxmp@strip@isbn@date\@acmISBN/!%
518     }%
519     \hyxmp@use@first@valid{pdfisbn}{\@pdfisbn}{%
520       \hyxmp@acm@isbn
521     }%
522   }%
523 }%
524 }%
525 {}%
```

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

```
526   \def\hyxmp@acm@publisher{Association for Computing Machinery}%
527   \hyxmp@use@first@valid{pdfpublisher}{\@pdfpublisher}{%
528     \hyxmp@acm@publisher
529   }%
```

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

```

530 \hyxmp@use@first@valid{pdfpublication}{\@pdfpublication}{%
531   \@journalName,%
532   \@acmBooktitle,%
533   \@acmConference
534 }%

```

`\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).

```

535 \if@ACM@journal
536   \def\hyxmp@acm@pubtype{journal}%
537 \else
538   \def\hyxmp@acm@pubtype{book}%
539 \fi
540 \hyxmp@use@first@valid{pdfpubtype}{\@pdfpubtype}{%
541   \hyxmp@acm@pubtype
542 }%

```

Journal articles have a volume and issue number.

```

543 \hyxmp@use@first@valid{pdfvolumenum}{\@pdfvolumenum}{%
544   \@acmVolume
545 }%
546 \hyxmp@use@first@valid{pdfissuenum}{\@pdfissuenum}{%
547   \@acmNumber
548 }%
549 }

```

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

```

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

```

`\hyxmp@set@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 3.6.3). Normally, these are the same. However, `Lang` accepts only a single language while `dc:language` accepts multiple languages. If the document loads either `babel` or `polyglossia` and does not specify `pdflang` in `\hypersetup`, `\hyxmp@set@dc@lang` is redefined below (in `\hyxmp@detect@langs`) to set `\hyxmp@dc@lang` to `babel`/`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`.

```

551 \newcommand*{\hyxmp@set@dc@lang}{%
552   \let\hyxmp@dc@lang=\@pdflang
553 }

```

`\hyxmp@detect@langs` If `pdflang` was not specified, try to determine the document language(s) using either `babel`'s or `polyglossia`'s definitions. If so, we redefine `\hyxmp@set@dc@lang` as a comma-separate list of languages used in the document.

```

554 \newcommand*{\hyxmp@detect@langs}{%
555   \ifundefined{mainbcp47id}{%
556     \ifundefined{LocaleForEach}{%

```

The document doesn't appear to have loaded either `babel` or `polyglossia`. In this case we have one small task to do. In older versions of `hyperref`, `\@pdflang` is set to `\@empty` if `pdflang` is not specified. In newer versions of `hyperref`, `\@pdflang` is set to `\relax` if `pdflang` is not specified. The latter is a bit problematic for `hyperxmp` because it makes `\@pdflang` non-expandable, which causes a literal “`\@pdflang`” to be written as XMP metadata. To avoid that situation we explicitly set `\@pdflang` to `\@empty` to avoid problems with non-expandable symbols.

```
557     \let\@pdflang=\@empty
558   }{%
```

Use `babel`'s `\LocaleForEach` and `\getlocaleproperty` to construct a comma-separated list of all document languages. We also set `\@pdflang` within `\hyxmp@set@dc@lang`. (`\hyxmp@detect@langs` should only be called if `\@pdflang` is empty.) Unlike `polyglossia`, `babel` does not provide information about the main language until after the `\begin{document}` so we have to wait until `\hyxmp@set@dc@lang` is called to set it.

```
559     \BabelEnsureInfo
560     \renewcommand*{\hyxmp@set@dc@lang}{%
561       \getlocaleproperty\@pdflang{\bbl@main@language}{identification/tag.bcp47}%
562       \hyxmp@write@pdflang
563       \let\hyxmp@dc@lang=\relax
564       \LocaleForEach{%
565         \getlocaleproperty\hyxmp@lang@tag{####1}{identification/tag.bcp47}%
566         \ifx\hyxmp@dc@lang\relax
567           \edef\hyxmp@dc@lang{\hyxmp@lang@tag}%
568         \else
569           \edef\hyxmp@dc@lang{\hyxmp@dc@lang,\hyxmp@lang@tag}%
570         \fi
571       }%
572     }%
573   }%
574 }{%
```

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

```
575     \xdef\@pdflang{\csname mainbcp47id\endcsname}%
576     \renewcommand*{\hyxmp@set@dc@lang}{%
577       \edef\hyxmp@dc@lang{\xpg@bcp@loaded}%
578     }%
579   }%
580 }
```

`\hyxmp@write@pdflang` Here's a conundrum: `babel` doesn't provide access to information on the document's main language until after the `\begin{document}`. However, `hyperref` allows `pdflang` to be set only *before* the `\begin{document}`. So what do we do?

Note that the underlying `\@pdflang` macro serves two purposes: `hyperref` uses it to set the `Lang` entry in the PDF document catalog, and `hyperxmp` uses it as the default metadata language. The latter is used in `dc:title` and other XMP properties that can include per-language variants. We set `\@pdflang` once we have the

main-language information from babel. As this is too late for `hyperref`, we bypass `hyperref` and manually write the `Lang` key into the document catalog. This is a bit kludgy, but we do leverage as much of `hyperref`'s infrastructure as possible to increase `\hyxmp@write@pdflang`'s robustness.

```
581 \newcommand*{\hyxmp@write@pdflang}{%
582   \ifundefined{pdfmark}{%
583     \ifundefined{@pdfmark}{%
584       \ifundefined{pdfcatalog}{%
```

We don't recognize the `TEX` engine or PDF generator. (This is unexpected.) We therefore issue a warning message and take no further action.

```
585       \PackageWarning{hyperxmp}{Unknown PDF generator; not setting the language (\@pdflang) in
586     }{%
```

We're running either pdf`TEX` or Lua`TEX`. (`hyperref` defines `\pdfcatalog` in the latter case.)

```
587       \pdfcatalog{/Lang (\@pdflang)}%
588     }%
589   }{%
```

We're running either X_Y`TEX` or `TEX` + `dvipdfm`..

```
590       \@pdfmark{docview << /Lang (\@pdflang) >>}%
591     }%
592 }{%
```

We're running either `TEX` + `Dvips` + Adobe Acrobat Distiller (or other converter from PostScript to PDF).

```
593   \pdfmark{%
594     pdfmark=/PUT,Raw={%
595       \string{Catalog\string} <<
596       /Lang (\@pdflang)
597       >>
598     }%
599   }%
600 }%
601 }
```

3.4 Manipulating author-supplied data

The author provides metadata information to `hyperxmp` via package options to `hyperref` or via `hyperref`'s `\hypersetup` command. The functions in this section convert author-supplied lists (e.g., `pdfkeywords={foo, bar, baz}`) into L^A`TEX` lists (e.g., `\@elt {foo} \@elt {bar} \@elt {baz}`) that can be more easily manipulated (Section 3.4.1); parse dates in both PDF and XMP formats (Section 3.4.2; trim spaces off the ends of strings (Section 3.4.3); convert text to XML (e.g., from `<scott+hyxmp@pakin.org>` to `<scott+hyxmp@pakin.org>`;) (Section 3.4.4); simplify the pretty-printing of a begin tag, XML text, and end tag (Section 3.4.5); and provide metadata in multiple languages (Section 3.4.6).

3.4.1 List manipulation

We define a macro for converting a list of comma-separated elements (e.g., the list of PDF keywords) to a list of L^AT_EX \@elt-separated elements.

```
\hyxmp@commas@to@list  Given a macro name (#1) and a comma-separated list (#2), define the macro name
                        as the elements of the list, each preceded by \@elt. (Executing the macro therefore
                        applies \@elt to each element in turn.)
602 \newcommand*{\hyxmp@commas@to@list}[2]{%
603   \gdef#1{%
604     \expandafter\hyxmp@commas@to@list@i\expandafter#1#2,,%
605   }

\hyxmp@commas@to@list@i  Recursively construct macro #1 from comma-separated list #2. Stop if #2 is empty.
\next 606 \def\hyxmp@commas@to@list@i#1#2,{%
607   \gdef\hyxmp@sublist{#2}%
608   \ifx\hyxmp@sublist\@empty
609     \let\next=\relax
610   \else
611     \hyxmp@trimspaces\hyxmp@sublist
612     \@cons{#1}{\hyxmp@sublist}%
613     \def\next{\hyxmp@commas@to@list@i{#1}}%
614   \fi
615   \next
616 }

\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.
617 \def\xmpcomma{,}%

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

618 \bgroup
619   \catcode'\^~C=11
620   \gdef\hyxmp@comma{^~C}
621 \egroup

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

622 \bgroup
623   \catcode'\^~U=11
624   \gdef\hyxmp@uscore{^~U}
625 \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.

```
626 \let\xmpquote=\relax
```

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

```
627 \bgroup
628 \catcode'\~ =12%
629 \gdef\xmptilde{~}%
630 \egroup
```

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

```
631 \newcommand{\XMPTruncateList}[1]{%
632   \PackageWarning{hyperxmp}{%
633     \noexpand\XMPTruncateList has been deprecated since\MessageBreak
634     hyperxmp 4.0 and may be removed in future\MessageBreak
635     versions of the package. \noexpand\XMPTruncateList\MessageBreak
636     was found}%
637   \edef\hyxmp@temp@str{\csname hyxmp@#1\endcsname}%
638   \hyxmp@commas@to@list{\hyxmp@temp@list}{\hyxmp@temp@str}%
639   \def\@elt##1{%
640     \expandafter\gdef\csname @#1\endcsname{##1}%
641     \let\@elt=\@gobble
642   }
643   \hyxmp@temp@list
644 }
```

3.4.2 Date manipulation

`hyperxmp` needs to manipulate two types of date (really, timestamp) formats: PDF format and XMP format. PDF timestamps are of the form “D:YYYYMMDDhhmmss+TT'tt'” (e.g., D:20200619092013-06'00') [3], while XMP timestamps are of the form “YYYY-MM-DDThh:mm:ss+TT:tt” (e.g., 2020-06-19T09:20:13-06:00) [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 645 \def\hyxmp@first@char#1{\hyxmp@first@char@i#1\relax}
646 \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.

```

647 \def\hyxmp@as@xmp@date#1{%
648   \expandafter\ifnum\expandafter'\hyxmp@first@char@i#1\relax='D
649     \hyxmp@pdf@to@xmp@date{#1}%
650   \else
651     #1%
652   \fi
653 }

```

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

```

654 \def\hyxmp@pdf@to@xmp@date#1:#2#3#4#5#6#7#8#9{%
655   #2#3#4#5-#6#7-#8#9%
656   \hyxmp@parse@time
657 }

```

`\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`.

```

658 \def\hyxmp@parse@time#1#2#3#4#5#6{%
659   T#1#2:#3#4:#5#6%
660   \hyxmp@parse@tz@char
661 }

```

`\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.

```

662 \def\hyxmp@parse@tz@char#1{%
663   #1%
664   \ifx#1-%
665     \expandafter\hyxmp@parse@tz
666   \else
667     \ifx#1+%
668       \expandafter\hyxmp@parse@tz
669     \fi
670   \fi
671 }

```

`\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.

```

672 \def\hyxmp@parse@tz#1'#2' {%
673   #1:#2%
674 }

```

`\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.

```

675 \def\hyxmp@as@pdf@date#1 {%
676   \expandafter\ifx\hyxmp@first@char@i#1\relax D%
677     #1%
678   \else
679     \hyxmp@xmp@to@pdf@date{#1}%
680   \fi
681 }

```

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

```

682 \def\hyxmp@xmp@to@pdf@date#1 {%
683   D:\hyxmp@xmp@to@pdf@date@i#1\relax\relax
684 }

```

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

```

685 \def\hyxmp@xmp@to@pdf@date@i#1#2#3#4#5#6 {%
686   #1#2#3#4%
687   \ifx#5-%
688     \expandafter\hyxmp@xmp@to@pdf@date@ii\expandafter#6%
689   \fi
690 }

```

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

```

691 \def\hyxmp@xmp@to@pdf@date@ii#1#2#3#4 {%
692   #1#2%
693   \ifx#3-%
694     \expandafter\hyxmp@xmp@to@pdf@date@iii\expandafter#4%
695   \fi
696 }

```

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

```

697 \def\hyxmp@xmp@to@pdf@date@iii#1#2#3#4 {%
698   #1#2%
699   \ifx#3T%
700     \expandafter\hyxmp@xmp@to@pdf@date@iv\expandafter#4%
701   \fi
702 }

```

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

```

703 \def\hyxmp@xmp@to@pdf@date@iv#1#2#3#4 {%
704   #1#2%
705   \ifx#3:%

```

```

706     \expandafter\hyxmp@xmp@to@pdf@date@v\expandafter#4%
707   \fi
708 }

```

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

```

709 \def\hyxmp@xmp@to@pdf@date@v#1#2#3#4{%
710   #1#2%
711   \ifx#3:%
712     \expandafter\hyxmp@xmp@to@pdf@date@vi\expandafter#4%
713   \fi
714 }

```

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

```

715 \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).

```

716 \def\hyxmp@xmp@to@pdf@date@vi#1#2#3#4{%
717   #1#2%
718   \ifx#3+%
719     +\expandafter\hyxmp@xmp@to@pdf@date@vii
720   \fi
721   \ifx#3-%
722     -\expandafter\hyxmp@xmp@to@pdf@date@vii
723   \fi
724   \ifx#3Z%
725     Z%
726   \fi
727   \ifx#3\relax
728     \expandafter\hyxmp@gobbletwo
729   \fi
730   \@gobbletwo #4%
731 }

```

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

```

732 \def\hyxmp@xmp@to@pdf@date@vii#1\@gobbletwo#2#3#4#5{%
733   #2#3%
734   \ifx#4:%
735     \expandafter\hyxmp@xmp@to@pdf@date@viii\expandafter#5%
736   \fi
737 }

```

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

```

738 \def\hyxmp@xmp@to@pdf@date@viii#1#2#3#4{%
739   '#1#2'%
740 }

```

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

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

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

```
742 \xdef#1{\the\year}%
743 \ifnum\month<10
744 \xdef#1{#1-0\the\month}%
745 \else
746 \xdef#1{#1-\the\month}%
747 \fi
748 \ifnum\day<10
749 \xdef#1{#1-0\the\day}%
750 \else
751 \xdef#1{#1-\the\day}%
752 \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.

```
753 \@tempcnta=\time
754 \divide\@tempcnta by 60
755 \ifnum\@tempcnta<10
756 \xdef#1{#1T0\the\@tempcnta}%
757 \else
758 \xdef#1{#1T\the\@tempcnta}%
759 \fi
760 \multiply\@tempcnta by -60
761 \advance\@tempcnta by \time
762 \ifnum\@tempcnta<10
763 \xdef#1{#1:0\the\@tempcnta}%
764 \else
765 \xdef#1{#1:\the\@tempcnta}%
766 \fi
767 \xdef#1{#1Z}%
768 }
```

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

```
769 \def\hyxmp@try@today#1#2{%
770 \ifmtargexp{\hyxmp@today@xmp}{%
771 \ifundefined{#1}{#2}%
772 }%
773 }%
774 }
```

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

```

775 \def\hyxmp@today@xmp{}
    Case 1: \pdfcreationdate is defined (pdfLATEX and pre-0.85 LuaLATEX).
776 \hyxmp@try@today{\pdfcreationdate}{%
777   \edef\hyxmp@today@xmp{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
778 }
    Case 2: \pdffeedback is defined (LuaLATEX 0.85+).
779 \hyxmp@try@today{\pdffeedback}{%
780   \edef\hyxmp@today@xmp{\expandafter\hyxmp@pdf@to@xmp@date\pdffeedback creationdate}%
781 }

\hyxmp@timestamp Case 3: \filemoddate is defined (XYLATEX). In this case, we treat the timestamp
of the job's .log file as the current date/time.
782 \hyxmp@try@today{\filemoddate}{%
783   \edef\hyxmp@today@xmp{\filemoddate{\jobname.log}}%
784   \edef\next{%
785     \edef\noexpand\hyxmp@today@xmp{\noexpand\hyxmp@as@xmp@date{\hyxmp@today@xmp}}%
786   }%
787   \next
788 }%
    Case 4: None of the above. Do the best we can using the available TEX primitives
(\year, \month, \day, and \time.
789 \hyxmp@try@today{\year}{%
790   \hyxmp@today@xmp@define\hyxmp@today@xmp
791 }

\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.
792 \expandafter\edef\expandafter\hyxmp@today@pdf\expandafter{%
793   \expandafter\hyxmp@xmp@to@pdf@date\expandafter{\hyxmp@today@xmp}%
794 }

```

3.4.3 Trimming leading and trailing spaces

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

```

\hyxmp@trimspaces Redefine a macro as its previous value but without leading or trailing spaces. This
code—as well as that for its helper macros, \hyxmp@trimb and \hyxmp@trimc—was
taken almost verbatim from a solution to an Around the Bend puzzle [6]. Inline
comments are also taken from the solution text.
795 \catcode'\Q=3
    \hyxmp@trimspaces\x redefines \x to have the same replacement text sans leading
and trailing space tokens.
796 \newcommand{\hyxmp@trimspaces}[1]{%

```

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

```
797 \begingroup
    Put “\toks 0 {” into the afterassignment queue.
798 \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.
799 \expandafter\hyxmp@trimb\expandafter\noexpand#1Q Q}%
    Transfer the trimmed text back into #1.
800 \edef#1{\the\toks0}%
801 }
```

`\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.

```
802 \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.

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

3.4.4 Converting text to XML

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

`\ifhyxmp@unicodetex` `XYTeX` and `LuaTeX` natively support Unicode. We define the conditional `\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.

```
805 \newif\ifhyxmp@unicodetex
806 \ifnum64='\^^^0040\relax
807 \hyxmp@unicodetextrue
808 \else
809 \hyxmp@unicodetexfalse
810 \fi
```

`\SE->pdfdoc003` 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.

```
811 \expandafter\def\csname SE->pdfdoc003\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.

```

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

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

813 \newcommand*{\hyxmp@xmlify}[1]{%
814 \gdef\hyxmp@xmlified{}%

Escaped PDF string → PDFDocEncoding/Unicode
815 \EdefUnescapeString\hyxmp@text{#1}%
816 \ifhyxmp@unicodetex

PDFDocEncoding/Unicode → UTF-32BE
817 \hyxmp@is@unicode\hyxmp@text{%
818 \StringEncodingConvert
819 \hyxmp@text\hyxmp@text{utf16be}{utf32be}%
820 }{%
821 \ifXeTeX
822 \hyxmp@xetex@crap
823 \else
824 \StringEncodingConvert
825 \hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
826 \fi
827 }%

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

UTF-32BE → XML in ASCII
829 \edef\hyxmp@text{%
830 \expandafter
831 }\expandafter\hyxmp@toxml@unicodetex\hyxmp@text
832 \relax\relax\relax\relax\relax\relax\relax\relax
833 \else

PDFDocEncoding/Unicode → UTF-8
834 \hyxmp@is@unicode\hyxmp@text{%
835 \StringEncodingConvert
836 \hyxmp@text\hyxmp@text{utf16be}{utf8}%
837 }{%
838 \StringEncodingConvert
839 \hyxmp@text\hyxmp@text{pdfdoc}{utf8}%
840 }%

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

```


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

```
842 \edef\hyxmp@text{%
843 \expandafter\hyxmp@toxml\hyxmp@text\@empty\@empty
844 }%
```

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

```
845 \EdefUnescapeHex\hyxmp@text{\hyxmp@text}%
846 \fi
847 \global\let\hyxmp@xmlified\hyxmp@text
848 }
```

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

```
849 \begingroup
850 \lccode'\<=254 %
851 \lccode'\>=255 %
852 \catcode254=12 %
853 \catcode255=12 %
854 \lowercase{\endgroup
855 \def\hyxmp@is@unicode#1{%
856 \expandafter\hyxmp@@is@unicode#1<>\@nil
857 }%
858 \def\hyxmp@@is@unicode#1<>#2\@nil{%
859 \ifx\#1\%
860 \expandafter\@firstoftwo
861 \else
862 \expandafter\@secondoftwo
863 \fi
864 }%
865 }
```

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

```
866 \def\hyxmp@toxml#1#2{%
867 \ifx#1\@empty
868 \else
869 \ifnum"#1#2='\& %
870 26616D703B% & ;
871 \else\ifnum"#1#2='\< %
872 266C743B% < ;
873 \else\ifnum"#1#2='\> %
874 2667743B% > ;
875 \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.

```

876      \@ifundefined{pdfmark}{%
877          #1#2%
878      }{%
879          \ifnum"#1#2='\'( %
880              5C28% \'(
881          \else\ifnum"#1#2='\' ) %
882              5C29% \' )
883          \else
884              #1#2%
885          \fi\fi
886      }%
887      \fi\fi\fi
888      \expandafter\hyxmp@toxml
889      \fi
890  }

```

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

```

891 \def\hyxmp@toxml@unicodetex#1#2#3#4#5#6#7#8{%
892     \ifx#1\relax
893     \else
894         \ifnum"#1#2#3#4#5#6#7#8>127 %
895             \uccode'\*="#1#2#3#4#5#6#7#8\relax
896             \uppercase{%
897                 \edef\hyxmp@text{\hyxmp@text *}%
898             }%
899         \else\ifnum"#7#8='\'< %
900             \edef\hyxmp@text{\hyxmp@text &lt;}%
901         \else\ifnum"#7#8='\'& %
902             \edef\hyxmp@text{\hyxmp@text &amp;}%
903         \else\ifnum"#7#8='\'> %
904             \edef\hyxmp@text{\hyxmp@text &gt;}%
905         \else\ifnum"#7#8='\' %
906             \edef\hyxmp@text{\hyxmp@text\space}%
907         \else
908             \uccode'\*="#7#8\relax
909             \uppercase{%
910                 \edef\hyxmp@text{\hyxmp@text *}%
911             }%
912         \fi\fi\fi\fi\fi
913         \expandafter\hyxmp@toxml@unicodetex
914     \fi
915 }

```

```

\hyxmp@skipzeros Skip over leading zeroes in the input argument.
916 \def\hyxmp@skipzeros#1{%
917   \ifx#10%
918     \expandafter\hyxmp@skipzeros
919   \fi
920 }

\hyxmp@xetex@crap \x In the case of XeTeX, the strings defined by \pdfstringdef can contain big
characters. In this case, the string is treated as Unicode.
\hyxmp@try 921 \begingroup
\hyxmp@crap@result 922 \def\x#1{\endgroup
\hyxmp@text 923   \def\hyxmp@xetex@crap{%
924     \edef\hyxmp@try{%
925       \expandafter\hyxmp@SpaceOther\hyxmp@text#1\@nil
926     }%
927     \let\hyxmp@crap@result=N%
928     \expandafter\hyxmp@crap@test\hyxmp@try\relax
929     \ifx\hyxmp@crap@result Y%
930       \let\hyxmp@text\@empty
931       \expandafter\hyxmp@crap@convert\hyxmp@try\relax
932     \else
933       \StringEncodingConvert\hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
934     \fi
935   }%
936 }
937 \x{ }

\hyxmp@SpaceOther Re-encode all spaces in a string with category code 12 (“other”).
938 \begingroup
939   \catcode'\~ =12 %
940   \lccode'\~ ='\ %
941   \lowercase{\endgroup
942   \def\hyxmp@SpaceOther#1 #2\@nil{%
943     #1%
944     \ifx\relax#2\relax
945       \expandafter\@gobble
946     \else
947       ~%
948       \expandafter\@firstofone
949     \fi
950     {\hyxmp@SpaceOther#2\@nil}%
951   }%
952 }

\hyxmp@crap@test Determine if we need to treat a string as Unicode.
953 \def\hyxmp@crap@test#1{%
954   \ifx#1\relax
955   \else
956     \ifnum'#1>127 %

```

```

957     \let\hyxmp@crap@result=Y%
958     \expandafter\expandafter\expandafter\hyxmp@skiptorelax
959   \else
960     \expandafter\expandafter\expandafter\hyxmp@crap@test
961   \fi
962 \fi
963 }

\hyxmp@skiptorelax Discard all tokens up to and including the first \relax.
964 \def\hyxmp@skiptorelax#1\relax{}

\hyxmp@crap@convert Convert a hexadecimal string to a number.
  \hyxmp@num 965 \def\hyxmp@crap@convert#1{%
  \hyxmp@text 966   \ifx#1\relax
967   \else
968     \edef\hyxmp@num{\number'#1}%
969     \ifnum\hyxmp@num>"FFFFFF %
970       \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"1000000}\relax
971       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
972       \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"1000000}}%
973     \else
974       \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
975     \fi
976     \ifnum\hyxmp@num>"FFFF %
977       \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"10000}\relax
978       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
979       \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"10000}}%
980     \else
981       \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
982     \fi
983     \ifnum\hyxmp@num>"FF %
984       \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"100}\relax
985       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
986       \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"100}}%
987     \else
988       \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
989     \fi
990     \ifnum\hyxmp@num>0 %
991       \lccode'\!=\hyxmp@num\relax
992       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
993     \else
994       \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
995     \fi
996     \expandafter\hyxmp@crap@convert
997   \fi
998 }

\hyxmp@zero Define a null character with category code 12 ("other").
999 \begingroup

```

```

1000 \catcode0=12 %
1001 \gdef\hyxmp@zero{^^00}%
1002 \endgroup

```

3.4.5 Outputting structured XML

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

`\hyxmp@extra@indent` This macro is used internally to increase the amount of indentation when writing certain XML data. It is normally defined as empty but can temporarily be redefined to a sequence of `\space` characters.

```

1003 \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.

```

1004 \newcommand*{\hyxmp@add@simple}[2]{%
1005 \ifnotmtargexp{#2}{%
1006 \hyxmp@xmllify{#2}%
1007 \hyxmp@add@to@xml{\hyxmp@extra@indent_<}}%
1008 \xdef\hyxmp@xml{\hyxmp@xml#1}%
1009 \hyxmp@add@to@xml{>\hyxmp@xmllified</}%
1010 \xdef\hyxmp@xml{\hyxmp@xml#1>^^J}%
1011 }%
1012 }

```

`\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.

```

1013 \newcommand*{\hyxmp@add@simple@var}[2]{%
1014 \expandafter\ifx\csname#2\endcsname\relax
1015 \else
1016 \hyxmp@xmllify{\csname#2\endcsname}%
1017 \hyxmp@add@to@xml{%
1018 \hyxmp@extra@indent_<#1>\hyxmp@xmllified</#1>^^J%
1019 }%
1020 \fi
1021 }

```

`\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.

```

1022 \newcommand*{\hyxmp@add@simple@lang}[2]{%
1023   \ifnotmtarg{#2}{%
1024     \hyxmp@xmlify{#2}%
1025     \expandafter\hyxmp@add@simple@lang@i\hyxmp@xmlified\relax{#1}%
1026   }%
1027 }

```

`\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.

```

1028 \newcommand*{\hyxmp@add@simple@lang@i}{%
1029   \ifnextchar[\hyxmp@add@simple@lang@iif\hyxmp@add@simple@lang@i[]}%
1030 }

```

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

```

1031 \def\hyxmp@add@simple@lang@iif[#1]#2\relax#3{%
1032   \ifnotmtarg{#2}{%
1033     \hyxmp@xmlify{#2}%
1034     \ifmtarg{#1}{%
1035       \hyxmp@add@to@xml{%
1036         <#3>\hyxmp@xmlified</#3>^^J%
1037       }%
1038     }%
1039     \hyxmp@add@to@xml{%
1040       <#3 xml:lang="#1">\hyxmp@xmlified</#3>^^J%
1041     }%
1042   }%
1043 }%
1044 }

```

`\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.

```

1045 \newcommand*{\hyxmp@add@simple@pfx}[3]{%
1046   \ifnotmtargexp{#3}{%
1047     \hyxmp@add@to@xml{\hyxmp@extra@indent_____<}%
1048     \xdef\hyxmp@xml{\hyxmp@xml#1}%
1049     \hyxmp@pdfstringdef\hyxmp@iprefix{#2}%
1050     \hyxmp@xmlify{\hyxmp@iprefix}%
1051     \hyxmp@add@to@xml{>\hyxmp@xmlified}%
1052     \hyxmp@xmlify{#3}%
1053     \hyxmp@add@to@xml{\hyxmp@xmlified</}%
1054     \xdef\hyxmp@xml{\hyxmp@xml#1>^^J}%

```

```

1055 }%
1056 }

```

3.4.6 Providing metadata in multiple languages

Certain XMP tags—dc:title, dc:description, and dc:rights (and others? Let me know.)—can be expressed in multiple languages. The same text is used for both language pdfmetalang (default: pdflang) and language “x-default”. To express the same metadata in multiple languages, we provide an \XMPLangAlt macro to construct a list of alternative forms for a piece of metadata.

\hyxmp@alt@title Each of these macros is a list in which each element is of the form “\do <language>
\hyxmp@alt@description <text>” in which <language> is an ISO 639-1 two-letter country code with an optional
\hyxmp@alt@rights ISO 3166-1 two-letter region code. For example, \hyxmp@alt@title may contain
an element, “\do {es-MX} {Este es mi documento}”.

```

1057 \def\hyxmp@alt@title{}
1058 \def\hyxmp@alt@description{}
1059 \def\hyxmp@alt@rights{}

```

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

```

1060 \newcommand{\hyxmp@LA@accept}[2]{%
1061   \define@key{hyxmp@LA}{#1}{%

```

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

```

1062   \hyxmp@pdfstringdef\hyxmp@value{##1}%
1063   \xdef#2{#2\noexpand\do {\hyxmp@cur@lang} {\hyxmp@value}}%
1064   }
1065 }

```

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

```

1066 \hyxmp@LA@accept{pdftitle}{\hyxmp@alt@title}
1067 \hyxmp@LA@accept{pdfsubject}{\hyxmp@alt@description}
1068 \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.

```

1069 \newcommand{\XMPLangAlt}[2]{%
1070   \let\do=\relax

```

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

```
1071 \edef\hyxmp@cur@lang{#1}%
1072 \setkeys{hyxmp@LA}{#2}%
1073 }
```

3.5 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 \TeX 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.

```
1074 \def\hyxmp@modulo@a#1{%
1075   \@tempcntb=\@tempcnta
1076   \divide\@tempcntb by #1
1077   \multiply\@tempcntb by #1
1078   \advance\@tempcnta by -\@tempcntb
1079 }
```

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

```
\hyxmp@big@prime@ii 1080 \def\hyxmp@big@prime{536870923}
1081 \def\hyxmp@big@prime@ii{536870027}
```

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

```
\hyxmp@one@token 1082 \def\hyxmp@seed@rng#1{%
1083   \@tempcnta=\hyxmp@big@prime
1084   \futurelet\hyxmp@one@token\hyxmp@seed@rng@i#1\@empty
1085 }
```

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

```
\next 1086 \def\hyxmp@seed@rng@i{%
1087   \ifx\hyxmp@one@token\@empty
1088     \let\next=\relax
1089   \else
1090     \def\next##1{%
1091       \multiply\@tempcnta by 3
1092       \advance\@tempcnta by '##1
1093       \hyxmp@modulo@a{\hyxmp@big@prime}%
1094       \futurelet\hyxmp@one@token\hyxmp@seed@rng@i
1095     }%
1096   \fi
1097 \next
1098 }
```


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

```

1099 \def\hyxmp@set@rand@num{%
1100   \@tempcnta=\hyxmp@rand@num
1101   \multiply\@tempcnta by 3
1102   \advance\@tempcnta by \hyxmp@big@prime@ii
1103   \hyxmp@modulo@a{\hyxmp@big@prime}%
1104   \xdef\hyxmp@rand@num{\the\@tempcnta}%
1105 }

```

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

```

1106 \def\hyxmp@append@hex#1{%
1107   \hyxmp@set@rand@num
1108   \@tempcnta=\hyxmp@rand@num
1109   \hyxmp@modulo@a{16}%
1110   \ifnum\@tempcnta<10
1111     \xdef#1{#1\the\@tempcnta}%
1112   \else

```

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

```

1113     \advance\@tempcnta by -10
1114     \ifcase\@tempcnta
1115       \xdef#1{#1a}%
1116     \or\xdef#1{#1b}%
1117     \or\xdef#1{#1c}%
1118     \or\xdef#1{#1d}%
1119     \or\xdef#1{#1e}%
1120     \or\xdef#1{#1f}%
1121   \fi
1122 \fi
1123 }

```

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

```

1124 \def\hyxmp@append@hex@iii#1{%
1125   \hyxmp@append@hex#1%
1126   \hyxmp@append@hex#1%
1127   \hyxmp@append@hex#1%
1128 }

```

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

```

1129 \def\hyxmp@append@hex@iv#1{%
1130   \hyxmp@append@hex@iii#1%
1131   \hyxmp@append@hex#1%
1132 }

```

`\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`.

```

1133 \def\hyxmp@create@uuid#1{%
1134   \def#1{uuid:}%
1135   \hyxmp@append@hex@iv#1%
1136   \hyxmp@append@hex@iv#1%
1137   \g@addto@macro#1{-}%
1138   \hyxmp@append@hex@iv#1%
1139   \g@addto@macro#1{-4}%
1140   \hyxmp@append@hex@iii#1%
1141   \g@addto@macro#1{-}%

```

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

```

1142   \hyxmp@set@rand@num
1143   \@tempcnta=\hyxmp@rand@num
1144   \hyxmp@modulo@a{4}%
1145   \ifcase\@tempcnta
1146     \g@addto@macro#1{8}%
1147     \or\g@addto@macro#1{9}%
1148     \or\g@addto@macro#1{a}%
1149     \or\g@addto@macro#1{b}%
1150   \fi
1151   \hyxmp@append@hex@iii#1%
1152   \g@addto@macro#1{-}%
1153   \hyxmp@append@hex@iv#1%
1154   \hyxmp@append@hex@iv#1%
1155   \hyxmp@append@hex@iv#1%
1156 }

```

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

```

1157 \newcommand*{\hyxmp@def@DocumentID}{%
1158   \edef\hyxmp@seed@string{\jobname:\@pdftitle:\@pdfauthor:}%
1159   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
1160   \edef\hyxmp@rand@num{\the\@tempcnta}%
1161   \hyxmp@create@uuid\hyxmp@DocumentID
1162 }

```

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

```

1163 \newcommand*{\hyxmp@def@InstanceID}{%

```

```

1164 \hyxmp@today@xmp@define{\hyxmp@seed@string}%
1165 \edef\hyxmp@seed@string{%
1166   \jobname:\@pdftitle:\@pdfauthor:\hyxmp@today@xmp:\hyxmp@seed@string
1167 }%
1168 \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
1169 \edef\hyxmp@rand@num{\the\@tempcnta}%
1170 \hyxmp@create@uuid\hyxmp@InstanceID
1171 }

```

3.6 Constructing the XMP packet

An XMP packet “shall consist of the following, in order: a header PI, the serialized XMP data model (the XMP packet) with optional white-space padding, and a trailer PI” [4]. (“PI” is an abbreviation for “processing instructions”). The serialized XMP includes blocks of XML for various XMP schemata: Adobe PDF (Section 3.6.2), Dublin Core (Section 3.6.3), XMP Rights Management (Section 3.6.4), XMP Media Management (Section 3.6.5), XMP Basic (Section 3.6.6), Photoshop (Section 3.6.7), IPTC Photo Metadata (Section 3.6.9), and PDF/* Identification (Section 3.6.8). The `\hyxmp@construct@packet` macro (Section 3.6.12) 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.

3.6.1 XMP utility functions

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

```

1172 \newcommand*{\hyxmp@add@to@xml}[1]{%
1173   \bgroup
1174   \@tempcnta=0
1175   \ifhyxmp@unicodetex
1176     \@tempcntb=65536%
1177   \else
1178     \@tempcntb=256%
1179   \fi
1180   \loop
1181     \lccode\@tempcnta=\@tempcnta
1182     \advance\@tempcnta by 1
1183     \ifnum\@tempcnta<\@tempcntb
1184       \repeat
1185       \lccode'\_='\ \relax
1186       \lccode'\^C='\,\relax
1187       \lccode'\^U='\_\relax
1188       \lowercase{\xdef\hyxmp@new@xml{#1}}%
1189       \xdef\hyxmp@xml{\hyxmp@xml\hyxmp@new@xml}%
1190   \egroup
1191 }

```

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

```
1192 \bgroup
1193 \catcode'\#=11
1194 \gdef\hyxmp@hash{#}
1195 \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.

```
1196 \bgroup
1197 \xdef\hyxmp+xml{%
1198 \hyxmp@add@to+xml{%
1199 -----^^J%
1200 }
1201 \xdef\hyxmp@padding{\hyxmp+xml}%
1202 \egroup
1203 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
1204 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
1205 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
1206 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
1207 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
```

`\hyxmp@x@default` Define an x-default string that we can use in comparisons with `\@pdfmetalang`.
1208 `\newcommand*{\hyxmp@x@default}{x-default}`

3.6.2 The Adobe PDF schema

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

`\@pdfproducer` Define `\@pdfproducer` using the banner string if available or the `TEX` engine’s
`\hyxmp@define@pdfproducer` version number if not.

```
1209 \newcommand*{\hyxmp@define@pdfproducer}{%
1210 \gdef\@pdfproducer{TeX}
1211 \ifLuaTeX
1212 \expandafter\hyxmp@banner@to@producer\expandafter{\luatexbanner}%
1213 \else
1214 \ifPDFTeX
1215 \expandafter\hyxmp@banner@to@producer\expandafter{\pdftexbanner}%
1216 \else
1217 \ifXeTeX
1218 \edef\@pdfproducer{XeTeX version \the\XeTeXversion\XeTeXrevision}%
1219 \fi
```

```

1220     \fi
1221     \fi
1222 }

\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  $\epsilon$ -TeX's
\scantokens primitive is available).
1223 \def\hyxmp@banner@to@producer#1{%
1224     \ifx\scantokens\relax
1225         \gdef\pdfproducer{#1}%
1226     \else
1227         {\scantokens{\makeatletter\hyxmp@remove@this#1\relax}}%
1228     \fi
1229 }

\pdfproducer Define \pdfproducer as a given banner string with the initial "This is" stripped
\hyxmp@remove@this off the beginning.
1230 \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.
1231 \AtBeginDocument{%
1232     \ifx\pdfproducer\relax
1233         \hyxmp@define@pdfproducer
1234     \fi
1235 }

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

\hyxmp@major@minor
1236 \newcommand*{\hyxmp@assign@major@minor}{%
1237     \ifundefined{pdfvariable}{%
1238         \ifundefined{pdfminorversion}{%
            Case 1: Neither \pdfvariable nor \pdfminorversion is defined (Xe2LaTeX and
            regular LaTeX).
1239         }{%
            Case 2: \pdfminorversion is defined (pdfLaTeX and pre-0.85 LuaLaTeX).
1240         \xdef\hyxmp@major@minor{\the\pdfminorversion}%
1241         \ifundefined{pdfmajorversion}{%
            Case 2(a): \pdfmajorversion is not defined (older versions of pdfLaTeX and
            LuaLaTeX).
1242         \xdef\hyxmp@major@minor{1.\hyxmp@major@minor}%
1243         }{%

```

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

```
1244     \xdef\hyxmp@major@minor{\the\pdfmajorversion.\hyxmp@major@minor}%
1245     }%
1246     }%
1247     }{%
```

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

```
1248     \xdef\hyxmp@major@minor{\the\pdfvariable majorversion.\the\pdfvariable minorversion}%
1249     }%
1250 }
```

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

```
1251 \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.
1252     \hyxmp@add@simple@var{pdf:Producer}{\pdfproducer}%
1253     \hyxmp@add@simple@var{pdf:Keywords}{\pdfkeywords}%
1254     \hyxmp@add@simple{pdf:Trapped}{\pdftrapped}%
1255     \hyxmp@assign@major@minor
1256     \hyxmp@add@simple@var{pdf:PDFVersion}{\hyxmp@major@minor}%
1257 }
```

3.6.3 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.

```
1258 \newcommand*{\hyxmp@rdf@dc}[3][\iffalse]{%
    Set \@tempswatrue only if the given text is nonempty or the provided conditional
    evaluates to TRUE.
1259     \ifmtargexp{#3}{\@tempswafalse}{\@tempswatrue}%
1260     #1
1261     \@tempswatrue
1262     \fi
    Append the corresponding XML only if \@tempswatrue.
1263     \if@tempswa
1264         \hyxmp+xmlify{#3}%
```

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

```

1265 \let\hyxmp@value=\hyxmp@xmlified
1266 \hyxmp@add@to@xml{%
1267 -----<dc:#2>^^J%
1268 -----<rdf:Alt>^^J%
1269 }%
1270 \ifx\@pdfmetalang\hyxmp@x@default
1271 \else
1272 \hyxmp@xmlify{\@pdfmetalang}%
1273 \hyxmp@add@to@xml{%
1274 -----<rdf:li xml:lang="\hyxmp@xmlified">\hyxmp@value</rdf:li>^^J%
1275 }%
1276 \fi
1277 \hyxmp@add@to@xml{%
1278 -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@value</rdf:li>^^J%
1279 }%

```

Include variants of the text expressed in other languages, as specified by the author using `\XMPLangAlt` (Section 3.4.6).

```

1280 \def\do##1##2{
1281 \hyxmp@xmlify{##2}%
1282 \hyxmp@add@to@xml{%
1283 -----<rdf:li xml:lang="##1">\hyxmp@xmlified</rdf:li>^^J%
1284 }%
1285 }%
1286 \csname hyxmp@alt@#2\endcsname

```

Complete this XMP element.

```

1287 \hyxmp@add@to@xml{%
1288 -----</rdf:Alt>^^J%
1289 -----</dc:#2>^^J%
1290 }%
1291 \fi
1292 }%

```

`\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.

```

1293 \newcommand*{\hyxmp@list@to@xml}[4][\iffalse]{%

```

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

```

1294 \@ifmtargexp{#4}{\@tempswafalse}{\@tempswatrue}%
1295 #1
1296 \@tempswatrue
1297 \fi

```

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

```

1298 \if@tempswa
1299 \hyxmp@add@to@xml{%
1300 -----<dc:#2>^^J%
1301 -----<rdf:#3>^^J%

```

```

1302     }%
1303     \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`.

```

1304     \hyxmp@xmlify{#4}%
1305     \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
1306     \def\@elt#1{%
1307         \hyxmp@add@to@xml{%
1308     -----<rdf:li>##1</rdf:li>^^J%
1309         }%
1310     }%
1311     \hyxmp@list
1312     \egroup
1313     \hyxmp@add@to@xml{%
1314     -----</rdf:#3>^^J%
1315     -----</dc:#2>^^J%
1316     }%
1317     \fi
1318 }

```

`\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.

```

1319 \newcommand{\hyxmp@singleton@dc}[3][Bag]{%
1320     \@ifnotmtargexp{#3}{%
1321         \hyxmp@xmlify{#3}%
1322         \hyxmp@add@to@xml{%
1323     -----<dc:#2>^^J%
1324     -----<rdf:#1>^^J%
1325     -----<rdf:li>\hyxmp@xmlified</rdf:li>^^J%
1326     -----</rdf:#1>^^J%
1327     -----</dc:#2>^^J%
1328     }%
1329     }
1330 }

```

`\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).

```

1331 \newcommand*{\hyxmp@cond@dc@identifier}[2]{%
1332     \ifx\hyxmp@xmlified\@empty
1333         \@ifnotmtargexp{#2}{%
1334             \hyxmp@add@simple@pfx{dc:identifier}{#1}{#2}%
1335         }%
1336     \fi
1337 }

```

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

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

```

1338 \newcommand*{\hyxmp@dc@schema}{%
1339   \hyxmp@add@simple{dc:format}{application/pdf}%
1340   \hyxmp@rdf@dc[\ifHy@pdfa]{title}{\@pdftitle}%
1341   \hyxmp@rdf@dc[\ifHy@pdfa]{description}{\@pdfsubject}%
1342   \hyxmp@rdf@dc{rights}{\@pdfcopyright}%
1343   \hyxmp@singleton@dc{publisher}{\@pdfpublisher}%
1344   \ifmtargexp{\@pdfdatetime}{%
1345     \hyxmp@singleton@dc[Seq]{date}{\hyxmp@today@xmp}%
1346   }{%
1347     \hyxmp@singleton@dc[Seq]{date}{\@pdfdatetime}%
1348   }%
1349   \hyxmp@singleton@dc{type}{\@pdftype}%
1350   \hyxmp@list@to@xml[\ifHy@pdfa]{creator}{Seq}{\hyxmp@pdfauthor}%
1351   \hyxmp@list@to@xml{subject}{Bag}{\hyxmp@pdfkeywords}%
1352   \ifx\@pdfsource\@empty
1353     \else
1354       \hyxmp@add@simple{dc:source}{\@pdfsource}%
1355     \fi
1356   \hyxmp@list@to@xml{language}{Bag}{\hyxmp@dc@lang}%
1357 % If |\@pdfidentifier| is empty, try setting it to each of |\@pdfdoi|,
1358 % |\@pdfeissn|, |\@pdfissn|, and |\@pdfisbn|, in turn, with proper
1359 % syntactic adjustments.
1360 %   \begin{macrocode}
1361   \ifmtargexp{\@pdfidentifier}{%
1362     \let\hyxmp@xmlified=\@empty
1363     \hyxmp@cond@dc@identifier{info:doi/}{\@pdfdoi}%
1364     \hyxmp@cond@dc@identifier{urn:ISSN:}{\@pdfeissn}%
1365     \hyxmp@cond@dc@identifier{urn:ISSN:}{\@pdfissn}%
1366     \hyxmp@cond@dc@identifier{urn:ISBN:}{\@pdfisbn}%
1367   }{%
1368     \hyxmp@add@simple{dc:identifier}{\@pdfidentifier}%
1369   }%
1370 }
```

3.6.4 The XMP Rights Management schema

`\hyxmp@xmpRights@schema` Add properties defined by the XMP Rights Management schema to the `\hyxmp@xml` macro. Currently, these are only the `xmpRights:Marked` property and the `xmpRights:WebStatement` property. If the author specified a copyright statement

we mark the document as copyrighted. If the author specified a license statement we include the URL in the metadata.

```
1371 \newcommand*{\hyxmp@xmpRights@schema}{%
```

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

```
1372 \let\hyxmp@rights=\@empty
1373 \ifx\@pdflicenseurl\@empty
1374 \else
1375 \def\hyxmp@rights{YES}%
1376 \fi
1377 \ifx\@pdfcopyright\@empty
1378 \else
1379 \def\hyxmp@rights{YES}%
1380 \fi
```

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

```
1381 \ifx\hyxmp@rights\@empty
1382 \else
1383 \ifx\@pdfcopyright\@empty
1384 \else
1385 \hyxmp@add@simple{xmpRights:Marked}{True}%
1386 \fi
1387 \hyxmp@add@simple{xmpRights:WebStatement}{\@pdflicenseurl}%
1388 \fi
1389 }
```

3.6.5 The XMP Media Management schema

`\hyxmp@mm@schema` Add properties defined by the XMP Media Management schema to the `\hyxmp+xml` macro. According to the XMP specification, the `xmpMM:DocumentID` property is supposed to uniquely identify a document, and the `xmpMM:InstanceID` property is supposed to change with each save operation [4]. As seen in Section 3.5, we do what we can to honor this intention from within a T_EX-based workflow. We additionally support the `xmpMM:VersionID` property, whose value is supplied by the author using `pdfversionid`.

```
1390 \gdef\hyxmp@mm@schema{%
1391 \ifmtargexp{\hyxmp@DocumentID}{\hyxmp@def@DocumentID}{}%
1392 \ifmtargexp{\hyxmp@InstanceID}{\hyxmp@def@InstanceID}{}%
1393 \hyxmp@add@simple{xmpMM:DocumentID}{\hyxmp@DocumentID}%
1394 \hyxmp@add@simple{xmpMM:InstanceID}{\hyxmp@InstanceID}%
1395 \hyxmp@add@simple{xmpMM:VersionID}{\@pdfversionid}%
1396 \hyxmp@add@simple{xmpMM:RenditionClass}{\@pdfrendition}%
1397 }
```

3.6.6 The XMP Basic schema

`\hyxmp@xmp@basic@schema` Add properties defined by the XMP Basic schema to the `\hyxmp+xml` macro. These include a bunch of dates (all set to the same value) and the base URL for the

document if specified with baseurl.

```
1398 \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.

```
1399 \ifmtargexp{\@pdfcreationdate}{%
1400   \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@today@xmp}%
1401 }{%
1402   \hyxmp@add@simple{xmp:CreateDate}{%
1403     \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfcreationdate}}%
1404 }%
```

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

```
1405 \ifmtargexp{\@pdfmoddate}{%
1406   \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@today@xmp}%
1407 }{%
1408   \hyxmp@add@simple{xmp:ModifyDate}{%
1409     \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfmoddate}}%
1410 }%
```

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

```
1411 \ifmtargexp{\@pdfmetadatetime}{%
1412   \hyxmp@add@simple{xmp:MetadataDate}{\hyxmp@today@xmp}%
1413 }{%
1414   \hyxmp@add@simple{xmp:MetadataDate}{\@pdfmetadatetime}%
1415 }%
```

Define the creation tool and the base URL.

```
1416 \hyxmp@add@simple{xmp:CreatorTool}{\@pdfcreator}%
1417 \hyxmp@add@simple{xmp:BaseURL}{\@baseurl}%
1418 }
```

3.6.7 The Photoshop schema

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

```
1419 \gdef\hyxmp@photoshop@schema{%
1420   \edef\hyxmp@photoshop@data{\@pdfauthor\@pdfcaptionwriter}%
1421   \hyxmp@add@simple{photoshop:AuthorsPosition}{\@pdfauthor}%
1422   \hyxmp@add@simple{photoshop:CaptionWriter}{\@pdfcaptionwriter}%
1423 }
```

3.6.8 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.

```

1424 \newcommand*{\hyxmp@pdfa@id@schema}{%
1425   \ifHy@pdfa
1426     \hyxmp@add@simple{pdfaid:part}{\@pdfapart}%
1427     \hyxmp@add@simple{pdfaid:conformance}{\@pdfaconformance}%
1428   \fi
1429 }
```

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

```

1430 \newcommand*{\hyxmp@pdfua@id@schema}{%
1431   \hyxmp@add@simple{pdfuaid:part}{\@pdfuapart}%
1432 }
```

`\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.

```

1433 \newcommand*{\hyxmp@pdfx@id@schema}{%
1434   \@tempcnta=0\hyxmp@pdfx@major\relax
1435   \ifnum\@tempcnta=0
1436   \else
1437     \ifnum\@tempcnta=1
1438       \hyxmp@add@simple{pdfx:GTS_PDFXVersion}{PDF/X-1:2001}%
1439       \hyxmp@add@simple{pdfx:GTS_PDFXConformance}{\@pdfxstandard}%
1440     \else
1441       \ifnum\@tempcnta<4
1442         \hyxmp@add@simple{pdfx:GTS_PDFXVersion}{\@pdfxstandard}%
1443       \else
1444         \hyxmp@add@simple{pdfx:GTS_PDFXVersion}{\@pdfxstandard}%
1445       \fi
1446     \fi
1447   \fi
1448 }
```

3.6.9 The IPTC Photo Metadata schema

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

```

1449 \begingroup
1450   \catcode'\&=12
1451   \catcode'\#=12
1452   \gdef\xmplinesep{&#xA;}
1453 \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.

```

1454 \newcommand*{\hyxmp@list@to@lines}[2]{%
1455   \@ifnotmtargexp{#2}{%
1456     \bgroup
1457     \hyxmp@add@to@xml{%
1458       \hyxmp@extra@indent_____<#1>%
1459     }%

```

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

```

1460   \def\@elt@first##1{%
1461     \hyxmp@add@to@xml{##1}%
1462     \let\@elt=\@elt@rest
1463   }%

```

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

```

1464   \def\@elt@rest##1{%
1465     \hyxmp@add@to@xml{\xmplinesep##1}%
1466   }%

```

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

```

1467   \let\@elt=\@elt@first
1468   \hyxmp@xmllify{#2}%
1469   \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmllified}%
1470   \hyxmp@list
1471   \hyxmp@add@to@xml{</#1>^^J}%
1472 \egroup
1473 }%
1474 }

```

`\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.

```

1475 \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.

```

1476   \ifx\hyxmp@iptc@data\@empty
1477   \else
1478     \hyxmp@add@to@xml{%
1479     _____<Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">^^J%
1480     }%

```

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

```

1481   \bgroup
1482   \edef\hyxmp@extra@indent{\hyxmp@extra@indent\space\space}%
1483   \hyxmp@list@to@lines{Iptc4xmpCore:CreatorContactInfo}{\@pdfcontactaddress}%

```

```

1484 \hyxmp@add@simple{Iptc4xmpCore:AdrCity}{\pdfcontactcity}%
1485 \hyxmp@add@simple{Iptc4xmpCore:AdrRegion}{\pdfcontactregion}%
1486 \hyxmp@add@simple{Iptc4xmpCore:AdrPcode}{\pdfcontactpostcode}%
1487 \hyxmp@add@simple{Iptc4xmpCore:AdrCtry}{\pdfcontactcountry}%

```

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

```

1488 \def\xmplinesep{,}%
1489 \hyxmp@list@to@lines{Iptc4xmpCore:TelWork}{\pdfcontactphone}%
1490 \hyxmp@list@to@lines{Iptc4xmpCore:EmailWork}{\pdfcontactemail}%
1491 \hyxmp@list@to@lines{Iptc4xmpCore:UrlWork}{\pdfcontacturl}%
1492 \egroup
1493 \hyxmp@add@to@xml{%
1494 -----</Iptc4xmpCore:CreatorContactInfo>^^J%
1495 }%
1496 \fi
1497 }

```

3.6.10 The PRISM Basic Metadata schema

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

```

1498 \newcommand*{\hyxmp@prism@schema}{%
1499 \ifx\hyxmp@prism@data\@empty
1500 \else
1501 \hyxmp@add@simple{prism:complianceProfile}{three}%
1502 \fi
1503 \hyxmp@add@simple@lang{prism:subtitle}{\pdfsubtitle}%
1504 \hyxmp@add@simple@lang{prism:publicationName}{\pdfpublication}%
1505 \hyxmp@add@simple{prism:aggregationType}{\pdfpubtype}%
1506 \hyxmp@add@simple@lang{prism:bookEdition}{\pdfbookedition}%
1507 \hyxmp@add@simple{prism:volume}{\pdfvolumenum}%
1508 \hyxmp@add@simple{prism:number}{\pdfissuenum}%
1509 \hyxmp@add@simple{prism:pageRange}{\pdfpagerange}%
1510 \hyxmp@add@simple{prism:isbn}{\pdfisbn}%
1511 \hyxmp@add@simple{prism:issn}{\pdfissn}%
1512 \hyxmp@add@simple{prism:eIssn}{\pdfeissn}%
1513 \hyxmp@add@simple{prism:doi}{\pdfdoi}%
1514 \hyxmp@add@simple{prism:url}{\pdfurl}%
1515 \hyxmp@add@simple{prism:byteCount}{\pdfbytes}%
1516 \hyxmp@add@simple{prism:pageCount}{\pdfnumpages}%
1517 }

```

3.6.11 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.

```
1518 \newcommand*{\hyxmp@check@iptc@data}{%
```

```
\hyxmp@iptc@data
```

```
1519 \edef\hyxmp@iptc@data{%
1520   \@pdfcontactaddress
1521   \@pdfcontactcity
1522   \@pdfcontactregion
1523   \@pdfcontactpostcode
1524   \@pdfcontactcountry
1525   \@pdfcontactphone
1526   \@pdfcontactemail
1527   \@pdfcontacturl
1528 }%
1529 }%
```

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

```
1530 \newcommand*{\hyxmp@check@prism@data}{%
```

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

```
1531 \edef\hyxmp@prism@data{%
1532   \@pdfbookedition
1533   \@pdfbytes
1534   \@pdfdoi
1535   \@pdfeissn
1536   \@pdfisbn
1537   \@pdfissn
1538   \@pdfissuenum
1539   \@pdfnumpages
1540   \@pdfpagerange
1541   \@pdfpublication
1542   \@pdfpubtype
1543   \@pdfsubtitle
1544   \@pdfurl
1545   \@pdfvolumenum
1546 }%
1547 }%
```

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

```
1548 \newcommand*{\hyxmp@begin@extension@decls}{%
1549   \hyxmp@add@to@xml{%
1550     <pdfaExtension:schemas>^^J%
1551     <rdf:Bag>^^J%
1552   }%
1553 }
```

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

```
1554 \newcommand*{\hyxmp@end@extension@decls}{%
1555   \hyxmp@add@to@xml{%
1556     </rdf:Bag>^^J%
1557     </pdfaExtension:schemas>^^J%
1558   }%
1559 }
```

`\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.

```
1560 \newcommand*{\hyxmp@begin@ext@decl}[3]{%
1561   \hyxmp@add@to@xml{%
1562     <rdf:li rdf:parseType="Resource">^^J%
1563     <pdfaSchema:schema>#1</pdfaSchema:schema>^^J%
1564     <pdfaSchema:prefix>#2</pdfaSchema:prefix>^^J%
1565     <pdfaSchema:namespaceURI>#3</pdfaSchema:namespaceURI>^^J%
1566     <pdfaSchema:property>^^J%
1567     <rdf:Seq>^^J%
1568   }%
1569 }
```

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

```
1570 \newcommand*{\hyxmp@end@ext@decl}{%
1571   \hyxmp@add@to@xml{%
1572     </rdf:Seq>^^J%
1573     </pdfaSchema:property>^^J%
1574     </rdf:li>^^J%
1575   }%
1576 }
```

`\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.

```
1577 \newcommand{\hyxmp@declare@property}[4][Text]{%
1578   \hyxmp@add@to@xml{%
1579     <rdf:li rdf:parseType="Resource">^^J%
1580     <pdfaProperty:name>}%
1581   \xdef\hyxmp@xml{\hyxmp@xml#2}%
1582   \hyxmp@add@to@xml{</pdfaProperty:name>^^J%
1583     <pdfaProperty:valueType>#1</pdfaProperty:valueType>^^J%
```



```

1584 -----<pdfaProperty:category>#3</pdfaProperty:category>^^J%
1585 -----<pdfaProperty:description>#4</pdfaProperty:description>^^J%
1586 -----</rdf:li>^^J%
1587  }%
1588 }%

```

`\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.

```

1589 \newcommand{\hyxmp@declare@field}[3][Text]{%
1590   \hyxmp@add@to+xml{%
1591     -----<rdf:li rdf:parseType="Resource">^^J%
1592     -----<pdfaField:name>#2</pdfaField:name>^^J%
1593     -----<pdfaField:valueType>#1</pdfaField:valueType>^^J%
1594     -----<pdfaField:description>#3</pdfaField:description>^^J%
1595     -----</rdf:li>^^J%
1596   }%
1597 }

```

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

```

1598 \newcommand*{\hyxmp@pdf@extensions}{%
1599   \hyxmp@begin@ext@decl
1600     {Adobe PDF Schema}%
1601     {pdf}%
1602     {http://ns.adobe.com/pdf/1.3/}%
1603   \hyxmp@declare@property
1604     {Trapped}%
1605     {internal}%
1606     {Indication if the document has been modified to include trapping information}%
1607   \hyxmp@end@ext@decl
1608 }%

```

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

```

1609 \newcommand*{\hyxmp@mm@extensions}{%
1610   \hyxmp@begin@ext@decl
1611     {XMP Media Management Schema}%
1612     {xmpMM}%
1613     {http://ns.adobe.com/xap/1.0/mm/}%
1614   \hyxmp@declare@property
1615     [URI]
1616     {DocumentID}%
1617     {internal}%
1618     {UUID based identifier for all versions and renditions of a document}%
1619   \hyxmp@declare@property
1620     [URI]
1621     {InstanceID}%
1622     {internal}%
1623     {UUID based identifier for specific incarnation of a document}%
1624   \hyxmp@declare@property

```

```

1625     {VersionID}%
1626     {internal}%
1627     {Document version identifier}%
1628 \hyxmp@declare@property
1629     {RenditionClass}%
1630     {internal}%
1631     {The manner in which a document is rendered}%
1632 \hyxmp@end@ext@decl
1633 }%

```

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

```

1634 \newcommand*{\hyxmp@pdfa@id@extensions}{%
1635 \hyxmp@begin@ext@decl
1636     {PDF/A Identification Schema}%
1637     {pdfaid}%
1638     {http://www.aiim.org/pdfa/ns/id/}%
1639 \hyxmp@declare@property
1640     [Integer]%
1641     {part}%
1642     {internal}%
1643     {Part of PDF/A standard}%
1644 \hyxmp@declare@property
1645     {conformance}%
1646     {internal}%
1647     {Conformance level of PDF/A standard}%
1648 \hyxmp@end@ext@decl
1649 }%

```

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

```

1650 \newcommand*{\hyxmp@pdfua@id@extensions}{%
1651 \hyxmp@begin@ext@decl
1652     {PDF/UA Universal Accessibility Schema}%
1653     {pdfuaid}%
1654     {http://www.aiim.org/pdfua/ns/id/}%
1655 \hyxmp@declare@property
1656     [Integer]%
1657     {part}%
1658     {internal}%
1659     {Part of ISO 14289 standard}%
1660 \hyxmp@end@ext@decl
1661 }%

```

`\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.

```

1662 \newcommand*{\hyxmp@pdfx@id@extensions}{%
1663 \ifx\hyxmp@pdfx@major\empty
1664 \else
1665 \hyxmp@begin@ext@decl
1666     {Adobe Document Info PDF/X eXtension Schema}%

```

```

1667     {pdfx}%
1668     {http://ns.adobe.com/pdfx/1.3/}%
1669     \hyxmp@declare@property
1670     {GTS_PDFXVersion}%
1671     {internal}%
1672     {ID of PDF/X standard}%
1673     \hyxmp@declare@property
1674     {GTS_PDFXConformance}%
1675     {internal}%
1676     {Conformance level of PDF/X standard}%
1677     \hyxmp@end@ext@decl
1678 \fi

```

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

```

1679 \@tempcnta=0\hyxmp@pdfx@major\relax
1680 \ifnum\@tempcnta>3
1681     \hyxmp@begin@ext@decl
1682     {PDF/X ID Schema}%
1683     {pdfxid}%
1684     {http://www.npes.org/pdfx/ns/id/}%
1685     \hyxmp@declare@property
1686     {GTS_PDFXVersion}%
1687     {internal}%
1688     {ID of PDF/X standard}%
1689     \hyxmp@end@ext@decl
1690 \fi
1691 }%

```

`\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.

```

1692 \newcommand*{\hyxmp@iptc@extensions}{%
1693     \hyxmp@begin@ext@decl
1694     {IPTC Core Schema}%
1695     {Iptc4xmpCore}%
1696     {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}%
1697     \hyxmp@declare@property
1698     [ContactInfo]
1699     {CreatorContactInfo}
1700     {external}
1701     {Document creator's contact information}

```

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

```

1702     \hyxmp@add@to+xml{%
1703     _____</rdf:Seq>^^J%
1704     _____</pdfaSchema:property>^^J%
1705     _____<pdfaSchema:valueType>^^J%
1706     _____<rdf:Seq>^^J%

```

```

1707 -----<rdf:li rdf:parseType="Resource">^^J%
1708 -----<pdfaType:type>ContactInfo</pdfaType:type>^^J%
1709 -----<pdfaType:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaType:
1710 -----<pdfaType:prefix>Iptc4xmpCore</pdfaType:prefix>^^J%
1711 -----<pdfaType:description>%
1712             Basic set of information to get in contact with a person%
1713             </pdfaType:description>^^J%
1714 -----<pdfaType:field>^^J%
1715 -----<rdf:Seq>^^J%
1716  }%
1717   \hyxmp@declare@field
1718       {CiAdrCity}%
1719       {Contact information city}%
1720   \hyxmp@declare@field
1721       {CiAdrCtry}%
1722       {Contact information country}%
1723   \hyxmp@declare@field
1724       {CiAdrExtadr}%
1725       {Contact information address}%
1726   \hyxmp@declare@field
1727       {CiAdrPcode}%
1728       {Contact information local postal code}%
1729   \hyxmp@declare@field
1730       {CiAdrRegion}%
1731       {Contact information regional information such as state or province}%
1732   \hyxmp@declare@field
1733       {CiEmailWork}%
1734       {Contact information email address(es)}%
1735   \hyxmp@declare@field
1736       {CiTelWork}%
1737       {Contact information telephone number(s)}%
1738   \hyxmp@declare@field
1739       {CiUrlWork}%
1740       {Contact information Web URL(s)}%
1741   \hyxmp@add@to+xml{%
1742 -----</rdf:Seq>^^J%
1743 -----</pdfaType:field>^^J%
1744 -----</rdf:li>^^J%
1745 -----</rdf:Seq>^^J%
1746 -----</pdfaSchema:valueType>^^J%
1747 -----</rdf:li>^^J%
1748  }%
1749 }

```

\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.

```

1750 \newcommand*{\hyxmp@prism@extensions}{%
1751   \hyxmp@begin@ext@decl

```

```

1752     {PRISM Basic Metadata}%
1753     {prism}%
1754     {http://prismstandard.org/namespaces/basic/2.1/}%
1755 \hyxmp@declare@property
1756     {complianceProfile}%
1757     {internal}%
1758     {PRISM specification compliance profile to which this document adheres}%
1759 \hyxmp@declare@property
1760     {publicationName}%
1761     {external}%
1762     {Publication name}%
1763 \hyxmp@declare@property
1764     {aggregationType}%
1765     {external}%
1766     {Publication type}%
1767 \hyxmp@declare@property
1768     {bookEdition}%
1769     {external}%
1770     {Edition of the book in which the document was published}%
1771 \hyxmp@declare@property
1772     {volume}%
1773     {external}%
1774     {Publication volume number}%
1775 \hyxmp@declare@property
1776     {number}%
1777     {external}%
1778     {Publication issue number within a volume}%
1779 \hyxmp@declare@property
1780     {pageRange}%
1781     {external}%
1782     {Page range for the document within the print version of its publication}%
1783 \hyxmp@declare@property
1784     {issn}%
1785     {external}%
1786     {ISSN for the printed publication in which the document was published}%
1787 \hyxmp@declare@property
1788     {eIssn}%
1789     {external}%
1790     {ISSN for the electronic publication in which the document was published}%
1791 \hyxmp@declare@property
1792     {isbn}%
1793     {external}%
1794     {ISBN for the publication in which the document was published}%
1795 \hyxmp@declare@property
1796     {doi}%
1797     {external}%
1798     {Digital Object Identifier for the document}%
1799 \hyxmp@declare@property
1800     [URL]
1801     {url}%

```

```

1802     {external}%
1803     {URL at which the document can be found}%
1804 \hyxmp@declare@property
1805     [Integer]
1806     {byteCount}%
1807     {internal}%
1808     {Approximate file size in octets}%
1809 \hyxmp@declare@property
1810     [Integer]
1811     {pageCount}%
1812     {internal}%
1813     {Number of pages in the print version of the document}%
1814 \hyxmp@declare@property
1815     {subtitle}%
1816     {external}%
1817     {Document's subtitle}%
1818 \hyxmp@end@ext@decl
1819 }%

```

\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.

```

1820 \newcommand*{\hyxmp@declare@extensions}{%

```

```

1821 \hyxmp@begin@extension@decls

```

Declare the Adobe PDF schema (always present).

```

1822 \hyxmp@pdf@extensions

```

Declare the XMP Media Management extensions (always present).

```

1823 \hyxmp@mm@extensions

```

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

```

1824 \ifHy@pdfa

```

```

1825 \hyxmp@pdfa@id@extensions

```

```

1826 \fi

```

Conditionally declare the PDF/UA Universal Accessibility extensions.

```

1827 \ifx\@pdfuapart\@empty

```

```

1828 \else

```

```

1829 \hyxmp@pdfua@id@extensions

```

```

1830 \fi

```

\next Conditionally declare the PDF/X extensions.

```

1831 \ifx\@pdfxversion\@empty

```

```

1832 \else

```

```

1833 \hyxmp@pdfx@id@extensions

```

```

1834 \fi

```

Conditionally declare IPTC photo metadata extensions.

```

1835 \ifx\hyxmp@iptc@data\@empty

```

```

1836 \else
1837   \hyxmp@iptc@extensions
1838 \fi

  Conditionally declare PRISM basic metadata extensions.

1839 \ifx\hyxmp@prism@data\@empty
1840 \else
1841   \hyxmp@prism@extensions
1842 \fi

1843 \hyxmp@end@extension@decls
1844 }

```

3.6.12 Combining schemata into an XMP packet

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

```

1845 \begingroup
1846   \ifhyxmp@unicodetex
1847     \lccode'\!="FEFF %
1848     \lowercase{%
1849       \gdef\hyxmp@bom{!}
1850     }%
1851   \else
1852     \catcode'\^^ef=12
1853     \catcode'\^^bb=12
1854     \catcode'\^^bf=12
1855     \gdef\hyxmp@bom{^^ef^^bb^^bf}%
1856   \fi
1857 \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.

```

1858 \def\hyxmp@construct@packet{%
1859   \gdef\hyxmp+xml{%
1860     \hyxmp@add@to+xml{<?xpacket begin="\hyxmp@bom" %
1861 id="W5MOMpCehiHzreSzNTczkc9d"?>^^J%
1862 <x:xmpmeta xmlns:x="adobe:ns:meta/">^^J%
1863 __<rdf:RDF %
1864 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\hyxmp@hash">^^J%
1865 ____<rdf:Description rdf:about=""^^J%

```

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

```

1866 _____xmlns:pdf="http://ns.adobe.com/pdf/1.3/"^^J%
1867 _____xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/"^^J%
1868 _____xmlns:dc="http://purl.org/dc/elements/1.1/"^^J%
1869 _____xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/"^^J%
1870 _____xmlns:xmp="http://ns.adobe.com/xap/1.0/"^^J%
1871 _____xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"^^J%
1872 _____xmlns:stEvt="http://ns.adobe.com/xap/1.0/sType/ResourceEvent\hyxmp@hash"^^J%
1873 _____xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/"^^J%

```

```

1874 -----xmlns:pdfuaid="http://www.aiim.org/pdfua/ns/id/"^^J%
1875 -----xmlns:pdfx="http://ns.adobe.com/pdfx/1.3/"^^J%
1876 -----xmlns:pdfxid="http://www.npes.org/pdfx/ns/id/"^^J%
1877 -----xmlns:prism="http://prismstandard.org/namespaces/basic/2.1/"^^J%
1878 -----xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"^^J%
1879 -----xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"^^J%
1880 -----xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema\hyxmp@hash"^^J%
1881 -----xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property\hyxmp@hash"^^J%
1882 -----xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type\hyxmp@hash"^^J%
1883 -----xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field\hyxmp@hash">^^J%
1884 }%

```

Declare non-standard schemata.

```

1885 \hyxmp@check@iptc@data
1886 \hyxmp@check@prism@data
1887 \hyxmp@declare@extensions

```

Insert all the metadata we know how to insert.

```

1888 \hyxmp@pdf@schema
1889 \hyxmp@xmpRights@schema
1890 \hyxmp@dc@schema
1891 \hyxmp@photoshop@schema
1892 \hyxmp@xmp@basic@schema
1893 \hyxmp@pdfa@id@schema
1894 \hyxmp@pdfua@id@schema
1895 \hyxmp@pdfx@id@schema
1896 \hyxmp@mm@schema
1897 \hyxmp@iptc@schema
1898 \hyxmp@prism@schema
1899 \hyxmp@add@to+xml{%
1900 ____</rdf:Description>^^J%
1901 __</rdf:RDF>^^J%
1902 </x:xmpmeta>^^J%
1903 \hyxmp@padding
1904 <?xpacket end="w"?>^^J%
1905 }%
1906 }

```

3.7 Embedding the XMP packet

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

```

\hyxmp@embed@packet Determine which hyperref driver is in use and invoke the appropriate embedding
\hyxmp@driver function.
1907 \newcommand*{\hyxmp@embed@packet}{%
1908 \hyxmp@construct@packet
1909 \def\hyxmp@driver{hpdfTeX}%
1910 \ifx\hyxmp@driver\Hy@driver

```



```

1911 \hyxmp@embed@packet@pdftex
1912 \else
1913 \def\hyxmp@driver{hlatex}%
1914 \ifx\hyxmp@driver\Hy@driver
1915 \hyxmp@embed@packet@luatex
1916 \else
1917 \def\hyxmp@driver{hdvipdfm}%
1918 \ifx\hyxmp@driver\Hy@driver
1919 \hyxmp@embed@packet@dvipdfm
1920 \else
1921 \def\hyxmp@driver{hxtex}%
1922 \ifx\hyxmp@driver\Hy@driver
1923 \hyxmp@embed@packet@xetex
1924 \else
1925 \@ifundefined{pdfmark}{%
1926 \PackageWarningNoLine{hyperxmp}{%
1927 Unrecognized hyperref driver ‘\Hy@driver’.\MessageBreak
1928 \jobname.tex’s XMP metadata will *not* be\MessageBreak
1929 embedded in the resulting file}%
1930 }{%
1931 \hyxmp@embed@packet@pdfmark
1932 }%
1933 \fi
1934 \fi
1935 \fi
1936 \fi
1937 }

```

3.7.1 Embedding using pdfTeX

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

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

```
1938 \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.

```

1939 \newcommand*{\hyxmp@embed@packet@pdftex}{%
1940   \bgroup
1941   \ifluatex
1942   \else
1943     \pdfcompresslevel=0
1944   \fi
1945   \immediate\pdfobj \ifluatex uncompressed\fi stream attr {%
1946     /Type /Metadata
1947     /Subtype /XML
1948   }{\hyxmp@xml}%
1949   \pdfcatalog {/Metadata \the\pdflastobj\space 0 R}%
1950 \egroup
1951 }

```

3.7.2 Embedding using LuaTeX 0.85+

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

```

1952 \newcommand*{\hyxmp@embed@packet@luatex}{%
1953   \immediate\pdfextension obj uncompressed stream attr {%
1954     /Type /Metadata
1955     /Subtype /XML
1956   }{\hyxmp@xml}%
1957   \pdfextension catalog {/Metadata \the\numexpr\pdffeedback lastobj\relax\space 0 R}%
1958 }

```

3.7.3 Embedding using any pdfmark-based backend

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

```

1959 \newcommand*{\hyxmp@embed@packet@pdfmark}{%
1960   \pdfmark{%
1961     pdfmark=/NamespacePush
1962   }%
1963   \pdfmark{%
1964     pdfmark=/OBJ,
1965     Raw={/_objdef \string\hyxmp@Metadata\string} /type /stream}%
1966   }%
1967   \pdfmark{%
1968     pdfmark=/PUT,
1969     Raw={\string\hyxmp@Metadata\string}
1970     2 dict begin
1971       /Type /Metadata def
1972       /Subtype /XML def
1973       currentdict
1974     end
1975   }%
1976 }%
1977 \pdfmark{%

```

```

1978 pdfmark=/PUT,
1979 Raw={\string{hyxmp@Metadata\string} (\hyxmp@xml)}}%
1980 }%
1981 \pdfmark{%
1982 pdfmark=/Metadata,
1983 Raw={\string{Catalog\string} \string{hyxmp@Metadata\string}}%
1984 }%
1985 \pdfmark{%
1986 pdfmark=/NamespacePop
1987 }%
1988 }

```

3.7.4 Embedding using dvipdfm

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

```

1989 \newcommand*{\hyxmp@embed@packet@dvipdfm}{%
1990 \hyxmp@string@len{\hyxmp@xml}}%
1991 \special{pdf: object @hyxmp@Metadata
1992 <<
1993 /Type /Metadata
1994 /Subtype /XML
1995 /Length \the\@tempcnta
1996 >>
1997 stream~J\hyxmp@xml endstream%
1998 }%
1999 \special{pdf: docview
2000 <<
2001 /Metadata @hyxmp@Metadata
2002 >>
2003 }%
2004 }

```

`\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.

```

2005 \newcommand*{\hyxmp@string@len}[1]{%
2006 \@tempcnta=0
2007 \expandafter\hyxmp@count@spaces#1 {} %
2008 \expandafter\hyxmp@count@non@spaces#1{}%
2009 }

```

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

```

2010 \def\hyxmp@count@spaces#1 {%
2011   \def\hyxmp@one@token{#1}%
2012   \ifx\hyxmp@one@token\@empty
2013     \advance\@tempcnta by -1
2014   \else
2015     \advance\@tempcnta by 1
2016     \expandafter\hyxmp@count@spaces
2017   \fi
2018 }

```

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

```

2019 \newcommand*{\hyxmp@count@non@spaces}[1]{%
2020   \def\hyxmp@one@token{#1}%
2021   \ifx\hyxmp@one@token\@empty
2022     \else
2023       \advance\@tempcnta by 1
2024       \expandafter\hyxmp@count@non@spaces
2025     \fi
2026 }

```

3.7.5 Embedding using X_YT_EX

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

```

2027 \newcommand*{\hyxmp@embed@packet@xetex}{%
2028   \special{pdf:stream @hyxmp@Metadata (\hyxmp@xml)
2029     <<
2030       /Type /Metadata
2031       /Subtype /XML
2032     >>
2033   }%
2034   \special{pdf:put @catalog
2035     <<
2036       /Metadata @hyxmp@Metadata
2037     >>
2038   }%
2039 }

```

3.8 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.

```

2040 \catcode'\\"=\hyxmp@dq@code

```

4 Help Wanted

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

A Sample XMP Packet

The following is an example of a complete XMP packet as may be produced by `hyperxmp`. This packet corresponds to the metadata included in the sample \LaTeX document presented on pages 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: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:lcnc/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.
- [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.
- [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.
- [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.
- [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.

- [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.
- [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	General: Initial version 1	v1.5	General: Made the XMP inclusion more robust. Thanks to Heiko Oberdiek for the bug report and suggested modifications. . . 17
v1.1	<code>\hyxmp@construct@packet:</code> Explicitly set the category codes of characters $\langle EF \rangle$, $\langle BB \rangle$, and $\langle BF \rangle$ to “letter”. Thanks to Daniel Schömer for the bug report 79	v2.0	<code>\ProcessKeyvalOptions:</code> Added this macro 28 <code>\XMPTruncateList:</code> Added this macro 41 <code>\hyxmp@ProcessKeyvalOptions:</code> Added this macro 28 <code>\hyxmp@SpaceOther:</code> Added by Heiko Oberdiek 51 <code>\hyxmp@add@simple:</code> Added this macro 53 <code>\hyxmp@add@to@xml:</code> Updated also to replace commas 59 <code>\hyxmp@bom:</code> Added by Heiko Oberdiek 79 <code>\hyxmp@comma:</code> Added this macro 40 <code>\hyxmp@construct@packet:</code> Modified by Heiko Oberdiek to use an appropriate BOM representation via <code>\hyxmp@bom</code> 79 <code>\hyxmp@crap@convert:</code> Added by Heiko Oberdiek 52 <code>\hyxmp@crap@test:</code> Added by Heiko Oberdiek 51 <code>\hyxmp@dc@schema:</code> Added support for <code>dc:language</code> and <code>dc:source</code> . 65 <code>\hyxmp@is@unicode:</code> Added by Heiko Oberdiek 49 <code>\hyxmp@list@to@xml:</code> Modified by Heiko Oberdiek to use the new
v1.2	General: Added support for the X _q TeX backend (<code>xdvipdfmx</code>) . . 1 Added support for the Photoshop schema 1 Made the package compatible with <code>ngerman</code> . Thanks to Tobias Mueller for the bug report. 17		
v1.3	General: Introduced the <code>pdfmetalang</code> package option, which enables an author to specify the language in which he wrote the document’s metadata 32		
v1.4	<code>\hyxmp@mm@schema:</code> Renamed the <code>xapMM</code> namespace prefix to <code>xmpMM</code> 66 <code>\hyxmp@rdf@dc:</code> Included metadata in the <code>x-default</code> language regardless of the specified metadata language 62 <code>\hyxmp@xmpRights@schema:</code> Renamed the <code>xapRights</code> namespace prefix to <code>xmpRights</code> 66		

Unicode-processing macros . . .	63	v2.1	
\hyxmp@photoshop@schema:			\hypersetup: Added this macro . . . 28
Simplified using			\hyxmp@hypersetup: Added this
\hyxmp@add@simple	67		macro 28
\hyxmp@skiptorelax: Added by			\hyxmp@redefine@Hyp: Added this
Heiko Oberdiek	52		macro 26
\hyxmp@skipzeros: Added by			General: Enabled hyperxmp and
Heiko Oberdiek	50		hyperref to be loaded in either
\hyxmp@toxml: Added by Heiko			order. This addresses a bug
Oberdiek	49		report by Yury Donskoy 26
Escaped parentheses written		v2.2	
with pdfmarks to prevent dvips			\hyxmp@iptc@extensions: Added
from line-wrapping the XMP			this macro to support PDF/A
packet	50		generation 75
\hyxmp@toxml@unicodetex: Added			\hyxmp@iptc@schema: Added this
by Heiko Oberdiek	50		macro 69
\hyxmp@xetex@crap: Added by			\hyxmp@list@to@lines: Added
Heiko Oberdiek	51		this macro 68
\hyxmp@xmllify: Completely			\xmpcomma: Changed the default
rewritten by Heiko Oberdiek to			from \relax to an ordinary
better support Unicode-enabled			comma 40
TeX programs	48		\xmplinesep: Added this macro . . 68
\hyxmp@xmp@basic@schema: Added			General: Added support for the
this macro	67		IPTC Photo Metadata schema . . 1
\hyxmp@xmpRights@schema:		v2.3	
Modified to include			\hyxmp@iptc@extensions: Gave
xmpRights:Marked only when			the
pdfcopyright is specified and			lptc4xmpCore:CreatorContactInfo
xmpRights:WebStatement only			fields a unique pdfaType:prefix
when pdflicenseurl is specified .	66		to better support conversion of
\hyxmp@zero: Added by Heiko			the document to PDF/A 75
Oberdiek	52	v2.3a	
\ifhyxmp@unicodetex: Added by			\hyxmp@detect@langs: Bug fix:
Heiko Oberdiek	47		Redefine \@pdflang as \@empty
\xmpcomma: Added this macro . . .	40		when hyperref has set it to
\xmpquote: Added this macro . . .	41		\relax 38
General: Added support for the		v2.3b	
XMP Basic schema and			\XMPTruncateList: Made all
miscellaneous other bits of			definitions local to avoid
metadata	1		spurious Too many
Heiko Oberdiek's major rewrite			unprocessed floats errors
of the code to better support			when running with memoir . . 41
native-Unicode TeX		v2.4	
implementations (XeTeX and			\hyxmp@add@simple@var: Added
LuaTeX)	1		this macro 53
New \AtBeginDocument code			\hyxmp@create@uuid: Modified
from Heiko Oberdiek to			this macro to produce a proper
properly encode			version 4 (random or
\@pdfmetalang	32		pseudorandom) UUID 58

\hyxmp@dc@schema: Made dc:language a Bag instead of an individual item so as to conform to the latest XMP specifications, a detail identified by Florian Breitwieser	65	v2.6	General: Added support for a new pdfdate key to explicitly specify the document date (and optionally time)	1
\hyxmp@parse@time: Added this macro	42	v2.7	\hyxmp@auto@assign@data: Automatically use \title and \author if pdftitle and pdfauthor are left unspecified. Thanks to Maciej Radziejewski for the suggestion	33
\hyxmp@parse@tz: Added this macro	42			
\hyxmp@parse@tz@char: Added this macro	42			
\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	62	v2.8	\hyxmp@add@to+xml: Corrected inadvertent lowercasing of non-Latin characters when run under Xe _{La} TeX or Lua _{La} TeX (bug reported by Leonid Sinev)	59
\hyxmp@pdf@to@xmp@date: Added this macro	42	v2.9	\hyxmp@iptc@schema: Use lptc4xmpCore instead of lptc4ContInfo as the contact-information metadata prefix. Leonid Sinev reports that Acrobat's PDF/A validator seems to prefer lptc4xmpCore	69
\hyxmp@pdfa@id@schema: Added this macro	68		\hyxmp@pdfa@id@schema: Let the author specify the PDF/A part and conformance IDs, as requested by Leonid Sinev	68
\hyxmp@today@xmp: Modified the code to parse the time and timezone from \pdfcreationdate, as proposed by Florian Breitwieser	45		General: Force inclusion of dc:creator, dc:title, and dc:description—even if empty—when hyperref is loaded with the pdfa option (suggested by Leonid Sinev)	1
\hyxmp@today@xmp@define: Added this macro	45		Introduced the pdftype package option, which enables an author to specify the type of document being produced	1
\hyxmp@xmp@to@pdf@date: Added this macro	43			
\xmp@tilde: Added this macro	41			
General: Added support for the PDF/A Identification schema, as requested by Florian Breitwieser	1			
v2.5				
\hyxmp@add@to+xml: Updated also to replace underscores and to modify only the text being added, not the already-modified text	59	v3.0	\hyxmp@embed@packet@luatex: Added this macro	82
\hyxmp@textunderscore: Added this macro	19		\hyxmp@today@xmp@define: Modified to accept the name of a macro to define	45
\hyxmp@uscore: Added this macro	40		\hyxmp@xmp@basic@schema: Made the XMP xmp:CreateDate, xmp:ModifyDate, and	
General: Enabled “_” to work within email addresses, as requested by Leonid Sinev	1			

xmp:MetadataDate match the PDF CreationDate	67	General: Don't overwrite an existing pdfmetalang with pdflang or x-default. This addresses a bug report by Niklas Beisert	32
General: Made the code compatible with LuaTeX 0.85. Thanks to Robert Schlicht, Leonid Sinev, and David Carlisle for bug reports and to Leonid Sinev for helping test the new hyperxmp code	1	v3.4 \hyxmp@seed@string: Correctly handle an author field of all spaces. Bug reported by Gaëtan Leurent	58
v3.1 \hyxmp@embed@packet@luatex: Updated to use \pdfextension obj uncompressed as suggested by Hans Hagen	82	General: Use ifmtarg to test for empty arguments, including non-empty but all spaces	1
\hyxmp@embed@packet@pdfTeX: Leave the XMP packet—and only the XMP packet—uncompressed in both pdfTeX and pre-0.85 LuaTeX	81	v3.5 \hyxmp@DocumentID: Added the pdfdocumentid option, at Michael Osipov's request	22
v3.2 \hyxmp@as@pdf@date: Added this macro	43	\hyxmp@InstanceID: Added the pdfinstanceid option, at Michael Osipov's request	22
\hyxmp@as@xmp@date: Added this macro	42	\hyxmp@mm@schema: Generate \hyxmp@DocumentID and \hyxmp@InstanceID only if the document does not already define these using the pdfdocumentid and pdfinstanceid options	66
\hyxmp@today@xmp@define: Modified to include hours and minutes	45	\hyxmp@seed@string: Seed with the TeX timestamp in addition to the document-specified timestamp	58
\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	67	v4.0 \XMPTruncateList: Deprecated this macro	41
v3.3 \@pdfsource: Added this macro and the corresponding pdfsource option, at Niklas Beisert's request	22	\hyxmp@add@simple@lang: Added this macro	54
\XMPLangAlt: Added this macro based on a request—and some code—by Niklas Beisert to support metadata expressed in multiple languages	55	\hyxmp@begin@ext@decl: Added this macro	72
\hyxmp@rdf@dc: Bug fix: Output the metadata language as correct XML even when hyperref is loaded with the unicode option	62	\hyxmp@declare@field: Replaced \hyxmp@declare@resource with this macro	73
		\hyxmp@declare@property: Added this macro	72
		\hyxmp@end@ext@decl: Added this macro	72
		\hyxmp@iptc@extensions: Moved the header code from here into \hyxmp@begin@extension@decls and the trailer code from here	

into		\hyxmp@check@standards: Added	
\hyxmp@end@extension@decls	75	this macro	29
Rewrote to more closely honor		\hyxmp@check@std: Added this	
the XMP specification	75	macro	21
\hyxmp@iptc@schema: Moved the		\hyxmp@declare@property: Insert	
definition of \hyxmp@iptc@data		the property name (#2)	
from here into		verbatim	72
\hyxmp@check@iptc@data	69	\hyxmp@define@pdfproducer:	
Renamed this macro to		Added this macro	60
\hyxmp@iptc@schema from		\hyxmp@no@info@lists: Renamed	
\hyxmp@photometa@schema	69	this macros from	
Rewrote this macro entirely to		\hyxmp@suppress@pdf@metadata	
correct the use of fields within a		and rewrote it to replace, if	
structure	69	possible, only Author and	
\hyxmp@mm@extensions: Added		Keywords	25
this macro	73	\hyxmp@pdf@extensions: Added	
\hyxmp@mm@schema: Include		this macro	73
xmpMM:VersionID in the XMP		\hyxmp@pdf@schema: Honor	
packet	66	pdftrapped	62
\hyxmp@no@info@lists: Added		\hyxmp@pdfua@id@extensions:	
this macro	25	Added this macro	74
\hyxmp@pdfa@id@extensions:		\hyxmp@pdfua@id@schema: Added	
Added this macro	74	this macro	68
\hyxmp@prism@extensions: Added		\hyxmp@pdfx@id@extensions:	
this macro	76	Added this macro	74
\hyxmp@prism@schema: Added this		\hyxmp@pdfx@id@schema: Added	
macro	70	this macro	68
General: Include all metadata		\hyxmp@today@pdf: Added this	
within a single pdf:Description		macro	46
block	1	\hyxmp@today@xmp: Support	
v4.1		X _Y TeX's \filemoddate	45
\hyxmp@singleton@dc: Added this		\hyxmp@today@xmp@define:	
macro	64	Modified to specify UTC	45
General: Invoke		General: Added support for	
\hyxmp@no@info@lists at the		PDF/UA standards, as requested	
beginning of the document, for		by Robin Schwab	1
compatibility with both newer		Added support for PDF/X	
and older versions of hyperref	31	standards, as requested by	
Updated the documentation to		Robin Schwab	1
refer to \pdfnumpages by its		Define a default producer	61
correct name. Thanks to Volker		Don't set any document dates	
RW Schaa for catching the		(creation, modification, or	
discrepancy	1	metadata) from pdfdate	1
v5.0		v5.1	
\@pdfrendition: Added the		\hyxmp@banner@to@producer:	
pdfrendition option	22	Prevent the category code of	
\@pdfxstandard: Added this macro	21	"@" from propagating past the	
\hyxmp@add@simple: Insert the tag		\begin{document}. Thanks to	
name (#1) verbatim	53	Robert Schlicht for noticing this	

catcode “leak” and providing a correction	61	Thanks to Nelson Posse Lago for pointing out why <code>atendddvi</code> is best avoided if possible	18
<code>\hyxmp@define@pdfproducer:</code> Check for Lua \TeX before checking for pdf \TeX to work around luatex85’s confusing <code>iftex</code> by defining <code>\pdftexversion</code> . Thanks to Robin Schwab for the bug report	60	<code>\hyxmp@auto@assign@data:</code> Consider other author-provided sources of metadata. Thanks to Robin Schwab for proposing that <code>hyperxmp</code> use the KOMA letter classes’s metadata	33
<code>\hyxmp@timestamp:</code> Don’t rely on <code>\jobname.aux</code> existing to query the current time under X \LaTeX . Instead, use <code>\jobname.log</code> . Thanks to Ulrike Fischer for the bug report and for her suggestion to use the log file. .	46	<code>\hyxmp@dc@schema:</code> Include all languages used in the document in <code>dc:language</code>	65
v5.2		<code>\hyxmp@detect@langs:</code> Acquire the default language from the <code>polyglossia</code> package, if loaded. Thanks to Robin Schwab for bringing that package to my attention	37
<code>\hyxmp@add@simple@pfx:</code> Added this macro	54	<code>\hyxmp@parse@acmart:</code> Added this macro	34
<code>\hyxmp@assign@major@minor:</code> Added this macro. <code>hyperxmp</code> now correctly specifies <code>pdf:PDFVersion</code> when generating PDF 2.0+. Thanks to Ulrike Fischer for alerting me to PDF 2.0’s availability in the \TeX ecosystem and informing me how to activate it	61	<code>\hyxmp@set@koma@phones:</code> Added this macro	32
<code>\hyxmp@cond@dc@identifier:</code> Added this macro	64	<code>\hyxmp@use@first@valid:</code> Added this macro	33
<code>\ifdraft:</code> Define <code>\ifdraft</code> only locally, at Niklas Beisert’s request	22	v5.4	
General: Introduced the <code>pdfidentifier</code> package option, which enables an author to specify a unique identifier for the document	1	<code>\hyxmp@dc@schema:</code> Bug fix: Use <code>\hyxmp@today@xmp</code> as the date only if <code>\@pdfdatetime</code> is undefined	65
v5.3		<code>\hyxmp@detect@langs:</code> Added support for <code>babel</code>	38
<code>\@if@def@and@nonempty:</code> Added this macro	18	Refactored language detection into a separate command	37
<code>\hyxmp@at@end:</code> Use <code>\AtEndDocument</code> in all \TeX back ends that provide it.		<code>\hyxmp@parse@acmart:</code> Bug fix: Correct a missing “else” argument in two invocations of <code>\@if@def@and@nonempty</code>	36
		<code>\hyxmp@write@pdflang:</code> Added this macro	39
		General: Moved the automatic assignment of <code>\@pdflang</code> and <code>\@pdfmetalang</code> from <code>\hyxmp@auto@assign@data</code> to within a call to <code>\hyxmp@at@end</code>	32

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.

[illegible]

<code>\@pdftitle</code>	<code>\bbl@main@language</code> . 561	182, 184, 186,
. 313, 346, 431,	BOM 79	210, 222, 243, 1061
1158, 1166, 1340		<code>\do</code> . . . 1063, 1070, 1280
<code>\@pdftrapped</code> 1254	C	DOI 2, 6, 24, 36
<code>\@pdftype</code> 54, 1349	CiAdrCity 2	draft (option) 8
<code>\@pdfuapart</code> 78,	CiAdrCtry 2	dvipdf (option) 82
314, 343, 1431, 1827	CiAdrExtadr 2	dvipdfm 39, 83
<code>\@pdfurl</code>	CiAdrPcode 2	dvips (option) 82
159, 315, 1514, 1544	CiAdrRegion 2	dvips 10, 39, 49, 50
<code>\@pdfversionid</code> 121, 1395	CiEmailWork 2	dvipsone (option) 82
<code>\@pdfvolumenum</code> . 151,	CiTelWork 3	dviwindo (option) 82
316, 543, 1507, 1545	<code>\city</code> 472	
<code>\@pdfxstandard</code>	CiUrlWork 3	E
. . 96, 317, 342,	classes	ϵ -TeX 61
1439, 1442, 1444	ACM 15, 32	<code>\EdefEscapeHex</code> 828, 841
<code>\@pdfxversion</code> 1831	acmart 15, 34, 36, 37	<code>\EdefUnescapeHex</code> . . 845
<code>\@publishers</code> 453	Koma 15, 32	<code>\EdefUnescapeString</code> 815
<code>\@subtitle</code> 450	scrlltr2 15	<code>\email</code> 460
<code>\@tempswafalse</code> 1259, 1294	<code>\country</code> 484	<code>\empty</code> 1663
<code>\@tempswatrue</code> . 1259,	CreationDate 31	<code>\equal</code> 92
1261, 1294, 1296		etoolbox (package) . . . 18
<code>\@title</code> 433	D	ETX 40, 47
<code>\^</code> 619, 623, 806, 1186,	Date 45	F
1187, 1852–1854	<code>\day</code> 748, 749, 751	<code>\filemoddate</code> 783
<code>_</code> 1185, 1187	dc:creator 2, 12, 65	
<code>\~</code> 628, 939, 940	dc:date 2, 65	G
	dc:description 3, 12, 55, 65	<code>\getlocaleproperty</code> .
<code>_</code> 905, 940, 1185	dc:format 2 561, 565
A	dc:identifier 3, 64, 65	Ghostsript 10, 11
ACM 15, 32, 36	dc:language 3, 15, 28, 37, 65	gitver (package) 6
acmart (class) 15, 34, 36, 37	dc:publisher 3	
<code>\addresses</code> 500	dc:rights 2, 16, 55, 65	H
<code>\affiliation</code> 496	dc:source 2, 65	<code>\hbox</code> 500
<code>\and</code> 222	dc:subject 2, 65	<code>\Hy@driver</code> 1910, 1914,
ASCII 19, 48	dc:title . . 3, 12, 38, 55, 64	1918, 1922, 1927
<code>\AtBeginDocument</code> . 1231	dc:type 3, 65	<code>\Hy@unicodetrue</code> 33, 44
<code>\AtEndDocument</code> 7	DCMI 8	hyperref (package) 1, 4–
<code>\AtEndDvi</code> 5	<code>\define@key</code> 31,	9, 12, 15, 18–20,
atenddvi (package) . . . 18	42, 53, 55, 57, 59,	25, 26, 28, 30–32,
<code>\AtEndPreamble</code> 266, 355	61, 63, 70, 75,	37–39, 60–62, 80–82
Author 11, 25	79, 98, 116, 118,	<code>\hypersetup</code> . . . 260, 423
	120, 122, 132,	hyperxmp (package) . .
B	134, 136, 138,	. 1, 2, 4–9, 12–
babel (package) . . 15,	140, 142, 144,	21, 24, 26, 28, 30–
28, 30, 32, 37–39	146, 148, 150,	33, 38–41, 47, 48,
<code>\BabelEnsureInfo</code> . . 559	152, 154, 156,	56, 60, 62, 71, 81, 85
<code>baseurl</code> (option)	158, 160, 162,	<code>\hyxmp@is@unicode</code> . 849
4, 6, 14, 18, 24, 67	164, 166, 174,	<code>\hyxmp@acm@isbn</code> . . . 512
	176, 178, 180,	

\hyxmp@acm@publisher	\hyxmp@and 222	\hyxmp@concat@metadata
. 526	\hyxmp@append@hex 265, 276
\hyxmp@acm@pubtype . 535 1106,	\hyxmp@cond@dc@identifier
\hyxmp@add@simple . .	1125–1127, . . 1131	. . 1331, 1363–1366
. . . 1004, 1254,	\hyxmp@append@hex@iii	\hyxmp@construct@packet
1339, 1354, 1368, 1124, 1858, 1908
1385, 1387, 1393–	1130, 1140, 1151	\hyxmp@count@non@spaces
1396, 1400, 1402,	\hyxmp@append@hex@iv 2008, 2019
1406, 1408, 1412, 1129,	\hyxmp@count@spaces
1414, 1416, 1417,	1135, 1136, 2007, 2010
1421, 1422, 1426,	1138, 1153–1155	\hyxmp@crap@convert
1427, 1431, 1438,	\hyxmp@as@pdf@date . 675 931, 965
1439, 1442, 1444,	\hyxmp@as@xmp@date .	\hyxmp@crap@result .
1484–1487, 1501, 36, 47, 921, 957
1505, 1507–1516	647, 785, 1403, 1409	\hyxmp@crap@test 928, 953
\hyxmp@add@simple@lang	\hyxmp@assign@major@minor	\hyxmp@create@uuid .
. 1022, 1236, 1255	. . 1133, 1161, 1170
1503, 1504, 1506	\hyxmp@at@end 3, 370, 383	\hyxmp@cur@lang . . .
\hyxmp@add@simple@lang@i	\hyxmp@auto@assign@data 1063, 1071
. 1025, 1028 361, 393, 430	\hyxmp@dc@lang
\hyxmp@add@simple@lang@ii	\hyxmp@banner@to@producer	. 551, 563, 566,
. 1029, 1031	. . 1212, 1215, 1223	567, 569, 577, 1356
\hyxmp@add@simple@pfx	\hyxmp@begin@ext@decl	\hyxmp@dc@schema . .
. 1045, 1334 1560, 1338, 1890
\hyxmp@add@simple@var	1599, 1610, 1635,	\hyxmp@declare@extensions
. 1013,	1651, 1665, 1820, 1887
1252, 1253, 1256	1681, 1693, 1751	\hyxmp@declare@field
\hyxmp@add@to+xml . .	\hyxmp@begin@extension@decls 1589,
. 1007, 1548, 1821	1717, 1720, 1723,
1009, 1017, 1035,	\hyxmp@big@prime . .	1726, 1729,
1039, 1047, 1051, 1080,	1732, 1735, 1738
1053, 1172, 1198,	1083, 1093, 1103	\hyxmp@declare@property
1266, 1273, 1277,	\hyxmp@big@prime@ii 1577,
1282, 1287, 1299, 1080, 1102	1603, 1614, 1619,
1307, 1313, 1322,	\hyxmp@bom . . 1845, 1860	1624, 1628, 1639,
1457, 1461, 1465,	\hyxmp@check@iptc@data	1644, 1655, 1669,
1471, 1478, 1493, 1518, 1885	1673, 1685, 1697,
1549, 1555, 1561,	\hyxmp@check@prism@data	1755, 1759, 1763,
1571, 1578, 1582, 1530, 1886	1767, 1771, 1775,
1590, 1702,	\hyxmp@check@standards	1779, 1783, 1787,
1741, 1860, 1899 327, 368	1791, 1795, 1799,
\hyxmp@address@val .	\hyxmp@check@std . .	1804, 1809, 1814
. 460, 466, 91, 103–111	\hyxmp@def@DocumentID
472, 478, 484, 490	\hyxmp@comma 1157, 1391
\hyxmp@alt@description	. 167, 223, 244, 618	\hyxmp@def@InstanceID
. 1057, 1067	\hyxmp@commas@to@list 1163, 1392
\hyxmp@alt@rights . .	. 602, 638, 1305, 1469	\hyxmp@define@pdfproducer
. 1057, 1068	\hyxmp@commas@to@list@i 1209, 1233
\hyxmp@alt@title 604, 606	\hyxmp@detect@langs
. 1057, 1066	 359, 554

\hyxmp@DocumentID ..	\hyxmp@iptc@schema ..	\hyxmp@pdf@to@xmp@date
..... 117, 1475, 1897	. 649, 654, 777, 780
1157, 1391, 1393	\hyxmp@is@unicode ..	\hyxmp@pdfa@id@extensions
\hyxmp@dq@code . 1, 2040 817, 834, 849 1634, 1825
\hyxmp@driver 1907	\hyxmp@koma@phones .	\hyxmp@pdfa@id@schema
\hyxmp@embed@packet 395, 444 1424, 1893
..... 372, 1907	\hyxmp@LA@accept ..	\hyxmp@pdfauthor ..
\hyxmp@embed@packet@dvipdfm	.. 1060, 1066–1068	... 213, 222, 1350
..... 1919, 1989	\hyxmp@lang@tag ...	\hyxmp@pdfkeywords .
\hyxmp@embed@packet@luatex 565, 567, 569	... 213, 243, 1351
..... 1915, 1952	\hyxmp@legal 1372	\hyxmp@pdfstringdef
\hyxmp@embed@packet@pdfmark	\hyxmp@list ... 1305, 24, 35,
..... 1931, 1959	1311, 1469, 1470	46, 53, 55, 57, 59,
\hyxmp@embed@packet@pdftex	\hyxmp@list@to@lines	61, 63, 72, 76, 81,
..... 1911, 1939 1454,	99, 116, 118, 120,
\hyxmp@embed@packet@xetex	1483, 1489–1491	122, 132, 134,
..... 1923, 2027	\hyxmp@list@to@xml .	136, 138, 140,
\hyxmp@end@ext@decl 1293,	142, 144, 146,
... 1570, 1607,	1350, 1351, 1356	148, 150, 152,
1632, 1648, 1660,	\hyxmp@major@minor 1236	154, 156, 158,
1677, 1689, 1818	\hyxmp@mm@extensions	160, 162, 164,
\hyxmp@end@extension@decls 1609, 1823	169, 174, 176,
..... 1554, 1843	\hyxmp@mm@schema ..	178, 180, 182,
\hyxmp@extra@indent 1390, 1896	184, 186, 1049, 1062
..... 1003,	\hyxmp@modulo@a ...	\hyxmp@pdfua@id@extensions
1007, 1018,	... 1074, 1093, 1650, 1829
1047, 1458, 1482	1103, 1109, 1144	\hyxmp@pdfua@id@schema
\hyxmp@find@metadata	\hyxmp@new@xml 1188, 1189 1430, 1894
..... 276, 371	\hyxmp@no@bad@parts	\hyxmp@pdfx@id@extensions
\hyxmp@first@char .. 645 64, 71, 80 1662, 1833
\hyxmp@first@char@i	\hyxmp@no@info@lists	\hyxmp@pdfx@id@schema
.... 645, 648, 676 187, 211, 369 1433, 1895
\hyxmp@gobbletwo 715, 728	\hyxmp@num 965	\hyxmp@pdfx@major ..
\hyxmp@hash	\hyxmp@one@token 88, 97, 113,
... 1192, 1864, 1082,	1434, 1663, 1679
1872, 1880–1883	1086, 2011,	\hyxmp@photoshop@data
\hyxmp@Hyp@pdfauthor	2012, 2020, 2021 1419
..... 216	\hyxmp@padding 1196, 1903	\hyxmp@photoshop@schema
\hyxmp@Hyp@pdfkeywords	\hyxmp@parse@acmart 1419, 1891
..... 237 455, 457, 550	\hyxmp@prism@data ..
\hyxmp@hypersetup .. 260	\hyxmp@parse@time 1499, 1531, 1839
\hyxmp@InstanceID 656, 658	\hyxmp@prism@extensions
..... 119,	\hyxmp@parse@tz 1750, 1841
1163, 1392, 1394 665, 668, 672	\hyxmp@prism@schema
\hyxmp@iprefix 1049, 1050	\hyxmp@parse@tz@char 1498, 1898
\hyxmp@iptc@data 660, 662	\hyxmp@ProcessKeyvalOptions
.. 1476, 1519, 1835	\hyxmp@pdf@extensions 255
\hyxmp@iptc@extensions 1598, 1822	\hyxmp@rand@num ...
..... 1692, 1837	\hyxmp@pdf@schema 1099, 1108,
 1251, 1888	1143, 1160, 1169

<code>\hyxmp@rdf@dc</code>	<code>\hyxmp@today@pdf</code> 364, <u>792</u>	<code>\hyxmp@xmllify</code>
.. <u>1258</u> , 1340–1342	<code>\hyxmp@today@xmp</code> 392, <u>813</u> ,
<code>\hyxmp@redefine@Hyp</code> 770, <u>775</u> ,	1006, 1016, 1024,
.... <u>215</u> , 257, 262	793, 1166, 1345,	1033, 1050, 1052,
<code>\hyxmp@remove@this</code> .	1400, 1406, 1412	1264, 1272, 1281,
..... 1227, <u>1230</u>	<code>\hyxmp@today@xmp@define</code>	1304, 1321, 1468
<code>\hyxmp@rights</code> . 1372,	... <u>741</u> , 790, 1164	<code>\hyxmp@xmp@basic@schema</code>
1375, 1379, 1381	<code>\hyxmp@toxml</code> .. 843, <u>866</u> <u>1398</u> , 1892
<code>\hyxmp@seed@rng</code> ...	<code>\hyxmp@toxml@unicodetex</code>	<code>\hyxmp@xmp@to@pdf@date</code>
.. <u>1082</u> , 1159, 1168 831, <u>891</u> 679, 682, 793
<code>\hyxmp@seed@rng@i</code> ..	<code>\hyxmp@trimb</code> .. 799, <u>802</u>	<code>\hyxmp@xmp@to@pdf@date@i</code>
..... 1084, <u>1086</u>	<code>\hyxmp@trimc</code> .. 802, <u>803</u> 683, <u>685</u>
<code>\hyxmp@seed@string</code> .	<code>\hyxmp@trimspaces</code> ..	<code>\hyxmp@xmp@to@pdf@date@i</code>
..... <u>1157</u> , <u>1163</u> 611, <u>795</u> 688, <u>691</u>
<code>\hyxmp@set@dc@lang</code> .	<code>\hyxmp@try</code> <u>921</u>	<code>\hyxmp@xmp@to@pdf@date@iiii</code>
. 384, <u>551</u> , 560, 576	<code>\hyxmp@try@today</code> 769, 694, <u>697</u>
<code>\hyxmp@set@koma@phones</code>	776, 779, 782, 789	<code>\hyxmp@xmp@to@pdf@date@iv</code>
..... 395, 442	<code>\hyxmp@unicodetexfalse</code> 700, <u>703</u>
<code>\hyxmp@set@pdfx@major</code> 805	<code>\hyxmp@xmp@to@pdf@date@v</code>
..... 83, 113	<code>\hyxmp@unicodetextrue</code> 706, <u>709</u>
<code>\hyxmp@set@pdfx@major@i</code> 805	<code>\hyxmp@xmp@to@pdf@date@vi</code>
..... 83, <u>84</u>	<code>\hyxmp@uscore</code> .. 26, <u>622</u> 712, <u>716</u>
<code>\hyxmp@set@pdfx@major@ii</code>	<code>\hyxmp@use@first@valid</code>	<code>\hyxmp@xmp@to@pdf@date@vii</code>
..... 85, <u>88</u> 409, 431, 719, 722, <u>732</u>
<code>\hyxmp@set@rand@num</code>	435, 439, 443,	<code>\hyxmp@xmp@to@pdf@date@viii</code>
.. <u>1099</u> , 1107, 1142	446, 449, 452, 735, <u>738</u>
<code>\hyxmp@singleton@dc</code>	462, 468, 474,	<code>\hyxmp@xmpRights@schema</code>
... <u>1319</u> , 1343,	480, 486, 492, <u>1371</u> , 1889
1345, 1347, 1349	505, 519, 527,	<code>\hyxmp@zero</code> 974,
<code>\hyxmp@skiptorelax</code> .	530, 540, 543, 546	981, 988, 994, <u>999</u>
..... 958, <u>964</u>	<code>\hyxmp@use@first@valid@i</code>	
<code>\hyxmp@skipzeros</code> .. <u>916</u> 411, <u>415</u>	
<code>\hyxmp@SpaceOther</code> ..	<code>\hyxmp@value</code> <u>1062</u> , <u>1265</u>	
..... 925, <u>938</u>	<code>\hyxmp@write@pdf@lang</code>	
<code>\hyxmp@standards</code> .. <u>340</u> 562, <u>581</u>	
<code>\hyxmp@string@len</code> ..	<code>\hyxmp@x@default</code> 387,	
..... 1990, <u>2005</u>	<u>1208</u> , 1270, 1278	
<code>\hyxmp@strip@isbn@date</code>	<code>\hyxmp@xetex@crap</code> ..	
..... 512 822, <u>921</u>	
<code>\hyxmp@sublist</code>	<code>\hyxmp@xml</code> 1008, 1010,	
. 607, 608, 611, 612	1048, 1054, 1189,	
<code>\hyxmp@suppress@pdf@info</code>	<u>1196</u> , 1581, <u>1858</u> ,	
..... 188	1948, 1956, 1979,	
<code>\hyxmp@temp@list</code> .. <u>631</u>	1990, 1997, 2028	
<code>\hyxmp@temp@str</code> ... <u>631</u>	<code>\hyxmp@xmllified</code> 813,	
<code>\hyxmp@text</code>	1009, 1018, 1025,	
. <u>813</u> , <u>891</u> , <u>921</u> , <u>965</u>	1036, 1040, 1051,	
<code>\hyxmp@textunderscore</code>	1053, 1265, 1274,	
..... <u>24</u>	1283, 1305, 1325,	
<code>\hyxmp@timestamp</code> .. <u>782</u>	1332, 1362, 1469	

I

IETF 5
<code>\if@ACM@journal</code>	... 535
<code>\if@temp@swa</code>	. 1263, 1298
ifdraft (package) 22
<code>\ifdraft</code> <u>123</u> , 126
<code>\iffalse</code>	... 1258, 1293
<code>\ifHy@pdfa</code>
	328, 1340, 1341,
	1350, 1425, 1824
<code>\ifhyxmp@unicodetex</code> 805, 816, 1175, 1846
<code>\ifLuaTeX</code> 1211
ifluatex (package) 81
<code>\ifluatex</code>	.. 1941, 1945
ifmtarg (package)	... 18
<code>\ifPDFTeX</code> 1214
<code>\IfSubStr</code>
	. 503, 504, 513, 514

iftex (package) 18
ifthen (package) 18
\ifthenelse 92
\ifXeTeX 821, 1217
Info . 8, 11–13, 25, 31, 60
intcalc (package) 18
\intcalcDiv 970, 977, 984
\intcalcMod 972, 979, 986
IPTC 13,
24, 59, 69–71, 75, 78
lptc4xmpCore:ContactInfo
. 69, 75
lptc4xmpCore:CreatorContactInfo
. 2, 3, 69
ISBN 2, 6, 23, 36
ISO 5–7, 15, 20, 55
ISSN 2, 6, 23

J

\jobname 115,
321, 376, 783,
1158, 1166, 1928

K

keeppdfinfo (option) .
. 8, 13, 25
Keywords 11, 25, 62
Koma (class) 15, 32
\KV@Hyp@pdfauthor . . 222
\KV@Hyp@pdfkeywords 243
kvoptions (package) 18, 28

L

Lang 37–39
LF 68
\LocaleForEach 564
Lua^LTeX 10,
13, 14, 46, 61, 62
LuaTeX 39,
47, 50, 81, 82, 85
\luatexbanner 1212

M

\makeatletter 1227
Metadata 11, 80, 84
\month 743, 744, 746

N

NAK 19, 40, 48
nativepdf (option) . . . 82

\newif 805
\next . 34, 45, 93, 100,
188, 416, 418,
424, 428, 606,
784, 787, 1086, 1831
ngerman (package) . . 17
\number 968, 970, 972,
977, 979, 984, 986
\numexpr 1957

O

options
baseurl
4, 6, 14, 18, 24, 67
draft 8
dvipdf 82
dvips 82
dvipsone 82
dviwindo 82
keeppdfinfo . 8, 13, 25
nativepdf 82
pdfa 8, 20, 30
pdfaformance .
. 4, 8, 68
pdfapart . 4, 8, 20, 68
pdfauthor 4, 5, 12,
14, 15, 18, 26, 27, 65
pdfauthortitle . 4, 5, 14
pdfbookedition . . 4, 7
pdfbytes 4, 8
pdfcaptionwriter . 4, 5
pdfcontactaddress .
. 4, 5, 13
pdfcontactcity . . 4, 5
pdfcontactcountry 4, 5
pdfcontactemail . 4, 5
pdfcontactphone . 4, 5
pdfcontactpostcode
. 4, 5
pdfcontactregion . 4, 5
pdfcontacturl . 4, 5, 14
pdfcopyright 4, 5, 65, 66
pdfcreationdate . .
. 4, 7, 8, 31
pdfdate
4, 7, 15, 19, 58, 65
pdfdocumentid . . 4, 6
pdfdoi 4, 6
pdfeissn 4, 6
pdfidentifier . . 4, 6, 65

pdfinstanceid 4, 6
pdfisbn 4, 6
pdfissn 4, 6
pdfissuenum 4, 7
pdfkeywords . . . 4,
12, 14, 18, 26, 65
pdflang . . 4–7, 15,
18, 32, 37, 38, 55, 65
pdflicenseurl . 5, 14, 61
pdfmark 82
pdfmetadate 5, 7, 8, 19
pdfmetalang 5, 6, 15, 55
pdfmoddate . . 4, 7, 8
pdfnumpages
. 5, 7, 16, 17
pdfpagerange
. 5, 7, 16, 17
pdfproducer . 4, 18, 61
pdfpublication . . 5, 7
pdfpublisher 5, 7
pdfpubtype 5, 7
pdfrendition 5, 8
pdfsource 5, 9
pdfsubject 4, 12, 18, 65
pdfsubtitle 5
pdftitle 4,
5, 12, 15, 18, 33, 65
pdftrapped . . . 4, 8, 18
pdftype 5, 8, 65
pdfuapart . . 5, 8, 20, 68
pdfurl 5, 6
pdfversionid . . 5, 6, 66
pdfvolumenum . . 5, 7
pdfxstandard
. 5, 8, 9, 21, 22, 68
ps2pdf 82
textures 82
unicode 15
vtexpdfmark 82

P

packages
atenddvi 18
babel 15,
28, 30, 32, 37–39
etoolbox 18
gitver 6
hyperref . . . 1, 4–
9, 12, 15, 18–20,

25, 26, 28, 30–32,	pdf:Producer . . . 3, 60, 62	pdfdoi (option) 4, 6
37–39, 60–62, 80–82	pdf:trapped 3	pdfeissn (option) . . . 4, 6
hyperxmp	\PDF@FinishDoc	pdfescape (package) . . 18
. 1, 2, 4–9, 12– 189, 197, 203	\pdfextension 1953, 1957
21, 24, 26, 28, 30–	pdfa (option) . . 8, 20, 30	\pdffeedback . 780, 1957
33, 38–41, 47, 48,	pdfaconformance (op-	pdfidentifier (option) .
56, 60, 62, 71, 81, 85	tion) . . . 4, 8, 68 4, 6, 65
ifdraft 22	pdfaid:conformance . . . 3	pdfinstanceid (option) 4, 6
ifluatex 81	pdfaid:part 3	pdfisbn (option) . . . 4, 6
ifmtarg 18	pdfapart (option)	pdfissn (option) 4, 6
iftex 18 4, 8, 20, 68	pdfissuenum (option) 4, 7
ifthen 18	pdfauthor (option) . . .	pdfkeywords (option) 4,
intcalc 18 4, 5, 12,	12, 14, 18, 26, 65
kvoptions 18, 28	14, 15, 18, 26, 27, 65	pdflang (option) 4–7, 15,
ngerman 17	pdfauthortitle (option)	18, 32, 37, 38, 55, 65
pdfescape 18 4, 5, 14	\pdflastobj 1949
polyglossia . . . 15,	pdfbookedition (option)	pdfL ^A T _E X 4,
28, 30, 32, 37, 38 4, 7	10, 13, 46, 61, 62
stringenc 18	pdfbytes (option) . . . 4, 8	pdflicenseurl (option) .
texdate 16	pdfcaptionwriter (op- 5, 14, 66
totpages 16, 17	tion) 4, 5	\pdfmajorversion . 1244
xmpincl 4	\pdfcatalog . . 587, 1949	pdfmark (option) 82
\PackageWarning . . .	\pdfcompresslevel . 1943	\pdfmark 593,
. 66, 101, 585, 632	pdfcontactaddress (op-	1960, 1963, 1967,
\PackageWarningNoLine	tion) . . . 4, 5, 13	1977, 1981, 1985
. 190, 320,	pdfcontactcity (option)	pdfmetadata (option) .
330, 347, 375, 1926 4, 5 5, 7, 8, 19
\patchcmd 196, 202	pdfcontactcountry (op-	pdfmetalang (option) .
PDF . 1–4, 7, 8, 10–14,	tion) 4, 5 5, 6, 15, 55
16, 19, 25, 29–31,	pdfcontactemail (op-	\pdfminorversion . 1240
37–43, 46, 48, 50,	tion) 4, 5	pdfmoddate (option) .
58–62, 73, 78–81, 84	pdfcontactphone (op- 4, 7, 8
Author 11, 25	tion) 4, 5	pdfnumpages (option)
CreationDate 31	pdfcontactpostcode (op- 5, 7, 16, 17
Info 8, 11–13, 25, 31, 60	tion) 4, 5	\pdfobj 1945
Keywords . . 11, 25, 62	pdfcontactregion (op-	pdfpagerange (option)
Lang 37–39	tion) 4, 5 5, 7, 16, 17
Metadata . . 11, 80, 84	pdfcontacturl (option)	pdfproducer (option) .
Producer 62 4, 5, 14 4, 18, 61
Subject 11	pdfcopyright (option) .	pdfpublication (option)
Title 11 4, 5, 65, 66 5, 7
PDF/A . . 3, 8, 12, 13,	pdfcreationdate (option)	pdfpublisher (option) 5, 7
20, 25, 30, 60, 62, 4, 7, 8, 31	pdfpubtype (option) . 5, 7
67, 68, 71, 74–76, 78	\pdfcreationdate . . 777	pdfrendition (option) 5, 8
PDF/UA 3,	pdfdate (option)	pdfsource (option) . . 5, 9
8, 21, 30, 68, 74, 78	4, 7, 15, 19, 58, 65	\pdfstringdef 27
PDF/X . . 3, 8, 9, 21,	PDFDocEncoding	pdfsubject (option) . .
22, 30, 68, 74, 75, 78 26, 47, 48 4, 12, 18, 65
pdf:Keywords . . . 2, 12, 62	pdfdocumentid (option)	pdfsubtitle (option) . . . 5
pdf:PDFVersion . . . 3, 62 4, 6	pdfT _E X . 8, 39, 49, 81, 85

<code>\pdfTeXbanner</code>	1215		
<code>pdftitle</code> (option)	4,		
	5, 12, 15, 18, 33, 65		
<code>pdftrapped</code> (option)	4, 8, 18		
<code>pdftype</code> (option)	5, 8, 65		
<code>pdfuaid:part</code>	3		
<code>pdfuapart</code> (option)	5, 8, 20, 68		
<code>pdfurl</code> (option)	5, 6		
<code>\pdfvariable</code>	1248		
<code>pdfversionid</code> (option)	5, 6, 66		
<code>pdfvolumenum</code> (option)	5, 7		
<code>pdfxid:GTS_PDFXVersion</code>	3		
<code>pdfxstandard</code> (option)	5, 8, 9, 21, 22, 68		
<code>photoshop:AuthorsPosition</code>	3, 67		
<code>photoshop:CaptionWriter</code>	3, 67		
<code>PI</code>	59		
<code>polyglossia</code> (package)	15,		
	28, 30, 32, 37, 38		
<code>\postcode</code>	<u>490</u>		
<code>PRISM</code>	7, 70, 71, 76, 79		
<code>prism:aggregationType</code>	3		
<code>prism:bookEdition</code>	2		
<code>prism:byteCount</code>	2		
<code>prism:doi</code>	2		
<code>prism:elssn</code>	2		
<code>prism:isbn</code>	2		
<code>prism:issn</code>	2		
<code>prism:number</code>	2		
<code>prism:pageCount</code>	3		
<code>prism:pageRange</code>	3		
<code>prism:publicationName</code>	3		
<code>prism:subtitle</code>	3		
<code>prism:url</code>	3		
<code>prism:volume</code>	3		
<code>\ProcessKeyvalOptions</code>	<u>255</u>		
<code>Producer</code>	62		
<code>properties, XMP</code>			
<code>dc:creator</code>	2, 12, 65		
<code>dc:date</code>	2, 65		
<code>dc:description</code>	3, 12, 55, 65		
<code>dc:format</code>	2		
<code>dc:identifier</code>	3, 64, 65		
<code>dc:language</code>	3, 15, 28, 37, 65		
<code>dc:publisher</code>	3		
<code>dc:rights</code>	2, 16, 55, 65		
<code>dc:source</code>	2, 65		
<code>dc:subject</code>	2, 65		
<code>dc:title</code>	3, 12, 38, 55, 64		
<code>dc:type</code>	3, 65		
<code>lptc4xmpCore:ContactInfo</code>	69, 75		
<code>lptc4xmpCore:CreatorContactInfo</code>	2, 3, 69		
<code>pdf:Keywords</code>	2, 12, 62		
<code>pdf:PDFVersion</code>	3, 62		
<code>pdf:Producer</code>	3, 60, 62		
<code>pdf:trapped</code>	3		
<code>pdfaid:conformance</code>	3		
<code>pdfaid:part</code>	3		
<code>pdfuaid:part</code>	3		
<code>pdfxid:GTS_PDFXVersion</code>	3		
<code>photoshop:AuthorsPosition</code>	3, 67		
<code>photoshop:CaptionWriter</code>	3, 67		
<code>prism:aggregationType</code>	3		
<code>prism:bookEdition</code>	2		
<code>prism:byteCount</code>	2		
<code>prism:doi</code>	2		
<code>prism:elssn</code>	2		
<code>prism:isbn</code>	2		
<code>prism:issn</code>	2		
<code>prism:number</code>	2		
<code>prism:pageCount</code>	3		
<code>prism:pageRange</code>	3		
<code>prism:publicationName</code>	3		
<code>prism:subtitle</code>	3		
<code>prism:url</code>	3		
<code>prism:volume</code>	3		
<code>xmp:BaseURL</code>	2		
<code>xmp:CreateDate</code>	2, 31		
<code>xmp:CreatorTool</code>	3		
<code>xmp:MetadataDate</code>	2		
<code>xmp:ModifyDate</code>	2		
<code>xmpMM:DocumentID</code>	3, 56, 66, 78		
<code>xmpMM:InstanceID</code>	3, 56, 66, 78		
<code>xmpMM:RenditionClass</code>	3		
<code>xmpMM:VersionID</code>	3, 66		
<code>xmpRights:Marked</code>	2, 65		
<code>xmpRights:WebStatement</code>	3, 65		
<code>ps2pdf</code> (option)	82		
		Q	
		<code>\Q</code>	795, 804
		R	
		<code>RDF</code>	63
		<code>rdf:li</code>	2
		<code>rdf:Seq</code>	2
		<code>\renewcommand</code>	256, 560, 576
		<code>\RequirePackage</code>	4, 9–16, 125, 1938
		S	
		<code>\scantokens</code>	1224, 1227
		<code>\scr@fromemail@var</code>	440
		<code>\scr@frommobilephone@var</code>	398, 400
		<code>\scr@fromname@var</code>	436
		<code>\scr@fromphone@var</code>	398, 404
		<code>\scr@fromurl@var</code>	447
		<code>\scr@subject@var</code>	432
		<code>scrLtr2</code> (class)	15
		<code>\SE->pdfdoc@03</code>	<u>811</u>
		<code>\SE->pdfdoc@15</code>	<u>812</u>
		<code>\setbox</code>	500
		<code>\setkeys</code>	1072
		<code>\special</code>	1991,
			1999, 2028, 2034
		<code>\state</code>	<u>478</u>
		<code>\streetaddress</code>	<u>466</u>
		<code>stringenc</code> (package)	18
		<code>\StringEncodingConvert</code>	818,
			824, 835, 838, 933
		<code>Subject</code>	11
		T	
		<code>T_EX</code>	16, 18, 39,
			45–47, 49, 50, 56,
			58, 60, 61, 66, 83–85

