

The hyperxmp package*

Scott Pakin
scott+hyxmp@pakin.org

April 8, 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` v5.1, dated 2020/04/08.

```

    <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 \LaTeX source file (`dc:source`)
- file size in bytes (`prism:byteCount`)
- ISBN (`prism:isbn`)
- ISSN—both print (`prism:issn`) and electronic (`prism:elssn`)
- issue number of parent publication (`prism:number`)
- 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)
- 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`
- `pdfinstanceid`
- `pdfbytes`
- `pdfcontactregion`
- `pdfisbn`
- `pdfcaptionwriter`
- `pdfcontacturl`
- `pdfissn`
- `pdfcontactaddress`
- `pdfcopyright`
- `pdfissuenum`
- `pdfcontactcity`
- `pdfdate`
- `pdflicenseurl`

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

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 13 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

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.

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 `uuid:xxxxxxxx-xxx-xxx-xxx-xxxxxxxxxxxx`, where “x” is a lowercase hexadecimal number. For example, `uuid:53ab7f19-a48c-5177-8bb2-403ad907f632` is a valid argument to `pdfdocumentid` (or `pdfinstanceid`). See Leach, Mealling, and Salz’s UUID specification document for details on how to produce the various forms of UUIDs [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 5.1 to match the package version); or labeled using any other approach. One possibility is to use a revision number or commit hash from the version-control software maintaining the document. For example, the `\gitVer` macro from the `gitver` package is an expandable (see Note 8 on page 15) version of the current Git hash that can suitably be passed to `pdfversionid`. If not specified, `pdfversionid` defaults to 1.

Already-published documents can be identified in a number of ways. `pdfisbn` specifies the ISBN. `pdfissn` refers to the ISSN of the *print* version of the document. `pdfissn` refers to the ISSN of the *electronic* version of the document. `pdfdoi` specifies the DOI and should include only the DOI name without any URL prefix. For example, specify `pdfdoi={10.1145/3149526.3149532}`, *not* `pdfdoi={https://doi.org/10.1145/3149526.3149532}`. `pdfurl` points to the complete URL for the document. In contrast, `baseurl` points one level up and is used to resolve relative URLs.

Already-published documents can further be identified by the publication in which they appear. `pdfpublication` specifies the title of the journal, magazine, or other parent document. The title language is assumed to be the same 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. For example, `pdfpublication={[fr]Charlie Hedbo}` indicates a French-language title. Were the language or pronunciation differences significant, `fr-FR` would indicate specifically the French spoken in France, as opposed to that spoken in, say, Canada (`fr-CA`) or Belgium (`fr-BE`). The publisher itself can be named using `pdfpublisher`.

`pdfpubtype` indicates the type of publication in which the document was published. This should be one of the PRISM aggregation types [8] such as `book`, `journal`, `magazine`, `manual`, `report`, or `whitepaper`. For publications in journals, magazines, and similar periodicals, a document can specify the volume number

pdfvolumenum	with pdfvolumenum and the issue number within the volume with pdfissuenum.
pdfissuenum	pdfpagerange indicates the page numbers at which the document appears within the publication. The intention is that this be a comma-separated list of dash-separated ranges, as in pdfpagerange={1,4-5}. See Note 9 on page 16 for advice
pdfpagerange	on how to assign pdfpagerange semi-automatically. For books, pdfbookedition names the edition of the book. This is specified as text, not a number. As with pdfpublication (above), pdfbookedition accepts a bracketed language code, as in pdfbookedition={[en]Second edition}.
pdfbookedition	The number of pages in the published, print version of the document can be expressed with pdfnumpages. Note 9 on page 16 explains how to automatically assign a value to pdfnumpages.
pdfnumpages	XMP metadata can include a number of dates (in fact, timestamps, as they include both date and time components). pdfdate specifies the document date. It is analogous to the L ^A T _E X \date command, and, like \date, 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 YYYY-MM-DDThh:mm:ss+TT:tt. ¹ 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 2014-09-23T14:15:09-06:00. This can be truncated (with loss of information) to 2014-09-23T14:15:09, 2014-09-23T14:15, 2014-09-23, 2014-09, or 2014 but no other subsets. PDF dates are written in the form D:YYYYMMDDhhmmss+TT'tt'. The same date in the preceding example would be written as D:20140923141509-06'00' in PDF format.
pdfdate	The document's creation date, modification date, and metadata date are normally set automatically, but pdfcreationdate, pdfmoddate, and pdfmetadate can be used to override the defaults. Like pdfdate, pdfmetadate can be specified in either XMP or PDF format. However, because hyperref defines pdfcreationdate and pdfmoddate and expects these to be written as PDF dates, hyperxmp concomitantly accepts these two dates only in PDF format as well. Note that it's rare that a document would need to specify any of pdfcreationdate, pdfmoddate, or pdfmetadate.
pdfcreationdate	pdftype describes the type of document being produced. This refers to "the nature or genre of the resource" [4] such as poem, novel or working paper, as opposed to the file format (always application/pdf when generated by hyperxmp). Although pdftype can be assigned an arbitrary piece of text, the XMP specification recommends selecting types from a "controlled vocabulary" such as the DCMI Type Vocabulary [5]. The DCMI Type Vocabulary currently consists of only Collection, Dataset, Event, Image, InteractiveResource, MovingImage, PhysicalObject, Service, Software, Sound, StillImage, and Text. pdftype defaults to Text, which refers to "books, letters, dissertations, poems, newspapers, articles, archives of mailing lists," [5] and other forms of text—all things L ^A T _E X is commonly used to typeset.
pdfmoddate	
pdfmetadate	
pdftype	
pdfrendition	Sometimes a base document is rendered in different forms. pdfrendition indi-

¹Although allowed by XMP, hyperxmp does not currently accept fractions of a second in timestamps.

cates the particular rendition the current document instance represents. The value should come from the following controlled vocabulary [4]: **default**, **draft**, **low-res**, **proof**, **screen**, and **thumbnail**. `hyperxmp`'s default value is **default**, which indicates the master document, unless the **draft** option is passed to `\documentclass`, in which case `hyperxmp` defaults to **draft**.

<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 pdfTeX's <code>\pdffilesize</code> primitive: " <code>pdfbytes={\pdffilesize{\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 False is provided, a document will additionally need to specify <code>keeppdfinfo</code> (page 12) 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>pdfpart</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.

<code>pdfsource</code>	A rarely needed option, <code>pdfsource</code> , overrides the name of the L ^A T _E X source file. It defaults to <code>\jobname.tex</code> but can be replaced by any other string. If <code>pdfsource</code> 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:6dlac9ec-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.p
pdfdoi={10.1002/andp.19053220607},
pdfbytes={\pdffilesize{\jobname.pdf}} % Requires pdflatex
}
\XMPLangAlt{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 + Dvipsdfm
- \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.

2.3 Usage notes

Note 1: Conflicting metadata in PDF/A documents A PDF file includes an Info dictionary containing Author, Title, Subject, and Keywords keys. The hyperref package's pdfauthor, pdftitle, pdfsubject, and pdfkeywords options assign values to those keys. The hyperxmp package additionally uses those options to assign values to various XMP metadata: dc:creator, dc:title, dc:description, and

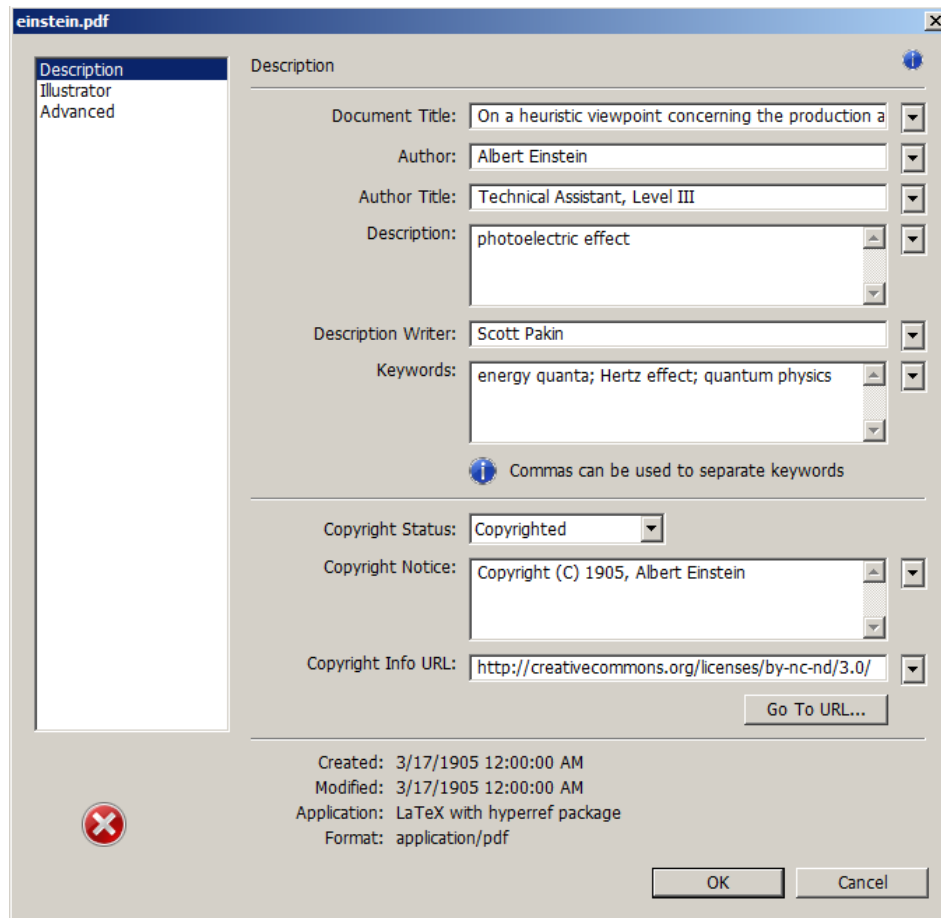


Figure 1: XMP metadata as it appears in Adobe Acrobat

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,

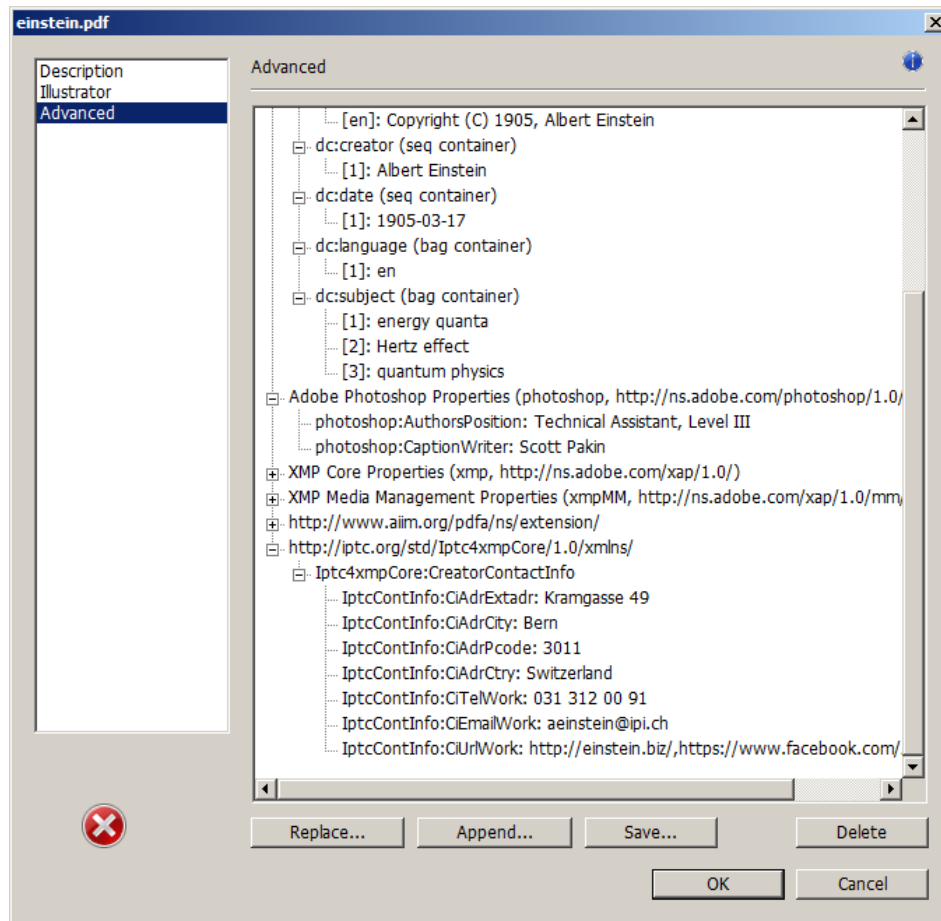


Figure 2: Additional XMP metadata as it appears in Adobe Acrobat

Larry Fine, Moe Howard” or “Curly Howard; Larry Fine; Moe Howard” or something else. I have not yet found a form of flat string that passes all PDF/A validators. Furthermore, when Adobe Acrobat—at least Adobe Acrobat DC (2019) and earlier versions—converts a PDF file to PDF/A format, it does so by discarding all but the first author, which is an unsatisfying solution.

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

`keeppdfinfo`

`\xmplinesep`

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. X \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 X \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}`

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

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

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

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

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

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

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

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

Note 6: Automatically specified metadata `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. An implication of automatic metadata specification is that an author can simply include `\usepackage{hyperxmp}` in a document’s preamble and benefit from a modicum of XMP metadata with no additional effort.

`\XMPLangAlt`

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

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

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

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

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

By way of example, the macros provided by the `texdate` package for typesetting dates are not expandable (at least at the time of this writing). Hence, the `\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\texdyr 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 \LaTeX to do that on your behalf:

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

3 Implementation

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

3.1 Initial preparation

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

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

`\hyxmp@at@end` `\hyxmp@driver` The `\hyxmp@at@end` macro includes code at the end of the document. For `pdf \TeX` , the standard `\AtEndDocument` works well enough. For all the other backends we use `\AtEndDvi` from the `atenddvi` package, which is more robust but requires an additional \LaTeX run.

```
3 \def\hyxmp@driver{hpdf $\text{\TeX}$ }
4 \ifx\hyxmp@driver\Hy@driver
5   \let\hyxmp@at@end=\AtEndDocument
6 \else
7   \RequirePackage{atenddvi}
8   \let\hyxmp@at@end=\AtEndDvi
9 \fi
```

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 L^AT_EX’s document-metadata naming conventions), we do not prefix metadata-related macro names with our package-specific `\hyxmp@` prefix. That is, we use names like `\@pdfcopyright` instead of `\hyxmp@pdfcopyright`.

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

```
10 \RequirePackage{koptions}
11 \RequirePackage{pdfescape}
12 \RequirePackage{stringenc}
13 \RequirePackage{intcalc}
14 \RequirePackage{iftex}
15 \RequirePackage{ifmtarg}
16 \RequirePackage{etoolbox}
17 \RequirePackage{ifthen}
```

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

```
18 \def\@ifmtargexp#1{\expandafter\@ifmtarg\expandafter{#1}}
19 \def\@ifnotmtargexp#1{\expandafter\@ifnotmtarg\expandafter{#1}}
```

`\hyxmp@pdfstringdef` `\hyxmp@textunderscore` 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.

```
20 \newcommand{\hyxmp@pdfstringdef}[2]{%
21   \let\hyxmp@textunderscore=\textunderscore
22   \let\textunderscore=\hyxmp@uscore
23   \pdfstringdef{#1}{#2}%
24   \let\textunderscore=\hyxmp@textunderscore
25 }
```

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

```
26 \def\@pdfdatetime{}
27 \define@key{Hyp}{pdfdate}{%
28   \begingroup
29     \Hy@unicodfalse
```

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

```

30     \edef\next{%
31         \noexpand\hyxmp@pdfstringdef\noexpand\@pdfdatetime{%
32             \noexpand\hyxmp@as@xmp@date{#1}}%
33     }%
34     \next
35 \endgroup
36 }

\@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 3.3.2) we always
                  store \@pdfmetadatetime as an XMP-format string.
37 \def\@pdfmetadatetime{
38 \define@key{Hyp}{pdfmetadate}{%
39     \begingroup
40         \Hy@unicodetofalse

\@pdfnext Expand pdfmetadate's argument and convert it to XMP format.
41     \edef\next{%
42         \noexpand\hyxmp@pdfstringdef\noexpand\@pdfmetadatetime{%
43             \noexpand\hyxmp@as@xmp@date{#1}}%
44     }%
45     \next
46 \endgroup
47 }

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

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

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

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

\@pdfcaptionwriter Prepare to store the name of the person who inserted the hyperxmp metadata.
56 \def\@pdfcaptionwriter{
57 \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.
58 \def\@pdfmetalang{
59 \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.

```

60 \def\hyxmp@no@bad@parts#1\relax{%
61   \@ifnotmtarg{#1}{%
62     \PackageWarning{hyperxmp}{pdfapart and pdfuapart must be numeric}%
63   }%
64 }

```

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

```

65 \def\@pdfapart{}
66 \define@key{Hyp}{pdfapart}{%
67   \afterassignment\hyxmp@no@bad@parts\@tempcnta=0#1\relax
68   \hyxmp@pdfstringdef\@pdfapart{\the\@tempcnta}%
69 }

```

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

```

70 \def\@pdfaconformance{}
71 \define@key{Hyp}{pdfaconformance}{%
72   \uppercase{\hyxmp@pdfstringdef\@pdfaconformance{#1}}%
73 }

```

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

```

74 \def\@pdfuapart{}
75 \define@key{Hyp}{pdfuapart}{%
76   \afterassignment\hyxmp@no@bad@parts\@tempcnta=0#1\relax
77   \hyxmp@pdfstringdef\@pdfuapart{\the\@tempcnta}%
78 }

```

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

```

79 \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.

```

80 \def\hyxmp@set@pdfx@major@i PDF/X-{%
81   \afterassignment\hyxmp@set@pdfx@major@ii
82   \@tempcnta=%
83 }

```

`\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.

```

84 \def\hyxmp@set@pdfx@major@ii#1!{%
85   \edef\hyxmp@pdfx@major{\the\@tempcnta}%
86 }

```

`\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.

```

87 \newcommand*\hyxmp@check@std[2]{%
88   \ifthenelse{\equal{#1}{#2}}{%
89     {\global\let\next=\relax}%
90     {}%
91 }%
```

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

```

92 \def\@pdfxstandard{}
93 \def\hyxmp@pdfx@major{}
94 \define@key{Hyp}{pdfxstandard}{%
95   \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 58 above the definition of `\hyxmp@pdfx@id@schema`, for example.)

```

96 \gdef\next{%
97   \PackageWarning{hyperxmp}{Unrecognized PDF/X standard ‘#1’}%
98 }%
99 \hyxmp@check@std{#1}{PDF/X-1a:2001}%
100 \hyxmp@check@std{#1}{PDF/X-1a:2003}%
101 \hyxmp@check@std{#1}{PDF/X-3:2002}%
102 \hyxmp@check@std{#1}{PDF/X-3:2003}%
103 \hyxmp@check@std{#1}{PDF/X-4}%
104 \hyxmp@check@std{#1}{PDF/X-4p}%
105 \hyxmp@check@std{#1}{PDF/X-5g}%
106 \hyxmp@check@std{#1}{PDF/X-5n}%
107 \hyxmp@check@std{#1}{PDF/X-5pg}%
108 \next
```

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

```

109 \hyxmp@set@pdfx@major{#1}%
110 }
```

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

```

111 \edef\@pdfsource{\jobname.tex}
112 \define@key{Hyp}{pdfsource}{\hyxmp@pdfstringdef\@pdfsource{#1}}
```

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

```

113 \def\hyxmp@DocumentID{}
114 \define@key{Hyp}{pdfdocumentid}{\hyxmp@pdfstringdef\hyxmp@DocumentID{#1}}
```

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

```

115 \def\hyxmp@InstanceID{}
116 \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”.

```

117 \def\@pdfversionid{1}
118 \define@key{Hyp}{pdfversionid}{\hyxmp@pdfstringdef\@pdfversionid{#1}}

119 \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`.

```

120 \ifdraft{%
121   \def\@pdfrendition{draft}%
122 }{%
123   \def\@pdfrendition{default}%
124 }
125 \define@key{Hyp}{pdfrendition}{\hyxmp@pdfstringdef\@pdfrendition{#1}}

```

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

```

126 \def\@pdfpublication{}
127 \define@key{Hyp}{pdfpublication}{\hyxmp@pdfstringdef\@pdfpublication{#1}}

```

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

```

128 \def\@pdfpubtype{}
129 \define@key{Hyp}{pdfpubtype}{\hyxmp@pdfstringdef\@pdfpubtype{#1}}

```

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

```

130 \def\@pdfbytes{}
131 \define@key{Hyp}{pdfbytes}{\hyxmp@pdfstringdef\@pdfbytes{#1}}

```

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

```

132 \def\@pdfnumpages{}
133 \define@key{Hyp}{pdfnumpages}{\hyxmp@pdfstringdef\@pdfnumpages{#1}}

```

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

```

134 \def\@pdfissn{}
135 \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.

```

136 \def\@pdfeissn{}
137 \define@key{Hyp}{pdfeissn}{\hyxmp@pdfstringdef\@pdfeissn{#1}}

```

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

```

138 \def\@pdfisbn{}
139 \define@key{Hyp}{pdfisbn}{\hymp@pdfstringdef\@pdfisbn{#1}}

```

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

```

140 \def\@pdfbookedition{}
141 \define@key{Hyp}{pdfbookedition}{\hymp@pdfstringdef\@pdfbookedition{#1}}

```

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

```

142 \def\@pdfpublisher{}
143 \define@key{Hyp}{pdfpublisher}{\hymp@pdfstringdef\@pdfpublisher{#1}}

```

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

```

144 \def\@pdfvolumenum{}
145 \define@key{Hyp}{pdfvolumenum}{\hymp@pdfstringdef\@pdfvolumenum{#1}}

```

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

```

146 \def\@pdfissuenum{}
147 \define@key{Hyp}{pdfissuenum}{\hymp@pdfstringdef\@pdfissuenum{#1}}

```

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

```

148 \def\@pdfpagerange{}
149 \define@key{Hyp}{pdfpagerange}{\hymp@pdfstringdef\@pdfpagerange{#1}}

```

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

```

150 \def\@pdfdoi{}
151 \define@key{Hyp}{pdfdoi}{\hymp@pdfstringdef\@pdfdoi{#1}}

```

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

```

152 \def\@pdfurl{}
153 \define@key{Hyp}{pdfurl}{\hymp@pdfstringdef\@pdfurl{#1}}

```

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

```

154 \def\@pdfsubtitle{}
155 \define@key{Hyp}{pdfsubtitle}{\hymp@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.

```

156 \def\@pdfcontactaddress{}
157 \define@key{Hyp}{pdfcontactaddress}{%
158   \let\xmpcomma=\hyxmp@comma
159   \def\xmpquote##1{##1}%
160   \hyxmp@pdfstringdef\@pdfcontactaddress{#1}%
161   \def\xmpcomma{,}%
162   \let\xmpquote=\relax
163 }

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

170 \def\@pdfcontactcountry{}
171 \define@key{Hyp}{pdfcontactcountry}{\hyxmp@pdfstringdef\@pdfcontactcountry{#1}}

```

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

```

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

```

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

```

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

```

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

```

176 \def\@pdfcontacturl{}
177 \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`.

```

178 \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.

```

179 \def\hyxmp@suppress@pdf@info{%
180   \global\let\PDF@FinishDoc=\@empty
181   \PackageWarningNoLine{hyperxmp}{%
182     Suppressing the _entire_ PDF Info dictionary.\MessageBreak
183     Please notify the hyperxmp maintainer%
184   }%
185 }%
186 \let\next=\relax
187 \patchcmd
188   {\PDF@FinishDoc}%
189   {/Author(\@pdfauthor)}%
190   {}%
191   {}%
192   {\let\next=\hyxmp@suppress@pdf@info}%
193 \patchcmd
194   {\PDF@FinishDoc}%
195   {/Keywords(\@pdfkeywords)}%
196   {}%
197   {}%
198   {\let\next=\hyxmp@suppress@pdf@info}%
199 \next
200 }

201 \define@key{Hyp}{\keeppdfinfo}[true]{%
202   \gdef\hyxmp@no@info@lists{}%
203 }

```

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 204 \def\hyxmp@pdfauthor{}
205 \def\hyxmp@pdfkeywords{}

```

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

```

206 \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.

```

207 \ifundefined{KV@Hyp@pdfauthor}{}%
208 \ifundefined{hyxmp@Hyp@pdfauthor}{%
209 \expandafter\let\expandafter\hyxmp@Hyp@pdfauthor
210 \csname KV@Hyp@pdfauthor\endcsname
211 }{}%
212 }%

```

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

```

213 \define@key{Hyp}{pdfauthor}{%
214 \let\xmpcomma=\hyxmp@comma
215 \def\xmpquote####1{####1}%
216 \let\hyxmp@and=\and
217 \def\and{,}%
218 \hyxmp@Hyp@pdfauthor{##1}%
219 \global\let\hyxmp@pdfauthor=\@pdfauthor
220 \def\and{and\space}%
221 \def\xmpcomma{,}%
222 \def\xmpquote####1{"####1"%
223 \hyxmp@Hyp@pdfauthor{##1}%
224 \def\xmpcomma{,}%
225 \let\xmpquote=\relax
226 \let\and=\hyxmp@and
227 }%

```

`\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`.

```

228 \ifundefined{KV@Hyp@pdfkeywords}{}%
229 \ifundefined{hyxmp@Hyp@pdfkeywords}{%
230 \expandafter\let\expandafter\hyxmp@Hyp@pdfkeywords
231 \csname KV@Hyp@pdfkeywords\endcsname
232 }{}%

```

233 }%

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

```
234 \define@key{Hyp}{pdfkeywords}{%
235   \let\xmpcomma=\hyxmp@comma
236   \def\xmpquote####1{####1}%
237   \hyxmp@Hyp@pdfkeywords{##1}%
238   \global\let\hyxmp@pdfkeywords=\@pdfkeywords
239   \def\xmpcomma{,%}
240   \def\xmpquote####1{"####1"%}
241   \hyxmp@Hyp@pdfkeywords{##1}%
242   \def\xmpcomma{,%}
243   \let\xmpquote=\relax
244 }%
245 }
```

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

```
246 \let\hyxmp@ProcessKeyvalOptions=\ProcessKeyvalOptions
247 \renewcommand*{\ProcessKeyvalOptions}{%
248   \hyxmp@redefine@Hyp
249   \hyxmp@ProcessKeyvalOptions
250 }
```

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

```
251 \let\hyxmp@hypersetup=\hypersetup
252 \def\hypersetup{%
253   \hyxmp@redefine@Hyp
254   \hyxmp@hypersetup
255 }
```

`\hyxmp@find@metadata` Issue a warning message if the author failed to specify any metadata at all. This excludes metadata that is included automatically such as the current timestamp. `\hyxmp@concat@metadata` 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.

```
256 \newcommand*{\hyxmp@find@metadata}{%
257   \edef\hyxmp@concat@metadata{%
258     \@baseurl
259     \@pdfauthor
```

```

260 \pdfauthortitle
261 \pdfbookedition
262 \pdfbytes
263 \pdfcaptionwriter
264 \pdfcontactaddress
265 \pdfcontactcity
266 \pdfcontactcountry
267 \pdfcontactemail
268 \pdfcontactphone
269 \pdfcontactpostcode
270 \pdfcontactregion
271 \pdfcontacturl
272 \pdfcopyright
273 \pdfcreationdate
274 \pdfdatetime
275 \pdfdoi
276 \pdfeissn
277 \pdfisbn
278 \pdfissn
279 \pdfissuenum
280 \pdfkeywords
281 \pdflang
282 \pdflicenseurl
283 \pdfmetadatetitle
284 \pdfmoddate
285 \pdfnumpages
286 \pdfpagerange
287 \pdfpublication
288 \pdfpubtype
289 \pdfsubject
290 \pdfsubtitle
291 \pdftitle
292 \pdfuapart
293 \pdfurl
294 \pdfvolumenum
295 \pdfxstandard
296 }%
297 \ifx\hyxmp@concat@metadata\@empty
298 \PackageWarningNoLine{hyperxmp}{%
299 \jobname.tex did not specify any metadata to\MessageBreak
300 include in the XMP packet.\space\space Please see the\MessageBreak
301 hyperxmp documentation for instructions on how to\MessageBreak
302 provide metadata values to hyperxmp}%
303 \fi
304 }

```

`\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.

```

305 \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.

```

306 \ifHy@pdfa
307   \ifmtargexp{\pdfapart}{%
308     \PackageWarningNoLine{hyperxmp}{%
309       'pdfa' was passed to hyperref, but 'pdfapart' was\MessageBreak
310       not specified.\space\space Setting pdfapart to '1' and\MessageBreak
311       pdfaconformance to 'B'%
312     }%
313     \gdef\pdfapart{1}%
314     \gdef\pdfaconformance{B}%
315   }%
316 \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).

```

317 \edef\hyxmp@standards{%
318   \pdfapart
319   \pdfxstandard
320   \pdfuapart
321 }%

```

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

```

322 \ifnotmtargexp{\hyxmp@standards}{%
323   \ifmtargexp{\pdftitle}{%
324     \PackageWarningNoLine{hyperxmp}{%
325       Missing pdftitle (required for PDF standards\MessageBreak
326       compliance)%
327     }%
328   }%
329 }%
330 }

```

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.

```

331 \AtBeginDocument{%
332   \ifpackageloaded{hyperref}{%

```

In older versions of `hyperref`, `\pdflang` is set to `\@empty` if `pdflang` is not specified. In newer versions of `hyperref`, `\pdflang` is set to `\relax` if `pdflang` is not specified. The latter is a bit problematic for `hyperxmp` because it makes `\pdflang` non-expandable, which causes a literal “`\@pdflang`” to be written as XMP metadata. To avoid that situation we redefine `\pdflang` as `\@empty` if we see it set to `\relax`.

```

333   \ifx\pdflang\relax
334     \let\pdflang=\@empty
335   \fi

```

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

```

336 \ifx\pdfmetalang\empty
337 \ifx\pdflang\empty
338 \let\pdfmetalang=\hyxmp@x@default
339 \else
340 \edef\pdfmetalang{\pdflang}%
341 \fi
342 \fi
343 \hyxmp@xmllify\pdfmetalang

```

If the author left `pdftitle` blank but specified `\title`, use the title for `pdftitle`. Likewise, if the author left `pdfauthor` blank but specified `\author`, use the author for `pdfauthor`.

```

344 \@ifmtargexp{\pdftitle}{%
345 \ifnotmtargexp{\@title}{%
346 \hypersetup{pdftitle={\@title}}}%
347 }%
348 }%
349 {}%
350 \@ifmtargexp{\pdfauthor}{%
351 \ifnotmtargexp{\@author}{%
352 \hypersetup{pdfauthor={\@author}}}%
353 }%
354 }%
355 {}%

```

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.

```

356 \@ifundefined{XeTeXversion}{-}{%
357 \@ifmtargexp{\pdfcreationdate}{%
358 \let\pdfcreationdate=\hyxmp@today@pdf
359 }%
360 {}%
361 }%

```

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

```

362 \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.

```
363 \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`.

```
364 \hyxmp@at@end{%
365   \hyxmp@find@metadata
366   \hyxmp@embed@packet
367 }%
368 }{%
369   \PackageWarningNoLine{hyperxmp}{%
370     \jobname.tex failed to include a\MessageBreak
371     \string\usepackage\string{hyperref\string}
372     in the preamble.\MessageBreak
373     Consequently, all hyperxmp functionality will be\MessageBreak
374     disabled}%
375 }%
376 }
```

3.3 Manipulating author-supplied data

The author provides metadata information to `hyperxmp` via package options to `hyperref` or via `hyperref`'s `\hypersetup` command. The functions in this section convert author-supplied lists (e.g., `pdfkeywords={foo, bar, baz}`) into `LATEX` lists (e.g., `\@elt {foo} \@elt {bar} \@elt {baz}`) that can be more easily manipulated (Section 3.3.1); parse dates in both PDF and XMP formats (Section 3.3.2; trim spaces off the ends of strings (Section 3.3.3); convert text to XML (e.g., from `<scott+hyxmp@pakin.org>` to `<scott+hyxmp@pakin.org>`) (Section 3.3.4); simplify the pretty-printing of a begin tag, XML text, and end tag (Section 3.3.5); and provide metadata in multiple languages (Section 3.3.6).

3.3.1 List manipulation

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

```
\hyxmp@commas@to@list
```

Given a macro name (#1) and a comma-separated list (#2), define the macro name as the elements of the list, each preceded by `\@elt`. (Executing the macro therefore applies `\@elt` to each element in turn.)

```
377 \newcommand*{\hyxmp@commas@to@list}[2]{%
378   \gdef#1{%
379     \expandafter\hyxmp@commas@to@list@i\expandafter#1#2,,%
380 }
```

```
\hyxmp@commas@to@list@i
```

Recursively construct macro #1 from comma-separated list #2. Stop if #2 is empty.

```
\next
```

```

381 \def\hyxmp@commas@to@list@i#1#2,{%
382   \gdef\hyxmp@sublist{#2}%
383   \ifx\hyxmp@sublist\@empty
384     \let\next=\relax
385   \else
386     \hyxmp@trimspaces\hyxmp@sublist
387     \@cons{#1}{\hyxmp@sublist}}%
388   \def\next{\hyxmp@commas@to@list@i{#1}}%
389   \fi
390   \next
391 }

```

`\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.

```

392 \def\xmpcomma{,}%

```

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

```

393 \bgroup
394   \catcode'\^^C=11
395   \gdef\hyxmp@comma{^^C}
396 \egroup

```

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

```

397 \bgroup
398   \catcode'\^^U=11
399   \gdef\hyxmp@uscore{^^U}
400 \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.

```

401 \let\xmpquote=\relax

```

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

```

402 \bgroup

```



```

403 \catcode'\~ =12%
404 \gdef\xmptilde{~}%
405 \egroup

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

406 \newcommand{\XMPTruncateList}[1]{%
407 \PackageWarning{hyperxmp}{%
408 \noexpand\XMPTruncateList has been deprecated since\MessageBreak
409 hyperxmp 4.0 and may be removed in future\MessageBreak
410 versions of the package. \noexpand\XMPTruncateList\MessageBreak
411 was found}%
412 \edef\hyxmp@temp@str{\csname hyxmp@#1\endcsname}%
413 \hyxmp@commas@to@list{\hyxmp@temp@list}{\hyxmp@temp@str}%
414 \def\@elt##1{%
415 \expandafter\gdef\csname @#1\endcsname{##1}%
416 \let\@elt=\@gobble
417 }
418 \hyxmp@temp@list
419 }}

```

3.3.2 Date manipulation

hyperxmp needs to manipulate two types of date (really, timestamp) formats: PDF format and XMP format. PDF timestamps are of the form “D:YYYYMMDDhhmmss+TT’tt” (e.g., D:20200408091001-06’00’) [3], while XMP timestamps are of the form “YYYY-MM-DDThh:mm:ss+TT:tt” (e.g., 2020-04-08T09:10:01-06:00) [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
420 \def\hyxmp@first@char#1{\hyxmp@first@char@i#1\relax}
421 \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.

422 \def\hyxmp@as@xmp@date#1{%
423 \expandafter\ifnum\expandafter'\hyxmp@first@char@i#1\relax='D
424 \hyxmp@pdf@to@xmp@date{#1}%
425 \else
426 #1%
427 \fi
428 }

```

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

```

429 \def\hyxmp@pdf@to@xmp@date#1:#2#3#4#5#6#7#8#9{%
430   #2#3#4#5-#6#7-#8#9%
431   \hyxmp@parse@time
432 }

```

`\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`.

```

433 \def\hyxmp@parse@time#1#2#3#4#5#6{%
434   T#1#2:#3#4:#5#6%
435   \hyxmp@parse@tz@char
436 }

```

`\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.

```

437 \def\hyxmp@parse@tz@char#1{%
438   #1%
439   \ifx#1-%
440     \expandafter\hyxmp@parse@tz
441   \else
442     \ifx#1+%
443       \expandafter\hyxmp@parse@tz
444     \fi
445   \fi
446 }

```

`\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.

```

447 \def\hyxmp@parse@tz#1'#2' {%
448   #1:#2%
449 }

```

`\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.

```

450 \def\hyxmp@as@pdf@date#1{%
451   \expandafter\ifx\hyxmp@first@char@i#1\relax D%
452   #1%
453   \else

```

```

454 \hyxmp@xmp@to@pdf@date{#1}%
455 \fi
456 }

```

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

```

457 \def\hyxmp@xmp@to@pdf@date#1{%
458 D:\hyxmp@xmp@to@pdf@date@i#1\relax\relax
459 }

```

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

```

460 \def\hyxmp@xmp@to@pdf@date@i#1#2#3#4#5#6{%
461 #1#2#3#4%
462 \ifx#5-%
463 \expandafter\hyxmp@xmp@to@pdf@date@ii\expandafter#6%
464 \fi
465 }

```

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

```

466 \def\hyxmp@xmp@to@pdf@date@ii#1#2#3#4{%
467 #1#2%
468 \ifx#3-%
469 \expandafter\hyxmp@xmp@to@pdf@date@iii\expandafter#4%
470 \fi
471 }

```

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

```

472 \def\hyxmp@xmp@to@pdf@date@iii#1#2#3#4{%
473 #1#2%
474 \ifx#3T%
475 \expandafter\hyxmp@xmp@to@pdf@date@iv\expandafter#4%
476 \fi
477 }

```

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

```

478 \def\hyxmp@xmp@to@pdf@date@iv#1#2#3#4{%
479 #1#2%
480 \ifx#3:%
481 \expandafter\hyxmp@xmp@to@pdf@date@v\expandafter#4%
482 \fi
483 }

```

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

```

484 \def\hyxmp@xmp@to@pdf@date@v#1#2#3#4{%
485 #1#2%
486 \ifx#3:%
487 \expandafter\hyxmp@xmp@to@pdf@date@vi\expandafter#4%
488 \fi
489 }

```

`\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.

```

490 \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).

```

491 \def\hyxmp@xmp@to@pdf@date@vi#1#2#3#4{%
492   #1#2%
493   \ifx#3+%
494     +\expandafter\hyxmp@xmp@to@pdf@date@vii
495   \fi
496   \ifx#3-%
497     -\expandafter\hyxmp@xmp@to@pdf@date@vii
498   \fi
499   \ifx#3Z%
500     Z%
501   \fi
502   \ifx#3\relax
503     \expandafter\hyxmp@gobbletwo
504   \fi
505   \@gobbletwo #4%
506 }

```

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

```

507 \def\hyxmp@xmp@to@pdf@date@vii#1\@gobbletwo#2#3#4#5{%
508   #2#3%
509   \ifx#4:%
510     \expandafter\hyxmp@xmp@to@pdf@date@viii\expandafter#5%
511   \fi
512 }

```

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

```

513 \def\hyxmp@xmp@to@pdf@date@viii#1#2#3#4{%
514   '#1#2'%
515 }

```

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

```

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

```

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

```

517   \xdef#1{\the\year}%
518   \ifnum\month<10
519     \xdef#1{#1-0\the\month}%
520   \else

```

```

521 \xdef#1{#1-\the\month}%
522 \fi
523 \ifnum\day<10
524 \xdef#1{#1-0\the\day}%
525 \else
526 \xdef#1{#1-\the\day}%
527 \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.

```

528 \@tempcnta=\time
529 \divide\@tempcnta by 60
530 \ifnum\@tempcnta<10
531 \xdef#1{#1T0\the\@tempcnta}%
532 \else
533 \xdef#1{#1T\the\@tempcnta}%
534 \fi
535 \multiply\@tempcnta by -60
536 \advance\@tempcnta by \time
537 \ifnum\@tempcnta<10
538 \xdef#1{#1:0\the\@tempcnta}%
539 \else
540 \xdef#1{#1:\the\@tempcnta}%
541 \fi
542 \xdef#1{#1Z}%
543 }

```

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

```

544 \def\hyxmp@try@today#1#2{%
545 \ifmtargexp{\hyxmp@today@xmp}{%
546 \ifundefined{#1}{#2}%
547 }{%
548 }

```

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

```

549 \def\hyxmp@today@xmp{}

```

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

```

550 \hyxmp@try@today{\pdfcreationdate}{%
551 \edef\hyxmp@today@xmp{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
552 }

```

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

```

553 \hyxmp@try@today{\pdffeedback}{%
554 \edef\hyxmp@today@xmp{\expandafter\hyxmp@pdf@to@xmp@date\pdffeedback creationdate}%
555 }

```

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

```

556 \hyxmp@try@today{filemoddate}{%
557   \edef\hyxmp@today@xmp{\filemoddate{\jobname.log}}%
558   \edef\next{%
559     \edef\noexpand\hyxmp@today@xmp{\noexpand\hyxmp@as@xmp@date{\hyxmp@today@xmp}}%
560   }%
561   \next
562 }%

Case 4: None of the above. Do the best we can using the available TeX primitives
(\year, \month, \day, and \time.
563 \hyxmp@try@today{year}{%
564   \hyxmp@today@xmp@define\hyxmp@today@xmp
565 }

\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.
566 \expandafter\edef\expandafter\hyxmp@today@pdf\expandafter{%
567   \expandafter\hyxmp@xmp@to@pdf@date\expandafter{\hyxmp@today@xmp}%
568 }
```

3.3.3 Trimming leading and trailing spaces

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

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

```

569 \catcode'\Q=3

\hyxmp@trimspaces\x redefines \x to have the same replacement text sans leading
and trailing space tokens.
570 \newcommand{\hyxmp@trimspaces}[1]{%

Use grouping to emulate a multi-token afterassignment queue.
571   \begingroup
Put “\toks 0 {” into the afterassignment queue.
572   \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.
573   \expandafter\hyxmp@trimb\expandafter\noexpand#1Q Q}%

```

Transfer the trimmed text back into #1.

```
574 \edef#1{\the\toks0}%
575 }
```

`\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.

```
576 \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.

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

3.3.4 Converting text to XML

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

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

```
579 \newif\ifhyxmp@unicodetex
580 \ifnum64='^^^^0040\relax
581 \hyxmp@unicodetextrue
582 \else
583 \hyxmp@unicodetexfalse
584 \fi
```

`\SE->pdfdoc@03` Preserve ETX (^^C), which is normally an invalid character in PDFDocEncoding. We use it in hyperxmp (and specifically in `\hyxmp@xmlify` below) as a list-element separator.

```
585 \expandafter\def\csname SE->pdfdoc@03\endcsname{0003}
```

`\SE->pdfdoc@15` Preserve NAK (^^U), which is normally an invalid character in PDFDocEncoding. We use it in hyperxmp (and specifically in `\hyxmp@xmlify` below) as a placeholder for an underscore character.

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

`\hyxmp@xmlify` Given a piece of text defined using `\pdfstringdef` (i.e., with many special characters redefined to have category code 11), set `\hyxmp@xmlified` to the same text `\hyxmp@text`

but with all occurrences of “<” replaced with <;, all occurrences of “>” replaced with >;, and all occurrences of “&” replaced with &.

```

587 \newcommand*{\hyxmp@xmlify}[1]{%
588   \gdef\hyxmp@xmlified{%
Escaped PDF string → PDFDocEncoding/Unicode
589   \EdefUnescapeString\hyxmp@text{#1}%
590   \ifhyxmp@unicodetex
PDFDocEncoding/Unicode → UTF-32BE
591     \hyxmp@is@unicode\hyxmp@text{%
592       \StringEncodingConvert
593       \hyxmp@text\hyxmp@text{utf16be}{utf32be}%
594     }{%
595       \ifXeTeX
596         \hyxmp@xetex@crap
597       \else
598         \StringEncodingConvert
599         \hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
600       \fi
601     }%
UTF-32BE → UTF-32BE as hex string
602     \EdefEscapeHex\hyxmp@text{\hyxmp@text}%
UTF-32BE → XML in ASCII
603     \edef\hyxmp@text{%
604       \expandafter
605       \expandafter\hyxmp@toxml@unicodetex\hyxmp@text
606       \relax\relax\relax\relax\relax\relax\relax\relax
607     \else
PDFDocEncoding/Unicode → UTF-8
608     \hyxmp@is@unicode\hyxmp@text{%
609       \StringEncodingConvert
610       \hyxmp@text\hyxmp@text{utf16be}{utf8}%
611     }{%
612       \StringEncodingConvert
613       \hyxmp@text\hyxmp@text{pdfdoc}{utf8}%
614     }%
UTF-8 → UTF-8 as hex string
615     \EdefEscapeHex\hyxmp@text{\hyxmp@text}%
UTF-8 as hex string → XML in UTF-8 as hex string
616     \edef\hyxmp@text{%
617       \expandafter\hyxmp@toxml\hyxmp@text\@empty\@empty
618     }%
XML in UTF-8 as hex string → XML in UTF-8
619     \EdefUnescapeHex\hyxmp@text{\hyxmp@text}%
620   \fi
621   \global\let\hyxmp@xmlified\hyxmp@text

```


622 }

`\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.

```

623 \begingroup
624   \lccode'\<=254 %
625   \lccode'\>=255 %
626   \catcode254=12 %
627   \catcode255=12 %
628 \lowercase{\endgroup
629   \def\hyxmp@is@unicode#1{%
630     \expandafter\hyxmp@@is@unicode#1<>\@nil
631   }%
632   \def\hyxmp@@is@unicode#1<>#2\@nil{%
633     \ifx\#1\%
634       \expandafter\@firstoftwo
635     \else
636       \expandafter\@secondoftwo
637     \fi
638   }%
639 }
```

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

```

640 \def\hyxmp@toxml#1#2{%
641   \ifx#1\@empty
642   \else
643     \ifnum"#1#2='\& %
644       26616D703B% &
645     \else\ifnum"#1#2='\< %
646       266C743B% <
647     \else\ifnum"#1#2='\> %
648       2667743B% >
649   \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.

```

650   \@ifundefined{pdfmark}{%
651     #1#2%
652   }{%
```

```

653     \ifnum"#1#2='\'( %
654         5C28% \((
655     \else\ifnum"#1#2='\' ) %
656         5C29% \)
657     \else
658         #1#2%
659     \fi\fi
660 }%
661 \fi\fi\fi
662 \expandafter\hyxmp@toxml
663 \fi
664 }

```

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

```

\hyxmp@text
665 \def\hyxmp@toxml@unicodetex#1#2#3#4#5#6#7#8{%
666     \ifx#1\relax
667     \else
668         \ifnum"#1#2#3#4#5#6#7#8>127 %
669             \uccode'\*"#1#2#3#4#5#6#7#8\relax
670             \uppercase{%
671                 \edef\hyxmp@text{\hyxmp@text *}%
672             }%
673         \else\ifnum"#7#8='\'< %
674             \edef\hyxmp@text{\hyxmp@text &lt;}%
675         \else\ifnum"#7#8='\'& %
676             \edef\hyxmp@text{\hyxmp@text &amp;%
677         \else\ifnum"#7#8='\'> %
678             \edef\hyxmp@text{\hyxmp@text &gt;}%
679         \else\ifnum"#7#8='\' %
680             \edef\hyxmp@text{\hyxmp@text\space}%
681         \else
682             \uccode'\*"#7#8\relax
683             \uppercase{%
684                 \edef\hyxmp@text{\hyxmp@text *}%
685             }%
686         \fi\fi\fi\fi\fi
687     \expandafter\hyxmp@toxml@unicodetex
688 \fi
689 }

```

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

```

690 \def\hyxmp@skipzeros#1{%
691     \ifx#10%
692     \expandafter\hyxmp@skipzeros
693     \fi
694 }

```

`\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@xetex@crap`

`\hyxmp@try`

`\hyxmp@crap@result`

`\hyxmp@text`

```

695 \begingroup
696 \def\x#1{\endgroup
697   \def\hyxmp@xetex@crap{%
698     \edef\hyxmp@try{%
699       \expandafter\hyxmp@SpaceOther\hyxmp@text#1\@nil
700     }%
701     \let\hyxmp@crap@result=N%
702     \expandafter\hyxmp@crap@test\hyxmp@try\relax
703     \ifx\hyxmp@crap@result Y%
704       \let\hyxmp@text\@empty
705       \expandafter\hyxmp@crap@convert\hyxmp@try\relax
706     \else
707       \StringEncodingConvert\hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
708     \fi
709   }%
710 }
711 \x{ }

```

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

```

712 \begingroup
713 \catcode'\~ =12 %
714 \lccode'\~ ='\ %
715 \lowercase{\endgroup
716 \def\hyxmp@SpaceOther#1 #2\@nil{%
717   #1%
718   \ifx\relax#2\relax
719     \expandafter\@gobble
720   \else
721     ~%
722     \expandafter\@firstofone
723   \fi
724   {\hyxmp@SpaceOther#2\@nil}%
725 }%
726 }

```

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

```

727 \def\hyxmp@crap@test#1{%
728   \ifx#1\relax
729   \else
730     \ifnum'#1>127 %
731       \let\hyxmp@crap@result=Y%
732       \expandafter\expandafter\expandafter\hyxmp@skiptorelax
733     \else
734       \expandafter\expandafter\expandafter\hyxmp@crap@test
735     \fi
736   \fi
737 }

```

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

```

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

```

```

\hyxmp@crap@convert  Convert a hexadecimal string to a number.
\hyxmp@num 739 \def\hyxmp@crap@convert#1{%
\hyxmp@text 740 \ifx#1\relax
741 \else
742 \edef\hyxmp@num{\number'#1}%
743 \ifnum\hyxmp@num>"FFFFFF %
744 \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"1000000}\relax
745 \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
746 \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"1000000}}%
747 \else
748 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
749 \fi
750 \ifnum\hyxmp@num>"FFFF %
751 \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"10000}\relax
752 \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
753 \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"10000}}%
754 \else
755 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
756 \fi
757 \ifnum\hyxmp@num>"FF %
758 \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"100}\relax
759 \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
760 \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"100}}%
761 \else
762 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
763 \fi
764 \ifnum\hyxmp@num>0 %
765 \lccode'\!=\hyxmp@num\relax
766 \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
767 \else
768 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
769 \fi
770 \expandafter\hyxmp@crap@convert
771 \fi
772 }

```

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

```

773 \begingroup
774 \catcode0=12 %
775 \gdef\hyxmp@zero{^^00}%
776 \endgroup

```

3.3.5 Outputting structured XML

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

`\hyxmp@extra@indent` This macro is used internally to increase the amount of indentation when writing

certain XML data. It is normally defined as empty but can temporarily be redefined to a sequence of `\space` characters.

```
777 \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.

```
778 \newcommand*{\hyxmp@add@simple}[2]{%
779   \ifnotmtargexp{#2}{%
780     \hyxmp@xmllify{#2}%
781     \hyxmp@add@to@xml{\hyxmp@extra@indent_<}}%
782     \xdef\hyxmp@xml{\hyxmp@xml#1}%
783     \hyxmp@add@to@xml{>\hyxmp@xmllified</}%
784     \xdef\hyxmp@xml{\hyxmp@xml#1>^^J}%
785   }%
786 }
```

`\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.

```
787 \newcommand*{\hyxmp@add@simple@var}[2]{%
788   \expandafter\ifx\csname#2\endcsname\relax
789   \else
790     \hyxmp@xmllify{\csname#2\endcsname}%
791     \hyxmp@add@to@xml{%
792       \hyxmp@extra@indent_<#1>\hyxmp@xmllified</#1>^^J%
793     }%
794   \fi
795 }
```

`\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.

```
796 \newcommand*{\hyxmp@add@simple@lang}[2]{%
797   \ifnotmtarg{#2}{%
798     \hyxmp@xmllify{#2}%
799     \expandafter\hyxmp@add@simple@lang{i\hyxmp@xmllified\relax{#1}%
800   }%
801 }
```

`\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.

```

802 \newcommand*{\hyxmp@add@simple@lang@i}{%
803   \@ifnextchar[\hyxmp@add@simple@lang@ii{\hyxmp@add@simple@lang@ii[]}%
804 }

```

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

```

805 \def\hyxmp@add@simple@lang@ii[#1]#2\relax#3{%
806   \@ifnotmtarg{#2}{%
807     \hyxmp@xmlify{#2}%
808     \@ifmtarg{#1}{%
809       \hyxmp@add@to@xml{%
810         <#3>\hyxmp@xmlified</#3>^^J%
811       }%
812     }%
813     \hyxmp@add@to@xml{%
814       <#3 xml:lang="#1">\hyxmp@xmlified</#3>^^J%
815     }%
816   }%
817 }%
818 }

```

3.3.6 Providing metadata in multiple languages

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

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

```

819 \def\hyxmp@alt@title{}
820 \def\hyxmp@alt@description{}
821 \def\hyxmp@alt@rights{}

```

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

```

822 \newcommand{\hyxmp@LA@accept}[2]{%
823   \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`.

```

824   \hyxmp@pdfstringdef\hyxmp@value{##1}%
825   \xdef#2{#2\noexpand\do {\hyxmp@cur@lang} {\hyxmp@value}}%

```

```
826 }
827 }
```

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

```
828 \hyxmp@LA@accept{pdftitle}{\hyxmp@alt@title}
829 \hyxmp@LA@accept{pdfsubject}{\hyxmp@alt@description}
830 \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 $\langle key \rangle = \langle value \rangle$ 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.

```
831 \newcommand{\XMPLangAlt}[2]{%
832   \let\do=\relax
```

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

```
833   \edef\hyxmp@cur@lang{#1}%
834   \setkeys{hyxmp@LA}{#2}%
835 }
```

3.4 UUID generation

We use a linear congruential generator to produce pseudorandom version 4 UUIDs [11]. True, this method has its flaws but it’s simple to implement in \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.

```
836 \def\hyxmp@modulo@a#1{%
837   \@tempcntb=\@tempcnta
838   \divide\@tempcntb by #1
839   \multiply\@tempcntb by #1
840   \advance\@tempcnta by -\@tempcntb
841 }
```

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

```
\hyxmp@big@prime@ii 842 \def\hyxmp@big@prime{536870923}
843 \def\hyxmp@big@prime@iii{536870027}
```

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

```
\hyxmp@one@token 844 \def\hyxmp@seed@rng#1{%
845   \@tempcnta=\hyxmp@big@prime
846   \futurelet\hyxmp@one@token\hyxmp@seed@rng@i#1\@empty
847 }
```

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

```

\next 848 \def\hyxmp@seed@rng@i{%
      849   \ifx\hyxmp@one@token\@empty
      850     \let\next=\relax
      851   \else
      852     \def\next##1{%
      853       \multiply\@tempcnta by 3
      854       \advance\@tempcnta by '##1
      855       \hyxmp@modulo@a{\hyxmp@big@prime}%
      856       \futurelet\hyxmp@one@token\hyxmp@seed@rng@i
      857     }%
      858   \fi
      859 \next
      860 }

```

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

```

      861 \def\hyxmp@set@rand@num{%
      862   \@tempcnta=\hyxmp@rand@num
      863   \multiply\@tempcnta by 3
      864   \advance\@tempcnta by \hyxmp@big@prime@ii
      865   \hyxmp@modulo@a{\hyxmp@big@prime}%
      866   \xdef\hyxmp@rand@num{\the\@tempcnta}%
      867 }

```

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

```

      868 \def\hyxmp@append@hex#1{%
      869   \hyxmp@set@rand@num
      870   \@tempcnta=\hyxmp@rand@num
      871   \hyxmp@modulo@a{16}%
      872   \ifnum\@tempcnta<10
      873     \xdef#1{#1\the\@tempcnta}%
      874   \else

```

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

```

      875     \advance\@tempcnta by -10
      876     \ifcase\@tempcnta
      877       \xdef#1{#1a}%
      878       \or\xdef#1{#1b}%
      879       \or\xdef#1{#1c}%
      880       \or\xdef#1{#1d}%
      881       \or\xdef#1{#1e}%
      882       \or\xdef#1{#1f}%
      883     \fi
      884   \fi
      885 }

```


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

```
886 \def\hyxmp@append@hex@iii#1{%  
887   \hyxmp@append@hex#1%  
888   \hyxmp@append@hex#1%  
889   \hyxmp@append@hex#1%  
890 }
```

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

```
891 \def\hyxmp@append@hex@iv#1{%  
892   \hyxmp@append@hex@iii#1%  
893   \hyxmp@append@hex#1%  
894 }
```

`\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`.

```
895 \def\hyxmp@create@uuid#1{%  
896   \def#1{uuid:}%  
897   \hyxmp@append@hex@iv#1%  
898   \hyxmp@append@hex@iv#1%  
899   \g@addto@macro#1{-}%  
900   \hyxmp@append@hex@iv#1%  
901   \g@addto@macro#1{-4}%  
902   \hyxmp@append@hex@iii#1%  
903   \g@addto@macro#1{-}%
```

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

```
904   \hyxmp@set@rand@num  
905   \@tempcnta=\hyxmp@rand@num  
906   \hyxmp@modulo@a{4}%  
907   \ifcase\@tempcnta  
908     \g@addto@macro#1{8}%  
909     \or\g@addto@macro#1{9}%  
910     \or\g@addto@macro#1{a}%  
911     \or\g@addto@macro#1{b}%  
912   \fi  
913   \hyxmp@append@hex@iii#1%  
914   \g@addto@macro#1{-}%  
915   \hyxmp@append@hex@iv#1%  
916   \hyxmp@append@hex@iv#1%  
917   \hyxmp@append@hex@iv#1%  
918 }
```

`\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.

```
919 \newcommand*{\hyxmp@def@DocumentID}{%
```

```

920 \edef\hyxmp@seed@string{\jobname:\@pdftitle:\@pdfauthor:}%
921 \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
922 \edef\hyxmp@rand@num{\the\@tempcnta}%
923 \hyxmp@create@uuid\hyxmp@DocumentID
924 }

```

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

```

925 \newcommand*{\hyxmp@def@InstanceID}{%
926   \hyxmp@today@xmp@define{\hyxmp@seed@string}%
927   \edef\hyxmp@seed@string{%
928     \jobname:\@pdftitle:\@pdfauthor:\hyxmp@today@xmp:\hyxmp@seed@string
929   }%
930   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
931   \edef\hyxmp@rand@num{\the\@tempcnta}%
932   \hyxmp@create@uuid\hyxmp@InstanceID
933 }

```

3.5 Constructing the XMP packet

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

3.5.1 XMP utility functions

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

```

934 \newcommand*{\hyxmp@add@to+xml}[1]{%
935   \bgroup
936   \@tempcnta=0
937   \ifhyxmp@unicodetex

```

```

938     \@tempcntb=65536%
939   \else
940     \@tempcntb=256%
941   \fi
942   \loop
943     \lccode\@tempcnta=\@tempcnta
944     \advance\@tempcnta by 1
945     \ifnum\@tempcnta<\@tempcntb
946   \repeat
947   \lccode'\_=' \relax
948   \lccode'\^C=' \relax
949   \lccode'\^U=' \relax
950   \lowercase{\xdef\hyxmp@new@xml{#1}}%
951   \xdef\hyxmp@xml{\hyxmp@xml\hyxmp@new@xml}%
952 \egroup
953 }

```

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

```

954 \bgroup
955 \catcode'\#=11
956 \gdef\hyxmp@hash{#}
957 \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.

```

958 \bgroup
959 \xdef\hyxmp@xml{%
960   \hyxmp@add@to@xml{%
961     -----~J%
962   }
963   \xdef\hyxmp@padding{\hyxmp@xml}%
964 \egroup
965 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
966 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
967 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
968 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
969 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}

```

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

```

970 \newcommand*{\hyxmp@x@default}{x-default}

```

3.5.2 The Adobe PDF schema

Older versions of `hyperref` defined a default producer; newer versions do not. Instead, they let the `TEX` engine define the producer itself. This poses a problem for PDF/A compliance because `hyperxmp` sees an empty producer and therefore omits writing a `pdf:Producer` to the XMP packet, causing a mismatch between the data in the

XMP packet and the data in the PDF Info dictionary. To ensure consistency between XMP and Info, we explicitly define our own default `\@pdfproducer` here.

```

\@pdfproducer Define \@pdfproducer using the banner string if available or the TEX engine's
\hyxmp@define@pdfproducer version number if not.
971 \newcommand*{\hyxmp@define@pdfproducer}{%
972   \gdef\@pdfproducer{TeX}
973   \ifLuaTeX
974     \expandafter\hyxmp@banner@to@producer\expandafter{\luatexbanner}%
975   \else
976     \ifPDFTeX
977       \expandafter\hyxmp@banner@to@producer\expandafter{\pdftexbanner}%
978     \else
979       \ifXeTeX
980         \edef\@pdfproducer{XeTeX version \the\XeTeXversion\XeTeXrevision}%
981       \fi
982     \fi
983   \fi
984 }

\@pdfproducer Define \@pdfproducer as the TEX engine's banner string (e.g., "This is pdfTeX,
\hyxmp@banner@to@producer Version 3.14159265-2.6-1.40.20 (TeX Live 2019/Debian) kpathsea
version 6.3.1"), removing the initial "This is" if possible (specifically, when
ε-TEX's \scantokens primitive is available).
985 \def\hyxmp@banner@to@producer#1{%
986   \ifx\scantokens\relax
987     \gdef\@pdfproducer{#1}%
988   \else
989     {\scantokens{\makeatletter\hyxmp@remove@this#1\relax}}%
990   \fi
991 }

\@pdfproducer Define \@pdfproducer as a given banner string with the initial "This is" stripped
\hyxmp@remove@this off the beginning.
992 \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.
993 \AtBeginDocument{%
994   \ifx\@pdfproducer\relax
995     \hyxmp@define@pdfproducer
996   \fi
997 }

\hyxmp@pdf@schema Add properties defined by the Adobe PDF schema to the \hyxmp+xml macro.
998 \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.

```

999 \hyxmp@add@simple@var{pdf:Producer}{@pdfproducer}%
1000 \hyxmp@add@simple@var{pdf:Keywords}{@pdfkeywords}%
1001 \hyxmp@add@simple{pdf:Trapped}{@pdftrapped}%

```

Specify the PDF version.

```

1002 \@ifundefined{pdfvariable}{%
1003   \@ifundefined{pdfminorversion}{%

```

Case 1: Neither `\pdfvariable` nor `\pdfminorversion` is defined (Xe_{La}TeX and regular L^AT_EX).

```

1004   }{%

```

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

```

1005     \hyxmp@add@simple{pdf:PDFVersion}{1.\the\pdfminorversion}%
1006   }%
1007 }{%

```

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

```

1008   \hyxmp@add@simple{pdf:PDFVersion}{1.\the\pdfvariable minorversion}%
1009 }%
1010 }

```

3.5.3 The Dublin Core schema

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

```

1011 \newcommand*{\hyxmp@rdf@dc}[3][\iffalse]{%

```

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

```

1012 \@ifmtargexp{#3}{\@tempswafalse}{\@tempswatrue}%
1013 #1
1014   \@tempswatrue
1015 \fi

```

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

```

1016 \if@tempswa
1017   \hyxmp@xmlify{#3}%

```

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

```

1018 \let\hyxmp@value=\hyxmp@xmlified
1019 \hyxmp@add@to@xml{%
1020 -----<dc:#2>^^J%
1021 -----<rdf:Alt>^^J%
1022 }%
1023 \ifx\@pdfmetalang\hyxmp@x@default
1024 \else
1025 \hyxmp@xmlify{\@pdfmetalang}%
1026 \hyxmp@add@to@xml{%
1027 -----<rdf:li xml:lang="\hyxmp@xmlified">\hyxmp@value</rdf:li>^^J%
1028 }%
1029 \fi
1030 \hyxmp@add@to@xml{%
1031 -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@value</rdf:li>^^J%
1032 }%

```

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

```

1033 \def\do##1##2{
1034 \hyxmp@xmlify{##2}%
1035 \hyxmp@add@to@xml{%
1036 -----<rdf:li xml:lang="##1">\hyxmp@xmlified</rdf:li>^^J%
1037 }%
1038 }%
1039 \csname hyxmp@alt@#2\endcsname

```

Complete this XMP element.

```

1040 \hyxmp@add@to@xml{%
1041 -----</rdf:Alt>^^J%
1042 -----</dc:#2>^^J%
1043 }%
1044 \fi
1045 }%

```

`\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.

```

1046 \newcommand*{\hyxmp@list@to@xml}[4][\iffalse]{%

```

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

```

1047 \ifmtargexp{#4}{\@tempswafalse}{\@tempswattrue}%
1048 #1
1049 \@tempswattrue
1050 \fi

```

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

```

1051 \if@tempswa

```

```

1052 \hyxmp@add@to@xml{%
1053 -----<dc:#2>^^J%
1054 -----<rdf:#3>^^J%
1055 }%
1056 \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`.

```

1057 \hyxmp@xmlify{#4}%
1058 \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
1059 \def\@elt#1{%
1060 \hyxmp@add@to@xml{%
1061 -----<rdf:li>#1</rdf:li>^^J%
1062 }%
1063 }%
1064 \hyxmp@list
1065 \egroup
1066 \hyxmp@add@to@xml{%
1067 -----</rdf:#3>^^J%
1068 -----</dc:#2>^^J%
1069 }%
1070 \fi
1071 }

```

`\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.

```

1072 \newcommand{\hyxmp@singleton@dc}[3][Bag]{%
1073 \@ifnotmtarg{#3}{%
1074 \hyxmp@xmlify{#3}%
1075 \hyxmp@add@to@xml{%
1076 -----<dc:#2>^^J%
1077 -----<rdf:#1>^^J%
1078 -----<rdf:li>\hyxmp@xmlified</rdf:li>^^J%
1079 -----</rdf:#1>^^J%
1080 -----</dc:#2>^^J%
1081 }%
1082 }
1083 }

```

`\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`, and the `dc:language` property if the author specified `pdflang`. We also specify the `dc:date` property using the date the document was run through L^AT_EX and the `dc:source` property using the base name of the source file with `.tex` appended.

```

1084 \newcommand*{\hyxmp@dc@schema}{%
1085   \hyxmp@add@simple{dc:format}{application/pdf}%
1086   \hyxmp@rdf@dc[\ifHy@pdfa]{title}{\@pdftitle}%
1087   \hyxmp@rdf@dc[\ifHy@pdfa]{description}{\@pdfsubject}%
1088   \hyxmp@rdf@dc{rights}{\@pdfcopyright}%
1089   \hyxmp@singleton@dc{publisher}{\@pdfpublisher}%
1090   \hyxmp@singleton@dc[Seq]{date}{\hyxmp@today@xmp}%
1091   \hyxmp@singleton@dc{language}{\@pdflang}%
1092   \hyxmp@singleton@dc{type}{\@pdftype}%
1093   \hyxmp@list@to@xml[\ifHy@pdfa]{creator}{Seq}{\hyxmp@pdfauthor}%
1094   \hyxmp@list@to@xml{subject}{Bag}{\hyxmp@pdfkeywords}%
1095   \ifx\@pdfsource\@empty
1096   \else
1097     \hyxmp@add@simple{dc:source}{\@pdfsource}%
1098   \fi
1099 }

```

3.5.4 The XMP Rights Management schema

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

```

1100 \newcommand*{\hyxmp@xmpRights@schema}{%

```

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

```

1101   \let\hyxmp@rights=\@empty
1102   \ifx\@pdflicenseurl\@empty
1103   \else
1104     \def\hyxmp@rights{YES}%
1105   \fi
1106   \ifx\@pdfcopyright\@empty
1107   \else
1108     \def\hyxmp@rights{YES}%
1109   \fi

```

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

```

1110   \ifx\hyxmp@rights\@empty
1111   \else
1112     \ifx\@pdfcopyright\@empty
1113     \else
1114       \hyxmp@add@simple{xmpRights:Marked}{True}%
1115     \fi
1116     \hyxmp@add@simple{xmpRights:WebStatement}{\@pdflicenseurl}%
1117   \fi
1118 }

```


3.5.5 The XMP Media Management schema

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

```
1119 \gdef\hyxmp@mm@schema{%
1120   \ifmtargexp{\hyxmp@DocumentID}{\hyxmp@def@DocumentID}{}%
1121   \ifmtargexp{\hyxmp@InstanceID}{\hyxmp@def@InstanceID}{}%
1122   \hyxmp@add@simple{xmpMM:DocumentID}{\hyxmp@DocumentID}%
1123   \hyxmp@add@simple{xmpMM:InstanceID}{\hyxmp@InstanceID}%
1124   \hyxmp@add@simple{xmpMM:VersionID}{\pdfversionid}%
1125   \hyxmp@add@simple{xmpMM:RenditionClass}{\pdfrendition}%
1126 }
```

3.5.6 The XMP Basic schema

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

```
1127 \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.
1128   \ifmtargexp{\pdfcreationdate}{%
1129     \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@today@xmp}%
1130   }{%
1131     \hyxmp@add@simple{xmp:CreateDate}{%
1132       \expandafter\hyxmp@as@xmp@date\expandafter{\pdfcreationdate}}%
1133   }%
  For the document's modification date, use the user-specified \pdfmoddate if
  defined and non-empty. Otherwise use our fabricated \hyxmp@today@xmp.
1134   \ifmtargexp{\pdfmoddate}{%
1135     \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@today@xmp}%
1136   }{%
1137     \hyxmp@add@simple{xmp:ModifyDate}{%
1138       \expandafter\hyxmp@as@xmp@date\expandafter{\pdfmoddate}}%
1139   }%
  For the document's metadata date, use the user-specified \pdfmetadatetime if
  defined and non-empty. Otherwise use our fabricated \hyxmp@today@xmp.
1140   \ifmtargexp{\pdfmetadatetime}{%
1141     \hyxmp@add@simple{xmp:MetadataDate}{\hyxmp@today@xmp}%
1142   }{%
1143     \hyxmp@add@simple{xmp:MetadataDate}{\pdfmetadatetime}%
1144   }%
}
```

Define the creation tool and the base URL.

```
1145 \hyxmp@add@simple{xmp:CreatorTool}{\@pdfcreator}%
1146 \hyxmp@add@simple{xmp:BaseURL}{\@baseurl}%
1147 }
```

3.5.7 The Photoshop schema

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

```
1148 \gdef\hyxmp@photoshop@schema{%
1149 \edef\hyxmp@photoshop@data{\@pdfauthortitle\@pdfcaptionwriter}%
1150 \hyxmp@add@simple{photoshop:AuthorsPosition}{\@pdfauthortitle}%
1151 \hyxmp@add@simple{photoshop:CaptionWriter}{\@pdfcaptionwriter}%
1152 }
```

3.5.8 PDF/* Identification schemata

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

```
1153 \newcommand*{\hyxmp@pdfa@id@schema}{%
1154 \ifHy@pdfa
1155 \hyxmp@add@simple{pdfaid:part}{\@pdfapart}%
1156 \hyxmp@add@simple{pdfaid:conformance}{\@pdfaconformance}%
1157 \fi
1158 }
```

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

```
1159 \newcommand*{\hyxmp@pdfua@id@schema}{%
1160 \hyxmp@add@simple{pdfuaid:part}{\@pdfuapart}%
1161 }
```

`\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.

```
1162 \newcommand*{\hyxmp@pdfx@id@schema}{%
1163 \@tempcnta=0\hyxmp@pdfx@major\relax
1164 \ifnum\@tempcnta=0
1165 \else
1166 \ifnum\@tempcnta=1
1167 \hyxmp@add@simple{pdfx:GTS_PDFXVersion}{PDF/X-1:2001}%
1168 \hyxmp@add@simple{pdfx:GTS_PDFXConformance}{\@pdfxstandard}%
1169 \else
1170 \ifnum\@tempcnta<4
1171 \hyxmp@add@simple{pdfx:GTS_PDFXVersion}{\@pdfxstandard}%

```

```

1172     \else
1173     \hyxmp@add@simple{pdfxid:GTS_PDFXVersion}{\@pdfxstandard}%
1174     \fi
1175 \fi
1176 \fi
1177 }

```

3.5.9 The IPTC Photo Metadata schema

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

```

1178 \begingroup
1179 \catcode'\&=12
1180 \catcode'\#=12
1181 \gdef\xmplinesep{&\#xA;}
1182 \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.

```

1183 \newcommand*{\hyxmp@list@to@lines}[2]{%
1184 \ifnotmtargexp{#2}{%
1185 \bgroup
1186 \hyxmp@add@to@xml{%
1187 \hyxmp@extra@indent_____<#1>%
1188 }%

```

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

```

1189 \def\@elt@first##1{%
1190 \hyxmp@add@to@xml{##1}%
1191 \let\@elt=\@elt@rest
1192 }%

```

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

```

1193 \def\@elt@rest##1{%
1194 \hyxmp@add@to@xml{\xmplinesep##1}%
1195 }%

```

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

```

1196 \let\@elt=\@elt@first
1197 \hyxmp@xmllify{#2}%
1198 \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmllified}%
1199 \hyxmp@list
1200 \hyxmp@add@to@xml{</#1>\sim J}%
1201 \egroup
1202 }%
1203 }

```

`\hyxmp@iptc@schema` Add properties defined by the IPTC Photo Metadata schema [9] to the `\hyxmp+xml` macro. We currently support only the `Iptc4xmpCore:CreatorContactInfo` property, although this is a structure containing multiple fields.

```

1204 \gdef\hyxmp@iptc@schema{%
    Because we currently support only Iptc4xmpCore:CreatorContactInfo it suffices to
    check if we have any relevant data. If so, we instantiate a Iptc4xmpCore:ContactInfo
    structure with all available fields.
1205 \ifx\hyxmp@iptc@data\@empty
1206 \else
1207 \hyxmp@add@to+xml{%
1208 -----<Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">^^J%
1209 }%
    We locally redefine \hyxmp@extra@indent to increase the indentation of the as-
    signments to Iptc4xmpCore:CreatorContactInfo's fields.
1210 \bgroup
1211 \edef\hyxmp@extra@indent{\hyxmp@extra@indent\space\space}%
1212 \hyxmp@list@to@lines{Iptc4xmpCore:CiAdrExtadr}{\@pdfcontactaddress}%
1213 \hyxmp@add@simple{Iptc4xmpCore:CiAdrCity}{\@pdfcontactcity}%
1214 \hyxmp@add@simple{Iptc4xmpCore:CiAdrRegion}{\@pdfcontactregion}%
1215 \hyxmp@add@simple{Iptc4xmpCore:CiAdrPcode}{\@pdfcontactpostcode}%
1216 \hyxmp@add@simple{Iptc4xmpCore:CiAdrCtry}{\@pdfcontactcountry}%

\xmplinesep The IPTC standard states that sets of telephone numbers, email addresses, and
URLs for the contact person or institution, “[m]ay have to be separated by a
comma in the user interface” [9]. This is rather ambiguous: Does the comma
appear only in the user interface or also in the generated XML? Here we assume
the latter interpretation and temporarily redefine \xmplinesep as a comma and
use \hyxmp@list@to@lines to insert the data. Unlike \hyxmp@add@simple, this
approach trims all spaces surrounding commas.
1217 \def\xmplinesep{,}%
1218 \hyxmp@list@to@lines{Iptc4xmpCore:CiTelWork}{\@pdfcontactphone}%
1219 \hyxmp@list@to@lines{Iptc4xmpCore:CiEmailWork}{\@pdfcontactemail}%
1220 \hyxmp@list@to@lines{Iptc4xmpCore:CiUrlWork}{\@pdfcontacturl}%
1221 \egroup
1222 \hyxmp@add@to+xml{%
1223 -----</Iptc4xmpCore:CreatorContactInfo>^^J%
1224 }%
1225 \fi
1226 }

```

3.5.10 The PRISM Basic Metadata schema

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

```

1227 \newcommand*{\hyxmp@prism@schema}{%
1228 \ifx\hyxmp@prism@data\@empty
1229 \else
1230 \hyxmp@add@simple{prism:complianceProfile}{three}%

```

```

1231 \fi
1232 \hyxmp@add@simple@lang{prism:subtitle}{\@pdfsubtitle}%
1233 \hyxmp@add@simple@lang{prism:publicationName}{\@pdfpublication}%
1234 \hyxmp@add@simple{prism:aggregationType}{\@pdfpubtype}%
1235 \hyxmp@add@simple@lang{prism:bookEdition}{\@pdfbookedition}%
1236 \hyxmp@add@simple{prism:volume}{\@pdfvolumenum}%
1237 \hyxmp@add@simple{prism:number}{\@pdfissuenum}%
1238 \hyxmp@add@simple{prism:pageRange}{\@pdfpagerange}%
1239 \hyxmp@add@simple{prism:isbn}{\@pdfisbn}%
1240 \hyxmp@add@simple{prism:issn}{\@pdfissn}%
1241 \hyxmp@add@simple{prism:eIssn}{\@pdfeissn}%
1242 \hyxmp@add@simple{prism:doi}{\@pdfdoi}%
1243 \hyxmp@add@simple{prism:url}{\@pdfurl}%
1244 \hyxmp@add@simple{prism:byteCount}{\@pdfbytes}%
1245 \hyxmp@add@simple{prism:pageCount}{\@pdfnumpages}%
1246 }

```

3.5.11 XMP extension schemata

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

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

```
1247 \newcommand*{\hyxmp@check@iptc@data}{%
```

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

```

1248 \edef\hyxmp@iptc@data{%
1249   \@pdfcontactaddress
1250   \@pdfcontactcity
1251   \@pdfcontactregion
1252   \@pdfcontactpostcode
1253   \@pdfcontactcountry
1254   \@pdfcontactphone
1255   \@pdfcontactemail
1256   \@pdfcontacturl
1257 }%
1258 }%

```

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

```
1259 \newcommand*{\hyxmp@check@prism@data}{%
```

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

```

1260 \edef\hyxmp@prism@data{%
1261   \@pdfbookedition

```

```

1262 \pdfbytes
1263 \pdfdoi
1264 \pdfeisn
1265 \pdfisbn
1266 \pdfissn
1267 \pdfissuenum
1268 \pdfnumpages
1269 \pdfpagerange
1270 \pdfpublication
1271 \pdfpubtype
1272 \pdfsubtitle
1273 \pdfurl
1274 \pdfvolumenum
1275 }%
1276 }%

```

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

```

1277 \newcommand*{\hyxmp@begin@extension@decls}{%
1278 \hyxmp@add@to@xml{%
1279 -----<pdfaExtension:schemas>^^J%
1280 -----<rdf:Bag>^^J%
1281 }%
1282 }

```

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

```

1283 \newcommand*{\hyxmp@end@extension@decls}{%
1284 \hyxmp@add@to@xml{%
1285 -----</rdf:Bag>^^J%
1286 -----</pdfaExtension:schemas>^^J%
1287 }%
1288 }

```

`\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.

```

1289 \newcommand*{\hyxmp@begin@ext@decl}[3]{%
1290 \hyxmp@add@to@xml{%
1291 -----<rdf:li rdf:parseType="Resource">^^J%
1292 -----<pdfaSchema:schema>#1</pdfaSchema:schema>^^J%
1293 -----<pdfaSchema:prefix>#2</pdfaSchema:prefix>^^J%
1294 -----<pdfaSchema:namespaceURI>#3</pdfaSchema:namespaceURI>^^J%
1295 -----<pdfaSchema:property>^^J%
1296 -----<rdf:Seq>^^J%
1297 }%
1298 }%

```

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

```

1299 \newcommand*{\hyxmp@end@ext@decl}{%
1300 \hyxmp@add@to@xml{%

```

```

1301 -----</rdf:Seq>^^J%
1302 -----</pdfaSchema:property>^^J%
1303 -----</rdf:li>^^J%
1304  }%
1305 }%

```

`\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.

```

1306 \newcommand{\hyxmp@declare@property}[4][Text]{%
1307   \hyxmp@add@to+xml{%
1308     -----<rdf:li rdf:parseType="Resource">^^J%
1309     -----<pdfaProperty:name>}%
1310     \xdef\hyxmp+xml{\hyxmp+xml#2}%
1311     \hyxmp@add@to+xml{</pdfaProperty:name>^^J%
1312     -----<pdfaProperty:valueType>#1</pdfaProperty:valueType>^^J%
1313     -----<pdfaProperty:category>#3</pdfaProperty:category>^^J%
1314     -----<pdfaProperty:description>#4</pdfaProperty:description>^^J%
1315     -----</rdf:li>^^J%
1316   }%
1317 }%

```

`\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.

```

1318 \newcommand{\hyxmp@declare@field}[3][Text]{%
1319   \hyxmp@add@to+xml{%
1320     -----<rdf:li rdf:parseType="Resource">^^J%
1321     -----<pdfaField:name>#2</pdfaField:name>^^J%
1322     -----<pdfaField:valueType>#1</pdfaField:valueType>^^J%
1323     -----<pdfaField:description>#3</pdfaField:description>^^J%
1324     -----</rdf:li>^^J%
1325   }%
1326 }

```

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

```

1327 \newcommand*{\hyxmp@pdf@extensions}{%
1328   \hyxmp@begin@ext@decl
1329     {Adobe PDF Schema}%
1330     {pdf}%
1331     {http://ns.adobe.com/pdf/1.3/}%
1332   \hyxmp@declare@property
1333     {Trapped}%
1334     {internal}%
1335     {Indication if the document has been modified to include trapping information}%
1336   \hyxmp@end@ext@decl
1337 }%

```

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

```

1338 \newcommand*{\hyxmp@mm@extensions}{%
1339   \hyxmp@begin@ext@decl
1340     {XMP Media Management Schema}%
1341     {xmpMM}%
1342     {http://ns.adobe.com/xap/1.0/mm/}%
1343   \hyxmp@declare@property
1344     [URI]
1345     {DocumentID}%
1346     {internal}%
1347     {UUID based identifier for all versions and renditions of a document}%
1348   \hyxmp@declare@property
1349     [URI]
1350     {InstanceID}%
1351     {internal}%
1352     {UUID based identifier for specific incarnation of a document}%
1353   \hyxmp@declare@property
1354     {VersionID}%
1355     {internal}%
1356     {Document version identifier}%
1357   \hyxmp@declare@property
1358     {RenditionClass}%
1359     {internal}%
1360     {The manner in which a document is rendered}%
1361   \hyxmp@end@ext@decl
1362 }%

```

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

```

1363 \newcommand*{\hyxmp@pdfa@id@extensions}{%
1364   \hyxmp@begin@ext@decl
1365     {PDF/A Identification Schema}%
1366     {pdfaid}%
1367     {http://www.aiim.org/pdfa/ns/id/}%
1368   \hyxmp@declare@property
1369     [Integer]%
1370     {part}%
1371     {internal}%
1372     {Part of PDF/A standard}%
1373   \hyxmp@declare@property
1374     {conformance}%
1375     {internal}%
1376     {Conformance level of PDF/A standard}%
1377   \hyxmp@end@ext@decl
1378 }%

```

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

```

1379 \newcommand*{\hyxmp@pdfua@id@extensions}{%
1380   \hyxmp@begin@ext@decl
1381     {PDF/UA Universal Accessibility Schema}%
1382     {pdfuaid}%
1383     {http://www.aiim.org/pdfua/ns/id/}%

```



```

1384 \hyxmp@declare@property
1385     [Integer]%
1386     {part}%
1387     {internal}%
1388     {Part of ISO 14289 standard}%
1389 \hyxmp@end@ext@decl
1390 }%

```

`\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.

```

1391 \newcommand*{\hyxmp@pdfx@id@extensions}{%
1392     \ifx\hyxmp@pdfx@major\empty
1393     \else
1394         \hyxmp@begin@ext@decl
1395         {Adobe Document Info PDF/X eXtension Schema}%
1396         {pdfx}%
1397         {http://ns.adobe.com/pdfx/1.3/}%
1398         \hyxmp@declare@property
1399         {GTS_PDFXVersion}%
1400         {internal}%
1401         {ID of PDF/X standard}%
1402         \hyxmp@declare@property
1403         {GTS_PDFXConformance}%
1404         {internal}%
1405         {Conformance level of PDF/X standard}%
1406         \hyxmp@end@ext@decl
1407     \fi

```

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

```

1408 \@tempcnta=0\hyxmp@pdfx@major\relax
1409 \ifnum\@tempcnta>3
1410     \hyxmp@begin@ext@decl
1411     {PDF/X ID Schema}%
1412     {pdfxid}%
1413     {http://www.npes.org/pdfx/ns/id/}%
1414     \hyxmp@declare@property
1415     {GTS_PDFXVersion}%
1416     {internal}%
1417     {ID of PDF/X standard}%
1418     \hyxmp@end@ext@decl
1419 \fi
1420 }%

```

`\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.

```

1421 \newcommand*{\hyxmp@iptc@extensions}{%
1422     \hyxmp@begin@ext@decl
1423     {IPTC Core Schema}%

```

```

1424      {Iptc4xmpCore}%
1425      {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}%
1426 \hyxmp@declare@property
1427   [ContactInfo]
1428   {CreatorContactInfo}
1429   {external}
1430   {Document creator's contact information}

```

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

```

1431 \hyxmp@add@to@xml{%
1432 -----</rdf:Seq>^^J%
1433 -----</pdfaSchema:property>^^J%
1434 -----<pdfaSchema:valueType>^^J%
1435 -----<rdf:Seq>^^J%
1436 -----<rdf:li rdf:parseType="Resource">^^J%
1437 -----<pdfaType:type>ContactInfo</pdfaType:type>^^J%
1438 -----<pdfaType:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaType:
1439 -----<pdfaType:prefix>Iptc4xmpCore</pdfaType:prefix>^^J%
1440 -----<pdfaType:description>%
1441         Basic set of information to get in contact with a person%
1442         </pdfaType:description>^^J%
1443 -----<pdfaType:field>^^J%
1444 -----<rdf:Seq>^^J%
1445 }%
1446 \hyxmp@declare@field
1447   {CiAdrCity}%
1448   {Contact information city}%
1449 \hyxmp@declare@field
1450   {CiAdrCtry}%
1451   {Contact information country}%
1452 \hyxmp@declare@field
1453   {CiAdrExtadr}%
1454   {Contact information address}%
1455 \hyxmp@declare@field
1456   {CiAdrPcode}%
1457   {Contact information local postal code}%
1458 \hyxmp@declare@field
1459   {CiAdrRegion}%
1460   {Contact information regional information such as state or province}%
1461 \hyxmp@declare@field
1462   {CiEmailWork}%
1463   {Contact information email address(es)}%
1464 \hyxmp@declare@field
1465   {CiTelWork}%
1466   {Contact information telephone number(s)}%
1467 \hyxmp@declare@field
1468   {CiUrlWork}%
1469   {Contact information Web URL(s)}%
1470 \hyxmp@add@to@xml{%

```

```

1471 -----</rdf:Seq>^^J%
1472 -----</pdfaType:field>^^J%
1473 -----</rdf:li>^^J%
1474 -----</rdf:Seq>^^J%
1475 -----</pdfaSchema:valueType>^^J%
1476 -----</rdf:li>^^J%
1477  }%
1478 }

```

`\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.

```

1479 \newcommand*{\hyxmp@prism@extensions}{%
1480   \hyxmp@begin@ext@decl
1481     {PRISM Basic Metadata}%
1482     {prism}%
1483     {http://prismstandard.org/namespaces/basic/2.1/}%
1484   \hyxmp@declare@property
1485     {complianceProfile}%
1486     {internal}%
1487     {PRISM specification compliance profile to which this document adheres}%
1488   \hyxmp@declare@property
1489     {publicationName}%
1490     {external}%
1491     {Publication name}%
1492   \hyxmp@declare@property
1493     {aggregationType}%
1494     {external}%
1495     {Publication type}%
1496   \hyxmp@declare@property
1497     {bookEdition}%
1498     {external}%
1499     {Edition of the book in which the document was published}%
1500   \hyxmp@declare@property
1501     {volume}%
1502     {external}%
1503     {Publication volume number}%
1504   \hyxmp@declare@property
1505     {number}%
1506     {external}%
1507     {Publication issue number within a volume}%
1508   \hyxmp@declare@property
1509     {pageRange}%
1510     {external}%
1511     {Page range for the document within the print version of its publication}%
1512   \hyxmp@declare@property
1513     {issn}%
1514     {external}%
1515     {ISSN for the printed publication in which the document was published}%

```

```

1516 \hyxmp@declare@property
1517     {eIssn}%
1518     {external}%
1519     {ISSN for the electronic publication in which the document was published}%
1520 \hyxmp@declare@property
1521     {isbn}%
1522     {external}%
1523     {ISBN for the publication in which the document was published}%
1524 \hyxmp@declare@property
1525     {doi}%
1526     {external}%
1527     {Digital Object Identifier for the document}%
1528 \hyxmp@declare@property
1529     [URL]
1530     {url}%
1531     {external}%
1532     {URL at which the document can be found}%
1533 \hyxmp@declare@property
1534     [Integer]
1535     {byteCount}%
1536     {internal}%
1537     {Approximate file size in octets}%
1538 \hyxmp@declare@property
1539     [Integer]
1540     {pageCount}%
1541     {internal}%
1542     {Number of pages in the print version of the document}%
1543 \hyxmp@declare@property
1544     {subtitle}%
1545     {external}%
1546     {Document's subtitle}%
1547 \hyxmp@end@ext@decl
1548 }%

```

`\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.

```

1549 \newcommand*{\hyxmp@declare@extensions}{%

```

```

1550   \hyxmp@begin@extension@decls

```

Declare the Adobe PDF schema (always present).

```

1551   \hyxmp@pdf@extensions

```

Declare the XMP Media Management extensions (always present).

```

1552   \hyxmp@mm@extensions

```

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

```

1553   \ifHy@pdfa

```

```

1554     \hyxmp@pdfa@id@extensions

```

```

1555   \fi

```

Conditionally declare the PDF/UA Universal Accessibility extensions.

```
1556 \ifx\@pdfuapart\@empty
1557 \else
1558 \hyxmp@pdfua@id@extensions
1559 \fi
```

\next Conditionally declare the PDF/X extensions.

```
1560 \ifx\@pdfxversion\@empty
1561 \else
1562 \hyxmp@pdfx@id@extensions
1563 \fi
```

Conditionally declare IPTC photo metadata extensions.

```
1564 \ifx\hyxmp@iptc@data\@empty
1565 \else
1566 \hyxmp@iptc@extensions
1567 \fi
```

Conditionally declare PRISM basic metadata extensions.

```
1568 \ifx\hyxmp@prism@data\@empty
1569 \else
1570 \hyxmp@prism@extensions
1571 \fi
1572 \hyxmp@end@extension@decls
1573 }
```

3.5.12 Combining schemata into an XMP packet

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

```
1574 \begingroup
1575 \ifhyxmp@unicodetex
1576 \lccode'\!="FEFF %
1577 \lowercase{%
1578 \gdef\hyxmp@bom{!}
1579 }%
1580 \else
1581 \catcode'\^^ef=12
1582 \catcode'\^^bb=12
1583 \catcode'\^^bf=12
1584 \gdef\hyxmp@bom{^^ef^^bb^^bf}%
1585 \fi
1586 \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.

```
1587 \def\hyxmp@construct@packet{%
1588 \gdef\hyxmp+xml{%
1589 \hyxmp@add@to+xml{<?xpacket begin="\hyxmp@bom" %
1590 id="W5M0MPCehiHzreSzNTczkc9d"?>^^J%
```

```

1591 <x:xmpmeta xmlns:x="adobe:ns:meta/">^^J%
1592 __<rdf:RDF %
1593 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\hyxmp@hash">^^J%
1594 ____<rdf:Description rdf:about=""^^J%

```

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

```

1595 _____xmlns:pdf="http://ns.adobe.com/pdf/1.3/"^^J%
1596 _____xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/"^^J%
1597 _____xmlns:dc="http://purl.org/dc/elements/1.1/"^^J%
1598 _____xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/"^^J%
1599 _____xmlns:xmp="http://ns.adobe.com/xap/1.0/"^^J%
1600 _____xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"^^J%
1601 _____xmlns:stEvt="http://ns.adobe.com/xap/1.0/sType/ResourceEvent\hyxmp@hash"^^J%
1602 _____xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/"^^J%
1603 _____xmlns:pdfuaid="http://www.aiim.org/pdfua/ns/id/"^^J%
1604 _____xmlns:pdfx="http://ns.adobe.com/pdfx/1.3/"^^J%
1605 _____xmlns:pdfxid="http://www.npes.org/pdfx/ns/id/"^^J%
1606 _____xmlns:prism="http://prismstandard.org/namespaces/basic/2.1/"^^J%
1607 _____xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"^^J%
1608 _____xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"^^J%
1609 _____xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema\hyxmp@hash"^^J%
1610 _____xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property\hyxmp@hash"^^J%
1611 _____xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type\hyxmp@hash"^^J%
1612 _____xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field\hyxmp@hash">^^J%
1613 }%

```

Declare non-standard schemata.

```

1614 \hyxmp@check@iptc@data
1615 \hyxmp@check@prism@data
1616 \hyxmp@declare@extensions

```

Insert all the metadata we know how to insert.

```

1617 \hyxmp@pdf@schema
1618 \hyxmp@xmpRights@schema
1619 \hyxmp@dc@schema
1620 \hyxmp@photoshop@schema
1621 \hyxmp@xmp@basic@schema
1622 \hyxmp@pdfa@id@schema
1623 \hyxmp@pdfua@id@schema
1624 \hyxmp@pdfx@id@schema
1625 \hyxmp@mm@schema
1626 \hyxmp@iptc@schema
1627 \hyxmp@prism@schema
1628 \hyxmp@add@to+xml{%
1629 ____</rdf:Description>^^J%
1630 __</rdf:RDF>^^J%
1631 </x:xmpmeta>^^J%
1632 \hyxmp@padding
1633 <?xpacket end="w"?>^^J%
1634 }%

```

1635 }

3.6 Embedding the XMP packet

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

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

```

1636 \newcommand*{\hyxmp@embed@packet}{%
1637   \hyxmp@construct@packet
1638   \def\hyxmp@driver{hpdfTeX}%
1639   \ifx\hyxmp@driver\Hy@driver
1640     \hyxmp@embed@packet@pdfTeX
1641   \else
1642     \def\hyxmp@driver{hluatex}%
1643     \ifx\hyxmp@driver\Hy@driver
1644       \hyxmp@embed@packet@luatex
1645     \else
1646       \def\hyxmp@driver{hdvipdfm}%
1647       \ifx\hyxmp@driver\Hy@driver
1648         \hyxmp@embed@packet@dvipdfm
1649       \else
1650         \def\hyxmp@driver{hxetex}%
1651         \ifx\hyxmp@driver\Hy@driver
1652           \hyxmp@embed@packet@xetex
1653         \else
1654           \@ifundefined{pdfmark}{%
1655             \PackageWarningNoLine{hyperxmp}{%
1656               Unrecognized hyperref driver ‘\Hy@driver’.\MessageBreak
1657               \jobname.tex’s XMP metadata will *not* be\MessageBreak
1658               embedded in the resulting file}%
1659           }{%
1660             \hyxmp@embed@packet@pdfmark
1661           }%
1662         \fi
1663       \fi
1664     \fi
1665   \fi
1666 }
```

3.6.1 Embedding using pdfTeX

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

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

```
1667 \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.

```
1668 \newcommand*{\hyxmp@embed@packet@pdftex}{%
1669   \bgroup
1670   \ifluatex
1671   \else
1672     \pdfcompresslevel=0
1673   \fi
1674   \immediate\pdfobj \ifluatex uncompressed\fi stream attr {%
1675     /Type /Metadata
1676     /Subtype /XML
1677   }\hyxmp@xml}%
1678   \pdfcatalog {/Metadata \the\pdflastobj\space 0 R}%
1679   \egroup
1680 }
```

3.6.2 Embedding using LuaTeX 0.85+

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

```
1681 \newcommand*{\hyxmp@embed@packet@luatex}{%
1682   \immediate\pdfextension obj uncompressed stream attr {%
1683     /Type /Metadata
1684     /Subtype /XML
1685   }\hyxmp@xml}%
1686   \pdfextension catalog {/Metadata \the\numexpr\pdffeedback lastobj\relax\space 0 R}%
1687 }
```

3.6.3 Embedding using any pdfmark-based backend

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

```
1688 \newcommand*{\hyxmp@embed@packet@pdfmark}{%
1689   \pdfmark{%
1690     pdfmark=/NamespacePush
```



```

1691 }%
1692 \pdfmark{%
1693   pdfmark=/OBJ,
1694   Raw={/_objdef \string{hyxmp@Metadata\string} /type /stream}%
1695 }%
1696 \pdfmark{%
1697   pdfmark=/PUT,
1698   Raw={\string{hyxmp@Metadata\string}
1699     2 dict begin
1700       /Type /Metadata def
1701       /Subtype /XML def
1702       currentdict
1703     end
1704   }%
1705 }%
1706 \pdfmark{%
1707   pdfmark=/PUT,
1708   Raw={\string{hyxmp@Metadata\string} (\hyxmp@xml)}%
1709 }%
1710 \pdfmark{%
1711   pdfmark=/Metadata,
1712   Raw={\string{Catalog\string} \string{hyxmp@Metadata\string}}%
1713 }%
1714 \pdfmark{%
1715   pdfmark=/NamespacePop
1716 }%
1717 }

```

3.6.4 Embedding using dvipdfm

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

```

1718 \newcommand*{\hyxmp@embed@packet@dvipdfm}{%
1719   \hyxmp@string@len{\hyxmp@xml}%
1720   \special{pdf: object @hyxmp@Metadata
1721     <<
1722       /Type /Metadata
1723       /Subtype /XML
1724       /Length \the\@tempcnta
1725     >>
1726     stream^^J\hyxmp@xml endstream%
1727   }%
1728   \special{pdf: docview
1729     <<
1730       /Metadata @hyxmp@Metadata
1731     >>
1732   }%
1733 }

```

`\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.

```
1734 \newcommand*{\hyxmp@string@len}[1]{%
1735   \@tempcnta=0
1736   \expandafter\hyxmp@count@spaces#1 {} %
1737   \expandafter\hyxmp@count@non@spaces#1}%
1738 }
```

`\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.

```
1739 \def\hyxmp@count@spaces#1 {%
1740   \def\hyxmp@one@token{#1}%
1741   \ifx\hyxmp@one@token\@empty
1742     \advance\@tempcnta by -1
1743   \else
1744     \advance\@tempcnta by 1
1745     \expandafter\hyxmp@count@spaces
1746   \fi
1747 }
```

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

```
1748 \newcommand*{\hyxmp@count@non@spaces}[1]{%
1749   \def\hyxmp@one@token{#1}%
1750   \ifx\hyxmp@one@token\@empty
1751   \else
1752     \advance\@tempcnta by 1
1753     \expandafter\hyxmp@count@non@spaces
1754   \fi
1755 }
```

3.6.5 Embedding using X_qT_EX

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

```
1756 \newcommand*{\hyxmp@embed@packet@xetex}{%
1757   \special{pdf:stream @hyxmp@Metadata (\hyxmp@xml)}
1758   <<
1759     /Type /Metadata
1760     /Subtype /XML
1761   >>
1762 }
```

```

1763 \special{pdf:put @catalog
1764     <<
1765         /Metadata @hyxmp@Metadata
1766     >>
1767 }%
1768 }

```

3.7 Final clean-up

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

```

1769 \catcode'\=\hyxmp@dq@code

```

4 Help Wanted

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

A Sample XMP Packet

The following is an example of a complete XMP packet as may be produced by `hyperxmp`. This packet corresponds to the metadata included in the sample \LaTeX document presented on pages 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>
<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: Pre-defined XMP Properties in PDF/A-1*, March 20, 2008. Available from http://www.pdfa.org/wp-content/uploads/2011/08/tn0008_predefined_xmp_properties_in_pdfa-1_2008-03-20.pdf.
- [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	which enables an author to specify the language in which he wrote the document's metadata	30
v1.1	\hyxmp@construct@packet:			
	Explicitly set the category codes of characters $\langle EF \rangle$, $\langle BB \rangle$, and $\langle BF \rangle$ to “letter”. Thanks to Daniel Schömer for the bug report	69	v1.4	
			\hyxmp@mm@schema: Renamed the xapMM namespace prefix to xmpMM	57
			\hyxmp@rdf@dc: Included metadata in the x-default language regardless of the specified metadata language	53
v1.2	General: Added support for the X _Y TeX backend (xdvipdfmx) . .	1	\hyxmp@xmpRights@schema: Renamed the xapRights namespace prefix to xmpRights	56
	Added support for the Photoshop schema	1		
	Made the package compatible with ngerman. Thanks to Tobias Mueller for the bug report.	17	v1.5	
			General: Made the XMP inclusion more robust. Thanks to Heiko Oberdiek for the bug report and suggested modifications. .	17
v1.3	General: Introduced the pdfmetalang package option,		v2.0	
			\ProcessKeyvalOptions: Added	

this macro	27	\hyxmp@xmp@basic@schema: Added	
\XMPTruncateList: Added this		this macro	57
macro	33	\hyxmp@xmpRights@schema:	
\hyxmp@ProcessKeyvalOptions:		Modified to include	
Added this macro	27	xmpRights:Marked only when	
\hyxmp@SpaceOther: Added by		pdfcopyright is specified and	
Heiko Oberdiek	43	xmpRights:WebStatement only	
\hyxmp@add@simple: Added this		when pdflicenseurl is specified .	56
macro	45	\hyxmp@zero: Added by Heiko	
\hyxmp@add@to@xml: Updated also		Oberdiek	44
to replace commas	50	\ifhyxmp@unicodetex: Added by	
\hyxmp@bom: Added by Heiko		Heiko Oberdiek	39
Oberdiek	69	\xmpcomma: Added this macro ...	32
\hyxmp@comma: Added this macro	32	\xmpquote: Added this macro ...	32
\hyxmp@construct@packet:		General: Added support for the	
Modified by Heiko Oberdiek to		XMP Basic schema and	
use an appropriate BOM		miscellaneous other bits of	
representation via \hyxmp@bom	69	metadata	1
\hyxmp@crap@convert: Added by		Heiko Oberdiek's major rewrite	
Heiko Oberdiek	44	of the code to better support	
\hyxmp@crap@test: Added by		native-Unicode T _E X	
Heiko Oberdiek	43	implementations (X _Y T _E X and	
\hyxmp@dc@schema: Added support		LuaT _E X)	1
for dc:language and dc:source .	55	New \AtBeginDocument code	
\hyxmp@is@unicode: Added by		from Heiko Oberdiek to	
Heiko Oberdiek	41	properly encode	
\hyxmp@list@to@xml: Modified by		\@pdfmetalang	30
Heiko Oberdiek to use the new			
Unicode-processing macros ..	54	v2.1	
\hyxmp@photoshop@schema:		\hypersetup: Added this macro .	27
Simplified using		\hyxmp@hypersetup: Added this	
\hyxmp@add@simple	58	macro	27
\hyxmp@skiptorelax: Added by		\hyxmp@redefine@Hyp: Added this	
Heiko Oberdiek	43	macro	26
\hyxmp@skipzeros: Added by		General: Enabled hyperxmp and	
Heiko Oberdiek	42	hyperref to be loaded in either	
\hyxmp@toxml: Added by Heiko		order. This addresses a bug	
Oberdiek	41	report by Yury Donskoy	25
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	41	generation	65
\hyxmp@toxml@unicodetex: Added		\hyxmp@iptc@schema: Added this	
by Heiko Oberdiek	42	macro	60
\hyxmp@xetex@crap: Added by		\hyxmp@list@to@lines: Added	
Heiko Oberdiek	42	this macro	59
\hyxmp@xmlify: Completely		\xmpcomma: Changed the default	
rewritten by Heiko Oberdiek to		from \relax to an ordinary	
better support Unicode-enabled		comma	32
T _E X programs	39	\xmplinesep: Added this macro .	59

General: Added support for the IPTC Photo Metadata schema . . .	1	\hyxmp@today@xmp: Modified the code to parse the time and timezone from \pdfcreationdate, as proposed by Florian Breitwieser	37
v2.3		\hyxmp@iptc@extensions: Gave the lptc4xmpCore:CreatorContactInfo fields a unique pdfaType:prefix to better support conversion of the document to PDF/A	65
v2.3a		General: Bug fix: Redefine \@pdflang as \@empty when hyperref has set it to \relax . . .	29
v2.3b		\XMPTruncateList: Made all definitions local to avoid spurious Too many unprocessed floats errors when running with memoir . . .	33
v2.4		\hyxmp@add@simple@var: Added this macro	45
		\hyxmp@create@uuid: Modified this macro to produce a proper version 4 (random or pseudorandom) UUID	49
		\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	55
		\hyxmp@parse@time: Added this macro	34
		\hyxmp@parse@tz: Added this macro	34
		\hyxmp@parse@tz@char: Added this macro	34
		\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	53
		\hyxmp@pdf@to@xmp@date: Added this macro	34
		\hyxmp@pdfa@id@schema: Added this macro	58
	v2.5	\hyxmp@add@to@xml: Updated also to replace underscores and to modify only the text being added, not the already-modified text	50
		\hyxmp@textunderscore: Added this macro	18
		\hyxmp@uscore: Added this macro . . .	32
		General: Enabled “_” to work within email addresses, as requested by Leonid Sinev	1
	v2.6	General: Added support for a new pdfdate key to explicitly specify the document date (and optionally time)	1
	v2.7	General: Automatically use \title and \author if pdftitle and pdfauthor are left unspecified. Thanks to Maciej Radziejewski for the suggestion	30
	v2.8	\hyxmp@add@to@xml: Corrected inadvertent lowercasing of non-Latin characters when run under Xe ^L A ^T E _X or Lua ^L A ^T E _X (bug reported by Leonid Sinev) . . .	50
	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 <code>lptc4xmpCore</code> .	60	<code>\hyxmp@today@xmp@define:</code> Modified to include hours and minutes	36
<code>\hyxmp@pdfa@id@schema:</code> Let the author specify the PDF/A part and conformance IDs, as requested by Leonid Sinev . . .	58	<code>\hyxmp@xmp@basic@schema:</code> Honor <code>hyperref</code> 's <code>pdfcreationdate</code> and <code>pdfmoddate</code> options plus a new <code>pdfmetadate</code> option. Leonid Sinev requested this additional control and helped test the resulting <code>hyperxmp</code> code	57
General: Force inclusion of <code>dc:creator</code> , <code>dc:title</code> , and <code>dc:description</code> —even if empty—when <code>hyperref</code> is loaded with the <code>pdfa</code> option (suggested by Leonid Sinev)	1	v3.3 <code>\@pdfsource:</code> Added this macro and the corresponding <code>pdfsource</code> option, at Niklas Beisert's request	21
Introduced the <code>pdftype</code> package option, which enables an author to specify the type of document being produced	1	<code>\XMPLangAlt:</code> Added this macro based on a request—and some code—by Niklas Beisert to support metadata expressed in multiple languages	47
v3.0 <code>\hyxmp@embed@packet@luatex:</code> Added this macro	72	<code>\hyxmp@rdf@dc:</code> Bug fix: Output the metadata language as correct XML even when <code>hyperref</code> is loaded with the <code>unicode</code> option	53
<code>\hyxmp@today@xmp@define:</code> Modified to accept the name of a macro to define	36	General: Don't overwrite an existing <code>pdfmetalang</code> with <code>pdflang</code> or <code>x-default</code> . This addresses a bug report by Niklas Beisert	30
<code>\hyxmp@xmp@basic@schema:</code> Made the XMP <code>xmp:CreateDate</code> , <code>xmp:ModifyDate</code> , and <code>xmp:MetadataDate</code> match the PDF <code>CreationDate</code>	57	v3.4 <code>\hyxmp@seed@string:</code> Correctly handle an author field of all spaces. Bug reported by Gaëtan Leurent	49
General: Made the code compatible with Lua _T _E X 0.85. Thanks to Robert Schlicht, Leonid Sinev, and David Carlisle for bug reports and to Leonid Sinev for helping test the new <code>hyperxmp</code> code	1	General: Use <code>ifmtarg</code> to test for empty arguments, including non-empty but all spaces	1
v3.1 <code>\hyxmp@embed@packet@luatex:</code> Updated to use <code>\pdfextension</code> <code>obj uncompressed</code> as suggested by Hans Hagen	72	v3.5 <code>\hyxmp@DocumentID:</code> Added the <code>pdfdocumentid</code> option, at Michael Osipov's request	21
<code>\hyxmp@embed@packet@pdftex:</code> Leave the XMP packet—and only the XMP packet—uncompressed in both <code>pdf_T_EX</code> and pre-0.85 Lua _T _E X . . .	72	<code>\hyxmp@InstanceID:</code> Added the <code>pdfinstanceid</code> option, at Michael Osipov's request	22
v3.2 <code>\hyxmp@as@pdf@date:</code> Added this macro	34	<code>\hyxmp@mm@schema:</code> Generate <code>\hyxmp@DocumentID</code> and <code>\hyxmp@InstanceID</code> only if the document does not already define these using the	
<code>\hyxmp@as@xmp@date:</code> Added this macro	33		

pdfdocumentid and pdfinstanceid options	57	\hyxmp@prism@schema: Added this macro	60
\hyxmp@seed@string: Seed with the T _E X timestamp in addition to the document-specified timestamp	50	General: Include all metadata within a single rdf:Description block	1
v4.0		v4.1	
\XMPTruncateList: Deprecated this macro	33	\hyxmp@singleton@dc: Added this macro	55
\hyxmp@add@simple@lang: Added this macro	45	General: Invoke \hyxmp@no@info@lists at the beginning of the document, for compatibility with both newer and older versions of hyperref .	31
\hyxmp@begin@ext@decl: Added this macro	62	Updated the documentation to refer to \pdfnumpages by its correct name. Thanks to Volker RW Schaa for catching the discrepancy	1
\hyxmp@declare@field: Replaced \hyxmp@declare@resource with this macro	63		
\hyxmp@declare@property: Added this macro	63	v5.0	
\hyxmp@end@ext@decl: Added this macro	62	\@pdfrendition: Added the pdfrendition option	22
\hyxmp@iptc@extensions: Moved the header code from here into \hyxmp@begin@extension@decls and the trailer code from here into \hyxmp@end@extension@decls	65	\@pdfxstandard: Added this macro	21
Rewrote to more closely honor the XMP specification	65	\hyxmp@add@simple: Insert the tag name (#1) verbatim	45
\hyxmp@iptc@schema: Moved the definition of \hyxmp@iptc@data from here into \hyxmp@check@iptc@data . . .	60	\hyxmp@check@standards: Added this macro	28
Renamed this macro to \hyxmp@iptc@schema from \hyxmp@photometa@schema . .	60	\hyxmp@check@std: Added this macro	21
Rewrote this macro entirely to correct the use of fields within a structure	60	\hyxmp@declare@property: Insert the property name (#2) verbatim	63
\hyxmp@mm@extensions: Added this macro	63	\hyxmp@define@pdfproducer: Added this macro	52
\hyxmp@mm@schema: Include xmpMM:VersionID in the XMP packet	57	\hyxmp@no@info@lists: Renamed this macros from \hyxmp@suppress@pdf@metadata and rewrote it to replace, if possible, only Author and Keywords	24
\hyxmp@no@info@lists: Added this macro	24	\hyxmp@pdf@extensions: Added this macro	63
\hyxmp@pdfa@id@extensions: Added this macro	64	\hyxmp@pdf@schema: Honor pdftrapped	53
\hyxmp@prism@extensions: Added this macro	67	\hyxmp@pdfua@id@extensions: Added this macro	64
		\hyxmp@pdfua@id@schema: Added this macro	58
		\hyxmp@pdfx@id@extensions: Added this macro	65

<code>\hyxmp@pdfx@id@schema</code> : Added this macro	58	“@” from propagating past the <code>\begin{document}</code> . Thanks to Robert Schlicht for noticing this catcode “leak” and providing a correction	52
<code>\hyxmp@today@pdf</code> : Added this macro	38		
<code>\hyxmp@today@xmp</code> : Support X _Y TeX’s <code>\filemoddate</code>	37		
<code>\hyxmp@today@xmp@define</code> : Modified to specify UTC	36	<code>\hyxmp@define@pdfproducer</code> : Check for LuaTeX before checking for pdfTeX to work around <code>luatex85</code> ’s confusing <code>iftex</code> by defining <code>\pdftexversion</code> . Thanks to Robin Schwab for the bug report	52
General: Added support for PDF/UA standards, as requested by Robin Schwab	1		
Added support for PDF/X standards, as requested by Robin Schwab	1		
Define a default producer	52	<code>\hyxmp@timestamp</code> : Don’t rely on <code>\jobname.aux</code> existing to query the current time under X _Y TeX. Instead, use <code>\jobname.log</code> . Thanks to Ulrike Fischer for the bug report and for her suggestion to use the log file. .	38
Don’t set any document dates (creation, modification, or metadata) from <code>pdfdate</code>	30		
v5.1 <code>\hyxmp@banner@to@producer</code> : Prevent the category code of			

Index

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

Symbols	345, 351, 779, 1184	\@pdfcontactcountry
\# 955, 1180	\@pdfaformance 170, 266, 1216, 1253	
\& 643, 675, 1179 70, 314, 1156	\@pdfcontactemail
\@author 351, 352	\@pdfapart 65, 174, 267, 1219, 1255	
\@baseurl 258, 1146	307, 313, 318, 1155	\@pdfcontactphone
\@elt 406, 1057, 1191, 1196	\@pdfauthor 189, 213, 172, 268, 1218, 1254	
\@elt@first 1189	259, 350, 920, 928	\@pdfcontactpostcode
\@elt@rest 1191, 1193	\@pdfauthortitle 168, 269, 1215, 1252	
\@ifmtarg 18, 808	54, 260, 1149, 1150	\@pdfcontactregion
\@ifmtargexp 18, 307, 323, 344, 350, 357, 545, 1012, 1047, 1120, 1121, 1128, 1134, 1140	\@pdfbookedition 166, 270, 1214, 1251	
	140, 261, 1235, 1261	\@pdfcontacturl
	\@pdfbytes 176, 271, 1220, 1256	
	130, 262, 1244, 1262	\@pdfcopyright
	\@pdfcaptionwriter 48, 272, 1088, 1106, 1112	
\@ifnextchar 803	56, 263, 1149, 1151	
\@ifnotmtarg 19, 61, 797, 806, 1073	\@pdfcontactaddress	\@pdfcreationdate
	156, 264, 1212, 1249 273,
\@ifnotmtargexp 18, 322, 164, 265, 1213, 1250	\@pdfcontactcity 357, 358, 1128, 1132	
	\@pdfcreator 1145	

\@pdfdatetime	.. 26, 274	\@pdfvolumenum 144, 294, 1236, 1274	dc:format 2
\@pdfdoi 150, 275, 1242, 1263	\@pdfxstandard 92, 295, 319, 1168, 1171, 1173	dc:language	. 3, 55, 81, 82
\@pdfeissn 136, 276, 1241, 1264	\@pdfxversion 1560	dc:publisher 3
\@pdfisbn 138, 277, 1239, 1265	\@tempswafalse	1012, 1047	dc:rights	... 2, 15, 46, 55
\@pdfissn 134, 278, 1240, 1266	\@tempswatru	. 1012, 1014, 1047, 1049	dc:source 2, 55, 81
\@pdfissuenum 146, 279, 1237, 1267	\@title 345, 346	dc:subject 2, 55
\@pdfkeywords 195, 234, 280	\^ 394, 398, 580, 948, 949, 1581–1583	dc:title	.. 3, 10, 46, 55, 83
\@pdflang	. 281, 333, 334, 337, 340, 1091	_ 947, 949	dc:type 3
\@pdflicenseurl	... 52, 282, 1102, 1116	\~ 403, 713, 714	DCMI 7
\@pdfmetadatatetime	.. 37, 283, 1140, 1143	_ 679, 714, 947	\define@key 27, 38, 49, 51, 53, 55, 57, 59, 66, 71, 75, 94, 112, 114, 116, 118, 125, 127, 129, 131, 133, 135, 137, 139, 141, 143, 145, 147, 149, 151, 153, 155, 157, 165, 167, 169, 171, 173, 175, 177, 201, 213, 234, 823
\@pdfmetalang 58, 336, 338, 340, 343, 1023, 1025	A		\do 825, 832, 1033
\@pdfmoddate 284, 1134, 1138	\and 213	DOI 2, 6, 23
\@pdfnumpages 132, 285, 1245, 1268	ASCII 18, 40	draft (option) 8
\@pdfpagerange 148, 286, 1238, 1269	\AtBeginDocument	331, 993	dvipdf (option) 72
\@pdfproducer 971, 985, 992, 994	\AtEndDocument 5	dvipdfm 73
\@pdfpublication	.. 126, 287, 1233, 1270	\AtEndDvi 8	dvips (option) 72
\@pdfpublisher	142, 1089	atenddvi 17	dvips 10, 41, 81
\@pdfpubtype 128, 288, 1234, 1271	Author 10, 24, 84	dvipsone (option) 72
\@pdfrendition	120, 1125	B		dviwindo (option)	... 72
\@pdfsource 111, 1095, 1097	Bag 82	E	
\@pdfsubject	. 289, 1087	baseurl (option) 4, 6, 14, 17, 23, 57	ϵ -TeX 52
\@pdfsubtitle 154, 290, 1232, 1272	BOM 69, 81	\EdefEscapeHex	602, 615
\@pdftitle	. 291, 323, 344, 920, 928, 1086	C		\EdefUnescapeHex	.. 619
\@pdftrapped 1001	CiAdrCity 2	\EdefUnescapeString	589
\@pdftype 50, 1092	CiAdrCtry 2	\empty 1392
\@pdfuapart 74, 292, 320, 1160, 1556	CiAdrExtadr 2	\equal 88
\@pdfurl 152, 293, 1243, 1273	CiAdrPcode 2	etoolbox 18
\@pdfversionid	117, 1124	CiAdrRegion 2	ETX 32, 39
		CiEmailWork 2	F	
		CiTelWork 3	\filemoddate 557
		CreationDate 30, 83	G	
		D		Ghostsript 10
		Date 37	gitver 6
		\day 523, 524, 526		
		dc:creator	.. 2, 10, 55, 83		
		dc:date 2, 55		
		dc:description 3, 10, 46, 55, 83		

H		
\Hy@driver	\hyxmp@alt@rights	\hyxmp@count@spaces
. 4, 1639, 1643, <u>819</u> , 830 1736, <u>1739</u>
1647, 1651, 1656	\hyxmp@alt@title <u>819</u> , 828	\hyxmp@crap@convert
\Hy@unicodedefalse 29, 40	\hyxmp@and <u>213</u> 705, <u>739</u>
hyperref 1, 4–	\hyxmp@append@hex	\hyxmp@crap@result
8, 10, 14, 17, 18,	. 868, 887–889, 893 695, 731
20, 24–27, 29–31,	\hyxmp@append@hex@iii	\hyxmp@crap@test 702, <u>727</u>
51–53, 71, 72, 81–84	. 886, 892, 902, 913	\hyxmp@create@uuid
\hypersetup <u>251</u> , 346, 352	\hyxmp@append@hex@iv <u>895</u> , 923, 932
hyperxmp . 1, 2, 4–10, <u>891</u> , 897,	\hyxmp@cur@lang 825, <u>833</u>
12–19, 21, 24, 25,	898, 900, 915–917	\hyxmp@dc@schema
29–33, 39, 47, 51,	\hyxmp@as@pdf@date . <u>450</u> <u>1084</u> , 1619
53, 61, 71, 75, 81, 83	\hyxmp@as@xmp@date	\hyxmp@declare@extensions
\hyxmp@is@unicode . <u>623</u> 32, 43, <u>1549</u> , 1616
\hyxmp@add@simple	<u>422</u> , 559, 1132, 1138	\hyxmp@declare@field
<u>778</u> , 1001, 1005,	\hyxmp@at@end . . . 3, 364 <u>1318</u> ,
1008, 1085, 1097,	\hyxmp@banner@to@producer	1446, 1449, 1452,
1114, 1116, 1122– 974, 977, <u>985</u>	1455, 1458,
1125, 1129, 1131,	\hyxmp@begin@ext@decl	1461, 1464, 1467
1135, 1137, 1141, <u>1289</u> ,	\hyxmp@declare@property
1143, 1145, 1146,	1328, 1339, 1364, <u>1306</u> ,
1150, 1151, 1155,	1380, 1394,	1332, 1343, 1348,
1156, 1160, 1167,	1410, 1422, 1480	1353, 1357, 1368,
1168, 1171, 1173,	\hyxmp@begin@extension@decls	1373, 1384, 1398,
1213–1216, 1230, <u>1277</u> , 1550	1402, 1414, 1426,
1234, 1236–1245	\hyxmp@big@prime	1484, 1488, 1492,
\hyxmp@add@simple@lang 842, 845, 855, 865	1496, 1500, 1504,
. <u>796</u> ,	\hyxmp@big@prime@ii	1508, 1512, 1516,
1232, 1233, 1235 <u>842</u> , 864	1520, 1524, 1528,
\hyxmp@add@simple@lang@i	\hyxmp@bom . . <u>1574</u> , 1589	1533, 1538, 1543
. 799, <u>802</u>	\hyxmp@check@iptc@data	\hyxmp@def@DocumentID
\hyxmp@add@simple@lang@ii <u>1247</u> , 1614 <u>919</u> , 1120
. 803, <u>805</u>	\hyxmp@check@prism@data	\hyxmp@def@InstanceID
\hyxmp@add@simple@var <u>1259</u> , 1615 <u>925</u> , 1121
. . . <u>787</u> , 999, 1000	\hyxmp@check@standards	\hyxmp@define@pdfproducer
\hyxmp@add@to@xml <u>305</u> , 362 <u>971</u> , 995
. 781, 783, 791,	\hyxmp@check@std	\hyxmp@DocumentID
809, 813, <u>934</u> , <u>87</u> , 99–107 <u>113</u> , <u>919</u> , 1120, 1122
960, 1019, 1026,	\hyxmp@comma	\hyxmp@dq@code . 1, 1769
1030, 1035, 1040,	. 158, 214, 235, <u>393</u>	\hyxmp@driver . . 3, <u>1636</u>
1052, 1060, 1066,	\hyxmp@commas@to@list	\hyxmp@embed@packet
1075, 1186, 1190,	<u>377</u> , 413, 1058, 1198 366, <u>1636</u>
1194, 1200, 1207,	\hyxmp@commas@to@list@i	\hyxmp@embed@packet@dvi
1222, 1278, 1284, 379, <u>381</u> 1648, <u>1718</u>
1290, 1300, 1307,	\hyxmp@concat@metadata	\hyxmp@embed@packet@luatex
1311, 1319, 1431, <u>256</u> 1644, <u>1681</u>
1470, 1589, 1628	\hyxmp@construct@packet	\hyxmp@embed@packet@pdfmark
\hyxmp@alt@description <u>1587</u> , 1637 1660, <u>1688</u>
. <u>819</u> , 829	\hyxmp@count@non@spaces	\hyxmp@embed@packet@pdftex
 1737, <u>1748</u> 1640, <u>1668</u>

\hyxmp@embed@packet@xetex	\hyxmp@new@xml 950, 951	\hyxmp@pdfx@id@schema
..... 1652, <u>1756</u>	\hyxmp@no@bad@parts <u>1162</u> , 1624
\hyxmp@end@ext@decl 60, 67, 76	\hyxmp@pdfx@major ..
... <u>1299</u> , 1336,	\hyxmp@no@info@lists	... <u>84</u> , 93, <u>109</u> ,
1361, 1377, 1389, <u>178</u> , 202, 363	1163, 1392, 1408
1406, 1418, 1547	\hyxmp@num <u>739</u>	\hyxmp@photoshop@data
\hyxmp@end@extension@decls	\hyxmp@one@token <u>1148</u>
..... <u>1283</u> , 1572	844, 848, 1740,	\hyxmp@photoshop@schema
\hyxmp@extra@indent	1741, 1749, 1750 <u>1148</u> , 1620
..... <u>777</u> ,	\hyxmp@padding <u>958</u> , 1632	\hyxmp@prism@data ..
781, 792, 1187, 1211	\hyxmp@parse@time 1228, <u>1260</u> , 1568
\hyxmp@find@metadata 431, <u>433</u>	\hyxmp@prism@extensions
..... <u>256</u> , 365	\hyxmp@parse@tz <u>1479</u> , 1570
\hyxmp@first@char .. <u>420</u> 440, 443, <u>447</u>	\hyxmp@prism@schema
\hyxmp@first@char@i	\hyxmp@parse@tz@char <u>1227</u> , 1627
.... <u>420</u> , 423, 451 435, <u>437</u>	\hyxmp@ProcessKeyvalOptions
\hyxmp@gobbletwo <u>490</u> , 503	\hyxmp@pdf@extensions <u>246</u>
\hyxmp@hash <u>954</u> , 1593, <u>1327</u> , 1551	\hyxmp@rand@num <u>861</u> ,
1601, 1609–1612	\hyxmp@pdf@schema ..	870, 905, 922, 931
\hyxmp@Hyp@pdfauthor <u>998</u> , 1617	\hyxmp@rdf@dc
..... <u>207</u>	\hyxmp@pdf@to@xmp@date	.. <u>1011</u> , 1086–1088
\hyxmp@Hyp@pdfkeywords	. 424, <u>429</u> , 551, 554	\hyxmp@redefine@Hyp
..... <u>228</u>	\hyxmp@pdfa@id@extensions <u>206</u> , 248, 253
\hyxmp@hypersetup .. <u>251</u> <u>1363</u> , 1554	\hyxmp@remove@this ..
\hyxmp@InstanceID ..	\hyxmp@pdfa@id@schema 989, <u>992</u>
<u>115</u> , <u>925</u> , 1121, 1123 <u>1153</u> , 1622	\hyxmp@rights . 1101,
\hyxmp@iptc@data ..	\hyxmp@pdfauthor ..	1104, 1108, 1110
.. 1205, <u>1248</u> , 1564	... <u>204</u> , <u>213</u> , 1093	\hyxmp@seed@rng ...
\hyxmp@iptc@extensions	\hyxmp@pdfkeywords <u>844</u> , 921, 930
..... <u>1421</u> , 1566	... <u>204</u> , <u>234</u> , 1094	\hyxmp@seed@rng@i ..
\hyxmp@iptc@schema .	\hyxmp@pdfstringdef 846, <u>848</u>
..... <u>1204</u> , 1626 <u>20</u> ,	\hyxmp@seed@string .
\hyxmp@is@unicode ..	31, 42, 49, 51, 53, <u>919</u> , <u>925</u>
.... 591, 608, <u>623</u>	55, 57, 59, 68,	\hyxmp@set@pdfx@major
\hyxmp@LA@accept ..	72, 77, 95, 112, <u>79</u> , 109
.... <u>822</u> , 828–830	114, 116, 118,	\hyxmp@set@pdfx@major@i
\hyxmp@legal <u>1101</u>	125, 127, 129, <u>79</u> , <u>80</u>
\hyxmp@list ... 1058,	131, 133, 135,	\hyxmp@set@pdfx@major@ii
1064, 1198, 1199	137, 139, 141, <u>81</u> , <u>84</u>
\hyxmp@list@to@lines	143, 145, 147,	\hyxmp@set@rand@num
..... <u>1183</u> ,	149, 151, 153, <u>861</u> , 869, 904
1212, 1218–1220	155, 160, 165,	\hyxmp@singleton@dc
\hyxmp@list@to@xml .	167, 169, 171,	.. <u>1072</u> , 1089–1092
.. <u>1046</u> , 1093, 1094	173, 175, 177, 824	\hyxmp@skiptorelax ..
\hyxmp@mm@extensions	\hyxmp@pdfua@id@extensions <u>732</u> , <u>738</u>
..... <u>1338</u> , 1552 <u>1379</u> , 1558	\hyxmp@skipzeros .. <u>690</u>
\hyxmp@mm@schema ..	\hyxmp@pdfua@id@schema	\hyxmp@SpaceOther ..
..... <u>1119</u> , 1625 <u>1159</u> , 1623 699, <u>712</u>
\hyxmp@modulo@a <u>836</u> ,	\hyxmp@pdfx@id@extensions <u>317</u>
855, 865, 871, 906 <u>1391</u> , 1562	

<code>\hyxmp@string@len</code> ..	<code>\hyxmp@xmlify</code>	Info
..... 1719, <u>1734</u>	. 343, <u>587</u> , 780,	12, 24, 25, 30, 31, 52
<code>\hyxmp@sublist</code>	790, 798, 807,	intcalc
. 382, 383, 386, 387	1017, 1025, 1034,	\intcalcDiv 744, 751, 758
<code>\hyxmp@suppress@pdf@info</code>	1057, 1074, 1197	\intcalcMod 746, 753, 760
..... <u>179</u>	<code>\hyxmp@xmp@basic@schema</code>	IPTC
<code>\hyxmp@temp@list</code> .. <u>406</u> <u>1127</u> , 1621	50, 60, 61, 65, 69, 82
<code>\hyxmp@temp@str</code> ... <u>406</u>	<code>\hyxmp@xmp@to@pdf@date</code>	lptc4xmpCore:ContactInfo
<code>\hyxmp@text</code> 454, <u>457</u> , 567 60, 66
. <u>587</u> , <u>665</u> , <u>695</u> , <u>739</u>	<code>\hyxmp@xmp@to@pdf@date@i</code>	lptc4xmpCore:CreatorContactInfo
<code>\hyxmp@textunderscore</code> 458, <u>460</u> 2, 3, 60, 82
..... <u>20</u>	<code>\hyxmp@xmp@to@pdf@date@ii</code>	ISBN
<code>\hyxmp@timestamp</code> .. <u>556</u> 463, <u>466</u>	2, 6, 23
<code>\hyxmp@today@pdf</code> 358, <u>566</u>	<code>\hyxmp@xmp@to@pdf@date@iii</code>	ISO
<code>\hyxmp@today@xmp</code> 469, <u>472</u>	5, 6, 15, 19, 46
..... 545, <u>549</u> ,	<code>\hyxmp@xmp@to@pdf@date@iv</code>	ISSN
567, 928, 1090, 475, <u>478</u>	2, 6, 22
1129, 1135, 1141	<code>\hyxmp@xmp@to@pdf@date@v</code>	
<code>\hyxmp@today@xmp@define</code> 481, <u>484</u>	J
.... <u>516</u> , 564, 926	<code>\hyxmp@xmp@to@pdf@date@vi</code>	\jobname 111, 299, 370,
<code>\hyxmp@toxml</code> .. 617, <u>640</u> 487, <u>491</u>	557, 920, 928, 1657
<code>\hyxmp@toxml@unicodetex</code>	<code>\hyxmp@xmp@to@pdf@date@vii</code>	K
..... 605, <u>665</u> 494, 497, <u>507</u>	keeppdfinfo (option) .
<code>\hyxmp@trimb</code> .. 573, <u>576</u>	<code>\hyxmp@xmp@to@pdf@date@viii</code> 8, 12, 24
<code>\hyxmp@trimc</code> .. 576, <u>577</u> 510, <u>513</u>	Keywords .. 10, 24, 53, 84
<code>\hyxmp@trimspaces</code> ..	<code>\hyxmp@xmpRights@schema</code>	\KV@Hyp@pdfauthor .. <u>213</u>
..... 386, <u>569</u> <u>1100</u> , 1618	\KV@Hyp@pdfkeywords <u>234</u>
<code>\hyxmp@try</code>	<code>\hyxmp@zero</code> 748,	kvoptions
<code>\hyxmp@try@today</code> <u>544</u> ,	755, 762, 768, <u>773</u>	L
550, 553, 556, 563		LF
<code>\hyxmp@unicodetexfalse</code>	I	Lua ^A TeX 10, 13, 37, 53, 82
..... <u>579</u>	IETF	LuaTeX
<code>\hyxmp@unicodetexttrue</code>	\if@tempswa .. 1016, 1051	71, 72, 75, 81, 83, 85
..... <u>579</u>	\ifdraft	luatex85
<code>\hyxmp@uscore</code> .. 22, <u>397</u>	\iffalse ... 1011, 1046	\luatexbanner
<code>\hyxmp@value</code> . <u>824</u> , <u>1018</u>	\ifHy@pdfa	M
<code>\hyxmp@x@default</code> ..	306, 1086, 1087,	\makeatletter
338, <u>970</u> , 1023, 1031	1093, 1154, 1553	memoir
<code>\hyxmp@xetex@crap</code> ..	<code>\ifhyxmp@unicodetex</code>	Metadata 10, 71, 74
..... 596, <u>695</u> <u>579</u> , 590, 937, 1575	\month 518, 519, 521
<code>\hyxmp@xml</code>	<code>\ifLuaTeX</code>	N
. 782, 784, 951,	ifluatex	NAK
<u>958</u> , 1310, <u>1587</u> ,	\ifluatex .. 1670, 1674	nativepdf (option) ... 72
1677, 1685, 1708,	ifmtarg	\newif
1719, 1726, 1757	\ifPDFTeX	\next
<code>\hyxmp@xmlified</code> ...	iftex	89, <u>96</u> , <u>179</u> , <u>381</u> ,
. <u>587</u> , 783, 792,	ifthen	558, 561, <u>848</u> , <u>1560</u>
799, 810, 814,	\ifthenelse	ngerman
1018, 1027, 1036,	\ifXeTeX	\number 742, 744, 746,
1058, 1078, 1198		751, 753, 758, 760

\numexpr 1686

O

options

baseurl
4, 6, 14, 17, 23, 57
draft 8
dvipdf 72
dvips 72
dvipsone 72
dviwindo 72
keeppdfinfo . . 8, 12, 24
nativepdf 72
pdfa . . . 8, 20, 29, 83
pdfaconformance .
. 4, 8, 58
pdfapart . 4, 8, 20, 58
pdfauthor 4,
5, 10, 13, 14, 17,
25, 26, 30, 55, 82
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, 55, 56, 81
pdfcreationdate . .
. 4, 7, 30, 83
pdfdate 4,
7, 14, 18, 50, 82, 85
pdfdocumentid . . .
. 4, 6, 83, 84
pdfdoi 4, 6
pdfeissn 4, 6
pdfinstanceid
. 4, 6, 83, 84
pdfisbn 4, 6
pdfissn 4, 6
pdfissuenum 4, 7

pdfkeywords . . . 4,
10, 13, 17, 25, 26, 55
pdflang 4–6, 14, 15,
17, 29, 30, 46, 55, 83
pdflicenseurl
. . . 4, 5, 14, 56, 81
pdfmark 72
pdfmetadate 5, 7, 19, 83
pdfmetalang . . . 5,
6, 14, 15, 46, 80, 83
pdfmoddate . . 4, 7, 83
pdfnumpages
. 5, 7, 16, 17
pdfpagerange
. 5, 7, 16, 17
pdfpart 8
pdfproducer . 4, 17, 52
pdfpublication . . . 5–7
pdfpublisher 5, 6
pdfpubtype 5, 6
pdfrendition . . 5, 7, 84
pdfsource 5, 8, 83
pdfsubject 4, 10, 17, 55
pdfsubtitle 5
pdftitle 4, 5,
10, 14, 17, 30, 55, 82
pdftrapped 4, 8, 17, 84
pdftype 5, 7, 83
pdfuapart . 5, 8, 20, 58
pdfurl 5, 6
pdfversionid . . 5, 6, 57
pdfvolumenum . . 5, 7
pdfxstandard
. . . 5, 8, 20, 21, 58
ps2pdf 72
textures 72
unicode 14, 83
vtexpdfmark 72

50, 52, 53, 63, 68,
69, 71, 72, 74, 82, 83
PDF/A 3,
8, 11, 12, 20, 24,
29, 51, 53, 58, 61,
64, 65, 67, 68, 81–83
PDF/UA 3, 8,
20, 29, 58, 64, 69, 85
PDF/X 3, 8, 20,
21, 29, 58, 65, 69, 85
pdf:Keywords . . . 2, 11, 53
pdf:PDFVersion . . . 3, 53
pdf:Producer . . . 3, 51, 53
pdf:trapped 3
\PDF@FinishDoc
. 180, 188, 194
pdfa (option) 8, 20, 29, 83
pdfaconformance (op-
tion) . . . 4, 8, 58
pdfaid:conformance . . . 3
pdfaid:part 3
pdfapart (option)
. 4, 8, 20, 58
pdfaType:prefix 82
pdfauthor (option) . 4,
5, 10, 13, 14, 17,
25, 26, 30, 55, 82
pdfauthortitle (option)
. 4, 5, 14
pdfbookedition (option)
. 4, 7
pdfbytes (option) . . . 4, 8
pdfcaptionwriter (op-
tion) 4, 5
\pdfcatalog 1678
\pdfcompresslevel . 1672
pdfcontactaddress (op-
tion) . . . 4, 5, 13
pdfcontactcity (option)
. 4, 5
pdfcontactcountry (op-
tion) 4, 5
pdfcontactemail (op-
tion) 4, 5
pdfcontactphone (op-
tion) 4, 5
pdfcontactpostcode (op-
tion) 4, 5
pdfcontactregion (op-
tion) 4, 5

P

\PackageWarning . . .
. 62, 97, 407
\PackageWarningNoLine
. 181, 298,
308, 324, 369, 1655
\patchcmd 187, 193
PDF 1–4, 7, 8,
10–13, 16, 18, 19,
24, 25, 28–31, 33–
35, 38, 40, 41, 49,

pdfcontacturl (option)	pdfproducer (option)	prism:publicationName
..... 4, 5, 14 4, 17, 52 3
pdfcopyright (option)	pdfpublication (option)	prism:subtitle
... 4, 5, 55, 56, 81 5-7 3
pdfcreationdate (option)	pdfpublisher (option)	prism:url
..... 4, 7, 30, 83	5, 6 3
\pdfcreationdate	pdfpubtype (option)	prism:volume
.. 551	5, 6 3
pdfdate (option)	pdfrendition (option)	\ProcessKeyvalOptions
... 4, 5, 7, 84 246
7, 14, 18, 50, 82, 85	pdfsource (option)	Producer
PDFDocEncoding	5, 8, 83 53
..... 25, 39, 40	\pdfstringdef	properties, XMP
pdfdocumentid (option)	23	dc:creator
..... 4, 6, 83, 84	pdfsubject (option)	2, 10, 55, 83
pdfdoi (option) 4, 10, 17, 55	dc:date
.... 4, 6	pdfsubtitle (option)	2, 55
pdfeissn (option) 5	dc:description
.... 4, 6	pdfTeX	... 3, 10, 46, 55, 83
pdfescape	8, 17,	dc:format
..... 18	41, 71, 72, 75, 83, 85	2
\pdfextension	\pdftexbanner	dc:language
1682, 1686 977 3, 55, 81, 82
\pdffeedback	pdftitle (option)	dc:publisher
.. 554, 1686	4, 5, 3
pdfinstanceid (option)	10, 14, 17, 30, 55, 82	dc:rights
..... 4, 6, 83, 84	pdftrapped (option)	2, 15, 46, 55
pdfisbn (option) 4, 8, 17, 84	dc:source
.... 4, 6	pdftype (option)	... 2, 55, 81
pdfissn (option)	5, 7, 83	dc:subject
.... 4, 6	pdfuaid:part 2, 55
pdfissuenum (option) 3	dc:title
4, 7	pdfuapart (option)	3, 10, 46, 55, 83
pdfkeywords (option) 5, 8, 20, 58	dc:type
4,	pdfurl (option)	3
10, 13, 17, 25, 26, 55 5, 6	lptc4xmpCore:ContactInfo
pdflang (option)	\pdfvariable 60, 66
.... 4-6, 14, 15, 1008	lptc4xmpCore:CreatorContactInfo
17, 29, 30, 46, 55, 83	pdfversionid (option) 2, 3, 60, 82
\pdflastobj 5, 6, 57	pdf:Keywords
..... 1678	pdfvolumenum (option)	2, 11, 53
pdfL ^A T _E X 5, 7	pdf:PDFVersion
4, 10, 13, 37, 53	pdfxid:GTS_PDFXVersion	3, 53
pdflicenseurl (option) 3	pdf:Producer
... 4, 5, 14, 56, 81	pdfxstandard (option)	3, 51, 53
pdfmark (option)	... 5, 8, 20, 21, 58	pdf:trapped
\pdfmark	photoshop:AuthorsPosition 3
..... 1689, 3, 58	pdfaid:conformance
1692, 1696,	photoshop:CaptionWriter	3
1706, 1710, 1714 3, 58	pdfaid:part
pdfmetadate (option)	PI 3
..... 5, 7, 19, 83 50	pdfaType:prefix
pdfmetalang (option)	PRISM	.. 82
5,	... 6, 60, 61, 67, 69	pdfuaid:part
6, 14, 15, 46, 80, 83	prism:aggregationType 3
\pdfminorversion	3	pdfxid:GTS_PDFXVersion
.. 1005	prism:bookEdition 3
pdfmoddate (option) 2	photoshop:AuthorsPosition
..... 4, 7, 83	prism:byteCount 3, 58
pdfnumpages (option) 2	photoshop:CaptionWriter
..... 5, 7, 16, 17	prism:doi 3, 58
\pdfobj	prism:elssn	prism:aggregationType
..... 1674	prism:isbn 3
pdfpagerange (option)	prism:issn	prism:bookEdition
..... 5, 7, 16, 17	prism:number	.. 2
pdfpart (option)	prism:pageCount	prism:byteCount
..... 8	prism:pageRange	... 2
		prism:doi
	 2
		prism:elssn
	 2
		prism:isbn
	 2
		prism:issn
	 2
		prism:number
	 2
		prism:pageCount
		... 3
		prism:pageRange
		.. 3

prism:publicationName	\setkeys	834	X _g TeX	39,
..... 3	\special	1720,	42, 74, 75, 80, 81, 85	
prism:subtitle		1728, 1757, 1763	\XeTeXrevision	980
prism:url	stringenc	18	\XeTeXversion	980
prism:volume	\StringEncodingConvert		XML	1, 2, 13, 31, 39–
xmp:BaseURL	592,	42, 44–46, 50, 53–	
xmp:CreateDate		598, 609, 612, 707	55, 59, 60, 62, 69, 83	
..... 2, 30, 83	Subject	10	XMP	1, 2, 4, 6, 7, 10–
xmp:CreatorTool			19, 21, 24, 29–35,	
xmp:MetadataDate			37, 38, 41, 44–47,	
..... 2, 83	T		50–54, 56, 57, 61,	
xmp:ModifyDate	TeX	15,	63, 68, 72–75, 80–84	
xmpMM:DocumentID		18, 36–39, 41, 42,	properties
..... 3, 47, 57, 68	texdate	47, 50–52, 57, 74, 75	see properties, XMP	
xmpMM:InstanceID	Text		xmp:BaseURL	2
..... 3, 47, 57, 68	\textunderscore		xmp:CreateDate	2, 30, 83
xmpMM:RenditionClass	21, 22, 24	xmp:CreatorTool	3
..... 3	textures (option)	72	xmp:MetadataDate	2, 83
xmpMM:VersionID	\time	528, 536	xmp:ModifyDate	2, 83
..... 3, 57, 84	Title	10	\xmpcomma	158,
xmpRights:Marked	totpages	16	161, 213, 234, 392	
..... 2, 56, 81	U		xmpincl	4
xmpRights:WebStatement	Unicode	14, 18, 39–	\XMPLangAlt	831
..... 3, 56, 81	43, 55, 59, 69, 75, 81		\xmplinesep
ps2pdf (option)	unicode (option)	14, 83	.. 1178, 1194, 1217	
..... 72	URL	2, 3, 5, 6, 14,	xmpMM:DocumentID	..
Q	19, 23, 24, 56–58, 60	 3, 47, 57, 68	
\Q	UTF-16BE	41	xmpMM:InstanceID	..
..... 569, 578	UTF-32BE	40 3, 47, 57, 68	
R	UTF-8	40	xmpMM:RenditionClass	3
RDF	UUID	3, 6,	xmpMM:VersionID	3, 57, 84
..... 54	21, 22, 47, 49, 50, 82		\xmpquote 159,
rdf:Description	V		162, 213, 234, 401	
..... 84	\vfuzz	577	xmpRights:Marked	2, 56, 81
rdf:li	vtexpdfmark (option)	72	xmpRights:WebStatement 3, 56, 81
..... 2	X		\xmptilde 402
rdf:Seq	\x	695	\XMPTruncateList	.. 406
..... 2	xdvipdfmx	13, 30, 74	Y	
\renewcommand	X _g TeX	10,	\year	517
..... 247	13, 30, 38, 53, 82, 85			
\RequirePackage				
.....				
7, 10–17, 119, 1667				
S				
\scantokens				
..... 986, 989				
\SE->pdfdoc@03				
..... 585				
\SE->pdfdoc@15				
..... 586				