

The hyperxmp package*

Scott Pakin
scott+hyxmp@pakin.org

April 5, 2019

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. The generated PDF file will contain, among other information, the following stanza of XMP code embedded within it:

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

*This document corresponds to `hyperxmp` v4.1, dated 2019/04/05.

```
</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 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 compliance level and version (pdfaid:part and pdfaid:conformance)
- position/title of primary author (photoshop:AuthorsPosition)
- publication name of parent publication (prism:publicationName)
- publisher of the document (dc:publisher)
- summary (dc:description)
- subtitle (prism:subtitle)
- telephone number(s) of primary author
(Iptc4xmpCore:CreatorContactInfo.CiTelWork)
- title (dc:title)
- 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.CiUriWork)
- 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`
- `pdfauthor`
- `pdfcreationdate`
- `pdfkeywords`
- `pdflang`
- `pdfmoddate`
- `pdfproducer`
- `pdfsubject`
- `pdftitle`

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

- `pdfaformance`
- `pdfapart`
- `pdfauthortitle`
- `pdfbookedition`
- `pdfbytes`
- `pdfcaptionwriter`
- `pdfcontactaddress`
- `pdfcontactcity`
- `pdfcontactcountry`
- `pdfcontactemail`
- `pdfcontactphone`
- `pdfcontactpostcode`
- `pdfcontactregion`
- `pdfcontacturl`
- `pdfcopyright`
- `pdfdate`
- `pdfdocumentid`
- `pdfdoi`
- `pdfeissn`
- `pdfinstanceid`
- `pdfisbn`
- `pdfissn`
- `pdfissuenum`
- `pdflicenseurl`
- `pdfmetadate`
- `pdfmetalang`
- `pdfnumpages`

- `pdfpagerange`
- `pdfsource`
- `pdfversionid`
- `pdfpublication`
- `pdfsubtitle`
- `pdfvolumenum`
- `pdfpublisher`
- `pdftype`
- `pdfpubtype`
- `pdfurl`

2.1 Option descriptions

`pdftitle` The document title is specified as normal for `hyperref` with `pdftitle`, but see Note 7 on page 14 for instructions on how to specify a title in multiple languages. `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`.

`pdfauthortitle` `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` as-

signs 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 4.1 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 14) version of the current Git hash that can suitably be passed to `pdfversionid`.

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 while `pdfeissn` 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 with `pdfvolumenum` and the issue number within the volume with `pdfissuenum`. `pdfpagerange` indicates the page numbers at which the document appears within the publication. The intention is that this be a comma-separated list of dash-separated ranges, as in `pdfpagerange={1,4-5}`. See Note 9 on page 15 for advice 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}`.

<code>pdfnumpages</code>	The number of pages in the published, print version of the document can be expressed with <code>pdfnumpages</code> . Note 9 on page 15 explains how to automatically assign a value to <code>pdfnumpages</code> .
<code>pdfdate</code>	XMP metadata can include a number of dates (in fact, timestamps, as they include both date and time components). <code>pdfdate</code> specifies the document date. It is analogous to the \LaTeX <code>\date</code> command, and, like <code>\date</code> , defaults to the date the document was built. It must be specified in either XMP format [4] or PDF format [3]. XMP dates are written in the form <code>YYYY-MM-DDThh:mm:ss+TT:tt</code> . 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 <code>2014-09-23T14:15:09-06:00</code> . This can be truncated (with loss of information) to <code>2014-09-23T14:15:09</code> , <code>2014-09-23T14:15</code> , <code>2014-09-23</code> , <code>2014-09</code> , or <code>2014</code> but no other subsets. PDF dates are written in the form <code>D:YYYYMMDDhhmmss+TT'tt'</code> . The same date in the preceding example would be written as <code>D:20140923141509-06'00'</code> in PDF format.
<code>pdfcreationdate</code> <code>pdfmoddate</code> <code>pdfmetadate</code>	The document's creation date, modification date, and metadata date are normally set automatically, but <code>pdfcreationdate</code> , <code>pdfmoddate</code> , and <code>pdfmetadate</code> can be used to override the defaults. Like <code>pdfdate</code> , <code>pdfmetadate</code> can be specified in either XMP or PDF format. However, because <code>hyperref</code> defines <code>pdfcreationdate</code> and <code>pdfmoddate</code> and expects these to be written as PDF dates, <code>hyperxmp</code> concomitantly accepts these two dates only in PDF format as well. Note that it's rare that a document would need to specify any of <code>pdfcreationdate</code> , <code>pdfmoddate</code> , or <code>pdfmetadate</code> .
<code>pdftype</code>	<code>pdftype</code> describes the type of document being produced. This refers to "the nature or genre of the resource" [4] such as <code>poem</code> , <code>novel</code> or <code>working paper</code> , as opposed to the file format (always <code>application/pdf</code> when generated by <code>hyperxmp</code>). Although <code>pdftype</code> can be assigned an arbitrary piece of text, the XMP specification recommends selecting types from a "controlled vocabulary" such as the DCMI Type Vocabulary [5]. The DCMI Type Vocabulary currently consists of only <code>Collection</code> , <code>Dataset</code> , <code>Event</code> , <code>Image</code> , <code>InteractiveResource</code> , <code>MovingImage</code> , <code>PhysicalObject</code> , <code>Service</code> , <code>Software</code> , <code>Sound</code> , <code>StillImage</code> , and <code>Text</code> . <code>pdftype</code> defaults to <code>Text</code> , which refers to "books, letters, dissertations, poems, newspapers, articles, archives of mailing lists," [5] and other forms of text—all things \LaTeX is commonly used to typeset.
<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>pdfsource</code>	A rarely needed option, <code>pdfsource</code> , overrides the name of the \LaTeX 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.

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

`pdfaconformance` Two other rarely needed options, `pdfaconformance` and `pdfapart`, are used in conjunction with `hyperref`'s `pdfa` option to claim a particular PDF/A standard by which the document abides. They default to `pdfapart=1` and `pdfaconformance=B`, indicating the PDF/A-1B standard. These can be changed (with caution) to assert that the document abides by a different standard (e.g., PDF/A-2U).

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

2.2 A complete example

The following is a sample L^AT_EX document that provides values for most of the metadata options that `hyperxmp` recognizes:

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

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

\hypersetup{%
  pdftitle={%
    On a heuristic viewpoint concerning the production and
    transformation of light},
  pdfsubtitle={[en-US]Putting that bum Maxwell in his place},
  pdfauthor={Albert Einstein},
  pdfauthortitle={\xmpquote{Technical Assistant\xmpcomma\ Level III}},
  pdfdate={1905-03-17},
  pdfcopyright={Copyright (C) 1905, Albert Einstein},
  pdfsubject={photoelectric effect},
  pdfkeywords={energy quanta, Hertz effect, quantum physics},
  pdflicenseurl={http://creativecommons.org/licenses/by-nc-nd/3.0/},
  pdfcaptionwriter={Scott Pakin},
  pdfcontactaddress={Kramgasse 49},
  pdfcontactcity={Bern},
  pdfcontactpostcode={3011},
  pdfcontactcountry={Switzerland},
  pdfcontactphone={031 312 00 91},
  pdfcontactemail={aeinstein@ipi.ch},
  pdfcontacturl={%
    http://einstein.biz/,
    https://www.facebook.com/AlbertEinstein
  }},
```



```

pdfdocumentid={uuid:6d1ac9ec-4ff2-515a-954b-648eeb4853b0},
pdfversionid={2.998e8},
pdfpublication={[de]Annalen der Physik},
pdfpublisher={Wiley-VCH},
pdfpubtype={journal},
pdfvolumenum={322},
pdfissuenum={6},
pdfpagerange={132-148},
pdfnumpages={17},
pdfissn={0003-3804},
pdfeissn={1521-3889},
pdflang={en},
pdfmetalang={en},
pdfurl={http://www.physik.uni-augsburg.de/annalen/history/einstein-papers/1905_17_132-148.p
pdfdoi={10.1002/andp.19053220607},
pdfbytes={\pdffilesize{\jobname.pdf}} % Requires pdflatex
}
\XMLLangAlt{de}{pdftitle={Über einen die Erzeugung und Verwandlung des
Lichtes betreffenden heuristischen Gesichtspunkt}}

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

```

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

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

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

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

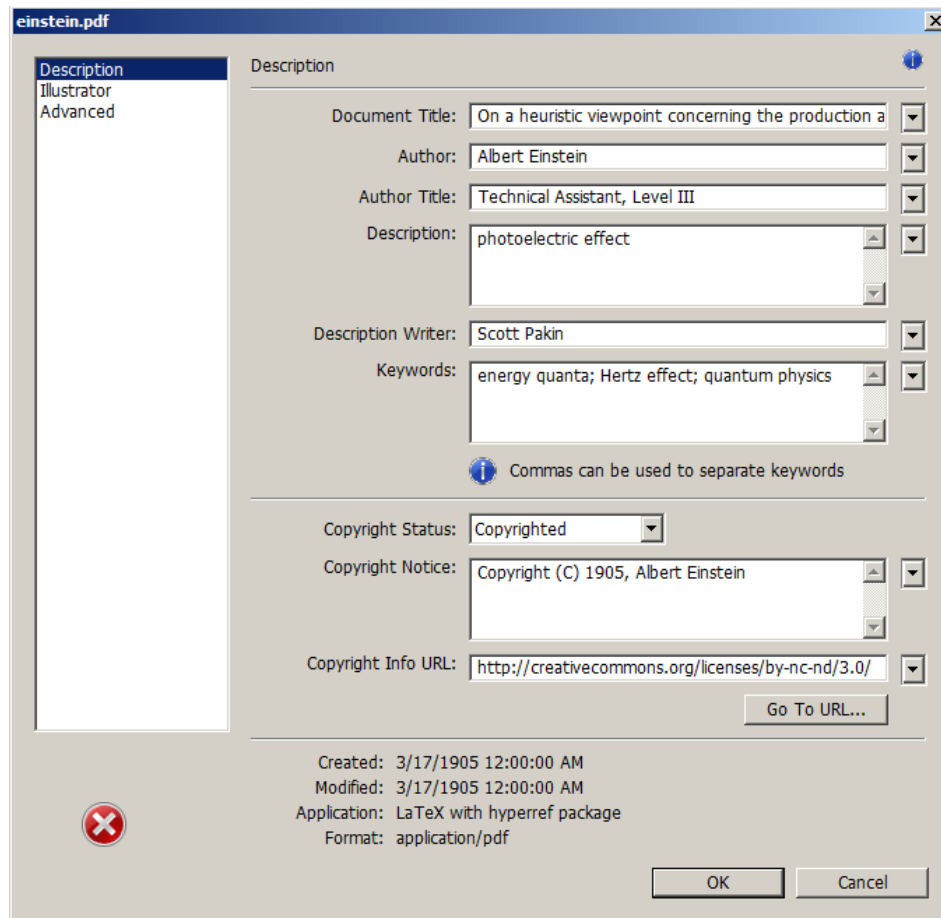


Figure 1: XMP metadata as it appears in Adobe Acrobat

2.3 Usage notes

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

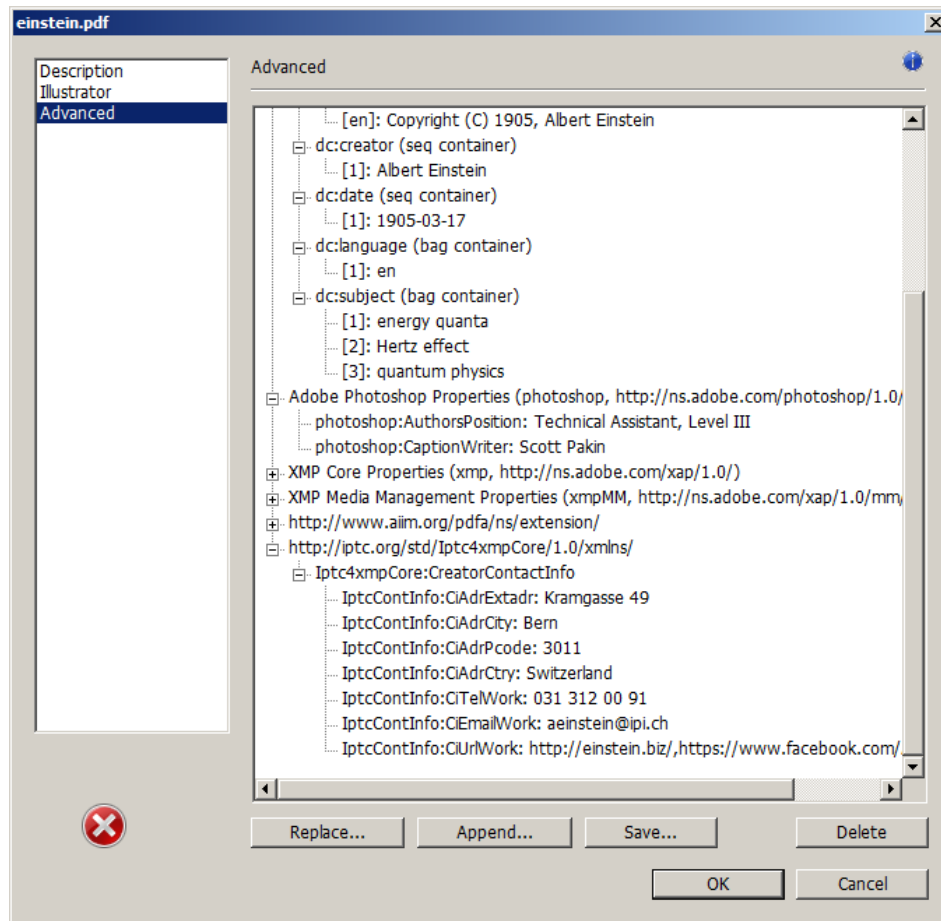


Figure 2: Additional XMP metadata as it appears in Adobe Acrobat

```

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

```

while in PDF, the author and keywords are specified as flat strings. Alas, there is no definition of how a list should be collapsed to a flat string: “Curly Howard, Larry Fine, Moe Howard” or “Curly Howard; Larry Fine; Moe Howard” or something else. I have not yet found a form of flat string that passes all PDF/A validators. Furthermore, when Adobe Acrobat—at least Adobe Acrobat DC (2019)

and earlier versions—converts a PDF file to PDF/A format, it does so by discarding all but the first author, which is an unsatisfying solution.

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

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

`\xmplinesep`

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

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

1. When run with versions of Lua \LaTeX earlier than 0.85, `hyperxmp` leaves all PDF objects uncompressed. This is due to Lua \LaTeX treating object compression as a global parameter, unlike pdf \LaTeX , which treats it as a local parameter. Hence, when `hyperxmp` requests that the XMP packet be left uncompressed, Lua \LaTeX in fact leaves *all* PDF streams uncompressed. Beginning with version 3.0, `hyperxmp` includes a workaround that correctly leaves only the XMP metadata uncompressed, but this workaround is implemented only for Lua \LaTeX v0.85 onwards.
2. Xe \LaTeX (or, more precisely, the `xdvipdfmx` back end) exhibits the opposite problem. It compresses *all* PDF objects, including the ones containing XMP metadata. While Adobe Acrobat can still detect and utilize the XMP metadata, non-PDF-aware applications are unlikely to see the metadata. Three options to consider are to (1) use a different program (e.g., Lua \LaTeX), (2) pass the `--output-driver="xdvipdfmx -z0"` option to Xe \LaTeX to instruct `xdvipdfmx` to turn off all compression (which will of course make the PDF file substantially larger), or (3) postprocess the generated PDF file by loading it into the commercial version of Adobe Acrobat and re-saving it with the Save As... menu option.

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

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

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

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

`\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.

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 T_EX terminology. This implies that any macros that are used in arguments are limited to a relatively small set of operations (such as conditionals and macro expansion) and must produce a string of text. Code (such as macro definitions and arithmetic operations) will be written to XMP as code, not as the result of executing the code.

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

```
\usepackage{texdate}
\initcurrdate
```

```

\hypersetup{%
  pdfcopyright={Copyright \textcopyright\ \printfddate{Y}, Scott Pakin}
}

```

Rather, it generates a `dc:rights` tag of the form “Copyright © =2=0=by-1by=02019, 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 L^AT_EX to do that on your behalf:

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

3 Implementation

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

3.1 Initial preparation

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

```
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 pdfT_EX, 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 addition L^AT_EX run.

```
3 \def\hyxmp@driver{hpdfTeX}
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 its XMP metadata from `hyperref`'s `baseurl`, `pdfauthor`, `pdfkeywords`, `pdflang`, `pdfproducer`, `pdfsubject`, 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: `kvoptions` for package-option processing, `pdfescape` and `stringenc` for re-encoding Unicode strings, `intcalc` for performing integer calculations (division and modulo), `ifxetex` for detecting X_YL^AT_EX, and `ifmtarg` for testing if a macro argument is empty or all spaces.

```

10 \RequirePackage{kvoptions}
11 \RequirePackage{pdfescape}
12 \RequirePackage{stringenc}
13 \RequirePackage{intcalc}
14 \RequirePackage{ifxetex}
15 \RequirePackage{ifmtarg}

```

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

```

16 \def\@ifmtargexp#1{\expandafter\@ifmtarg\expandafter{#1}}
17 \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.

```

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

```

`\@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.

```

24 \def\@pdfdatetime{}
25 \define@key{Hyp}{pdfdate}{%
26   \begingroup
27     \Hy@unicodedefalse

```

```

\pdfdate Expand pdfdate's argument and convert it to XMP format.
28 \edef\next{%
29 \noexpand\hyxmp@pdfstringdef\noexpand\pdfdatettime{%
30 \noexpand\hyxmp@as@xmp@date{#1}}%
31 }%
32 \next
33 \endgroup
34 }

\pdfmetadatettime Prepare to store the document's metadata date and (optionally) time. Whether
specified by the author in XMP format or PDF format (see Section 3.3.2) we always
store \pdfmetadatettime as an XMP-format string.
35 \def\pdfmetadatettime{}
36 \define@key{Hyp}{pdfmetadate}{%
37 \beginngroup
38 \Hy@unicodetfalse

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

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

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

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

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

\pdfcaptionwriter Prepare to store the name of the person who inserted the hyperxmp metadata.
54 \def\pdfcaptionwriter{}
55 \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.
56 \def\pdfmetalang{}
57 \define@key{Hyp}{pdfmetalang}{\hyxmp@pdfstringdef\pdfmetalang{#1}}

```

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

```
58 \def\@pdfapart{1}
59 \define@key{Hyp}{pdfapart}{\hyxmp@pdfstringdef\@pdfapart{#1}}
```

`\@pdfaconformance` Prepare to store the PDF/A conformance ID, which defaults to “B”.

```
60 \def\@pdfaconformance{B}
61 \define@key{Hyp}{pdfaconformance}{\hyxmp@pdfstringdef\@pdfaconformance{#1}}
```

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

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

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

```
64 \def\hyxmp@DocumentID{}
65 \define@key{Hyp}{pdfdocumentid}{\hyxmp@pdfstringdef\hyxmp@DocumentID{#1}}
```

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

```
66 \def\hyxmp@InstanceID{}
67 \define@key{Hyp}{pdfinstanceid}{\hyxmp@pdfstringdef\hyxmp@InstanceID{#1}}
```

`\@pdfversionid` Prepare to store a UUID that represents the current version of the document.

```
68 \def\@pdfversionid{}
69 \define@key{Hyp}{pdfversionid}{\hyxmp@pdfstringdef\@pdfversionid{#1}}
```

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

```
70 \def\@pdfpublication{}
71 \define@key{Hyp}{pdfpublication}{\hyxmp@pdfstringdef\@pdfpublication{#1}}
```

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

```
72 \def\@pdfpubtype{}
73 \define@key{Hyp}{pdfpubtype}{\hyxmp@pdfstringdef\@pdfpubtype{#1}}
```

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

```
74 \def\@pdfbytes{}
75 \define@key{Hyp}{pdfbytes}{\hyxmp@pdfstringdef\@pdfbytes{#1}}
```

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

```
76 \def\@pdfnumpages{}
77 \define@key{Hyp}{pdfnumpages}{\hyxmp@pdfstringdef\@pdfnumpages{#1}}
```

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

```
78 \def\@pdfissn{}
79 \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.

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

`\@pdfisbn` Prepare to store the ISBN of the publication in which the document was published.
82 `\def\@pdfisbn{}`
83 `\define@key{Hyp}{pdfisbn}{\hyxmp@pdfstringdef\@pdfisbn{#1}}`

`\@pdfbookedition` Prepare to store the edition of the book in which the document was published.
84 `\def\@pdfbookedition{}`
85 `\define@key{Hyp}{pdfbookedition}{\hyxmp@pdfstringdef\@pdfbookedition{#1}}`

`\@pdfpublisher` Prepare to store the name of the document’s publisher.
86 `\def\@pdfpublisher{}`
87 `\define@key{Hyp}{pdfpublisher}{\hyxmp@pdfstringdef\@pdfpublisher{#1}}`

`\@pdfvolumenum` Prepare to store the volume identifier of the publication in which the document was published.
88 `\def\@pdfvolumenum{}`
89 `\define@key{Hyp}{pdfvolumenum}{\hyxmp@pdfstringdef\@pdfvolumenum{#1}}`

`\@pdfissuenum` Prepare to store the identifier of the issue within a volume of the publication in which the document was published.
90 `\def\@pdfissuenum{}`
91 `\define@key{Hyp}{pdfissuenum}{\hyxmp@pdfstringdef\@pdfissuenum{#1}}`

`\@pdfpagerange` Prepare to store the document’s range of pages within the publication in which the document was published.
92 `\def\@pdfpagerange{}`
93 `\define@key{Hyp}{pdfpagerange}{\hyxmp@pdfstringdef\@pdfpagerange{#1}}`

`\@pdfdoi` Prepare to store a DOI that represents the current instance of the document.
94 `\def\@pdfdoi{}`
95 `\define@key{Hyp}{pdfdoi}{\hyxmp@pdfstringdef\@pdfdoi{#1}}`

`\@pdfurl` Prepare to store a URL that represents where the document can be found. Note that we do not prepend `baseurl` to the value provided.
96 `\def\@pdfurl{}`
97 `\define@key{Hyp}{pdfurl}{\hyxmp@pdfstringdef\@pdfurl{#1}}`

`\@pdfsubtitle` Prepare to store the document’s subtitle.
98 `\def\@pdfsubtitle{}`
99 `\define@key{Hyp}{pdfsubtitle}{\hyxmp@pdfstringdef\@pdfsubtitle{#1}}`

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

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

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

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

```
100 \def\@pdfcontactaddress{}
101 \define@key{Hyp}{pdfcontactaddress}{%
102   \let\xmpcomma=\hyxmp@comma
103   \def\xmpquote##1{##1}%
104   \hyxmp@pdfstringdef\@pdfcontactaddress{#1}%
105   \def\xmpcomma{,}%
106   \let\xmpquote=\relax
107 }
```

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

```
108 \def\@pdfcontactcity{}
109 \define@key{Hyp}{pdfcontactcity}{\hyxmp@pdfstringdef\@pdfcontactcity{#1}}
```

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

```
110 \def\@pdfcontactregion{}
111 \define@key{Hyp}{pdfcontactregion}{\hyxmp@pdfstringdef\@pdfcontactregion{#1}}
```

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

```
112 \def\@pdfcontactpostcode{}
113 \define@key{Hyp}{pdfcontactpostcode}{\hyxmp@pdfstringdef\@pdfcontactpostcode{#1}}
```

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

```
114 \def\@pdfcontactcountry{}
115 \define@key{Hyp}{pdfcontactcountry}{\hyxmp@pdfstringdef\@pdfcontactcountry{#1}}
```

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

```
116 \def\@pdfcontactphone{}
117 \define@key{Hyp}{pdfcontactphone}{\hyxmp@pdfstringdef\@pdfcontactphone{#1}}
```

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

```
118 \def\@pdfcontactemail{}
119 \define@key{Hyp}{pdfcontactemail}{\hyxmp@pdfstringdef\@pdfcontactemail{#1}}
```

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

```
120 \def\@pdfcontacturl{}
121 \define@key{Hyp}{pdfcontacturl}{\hyxmp@pdfstringdef\@pdfcontacturl{#1}}
```

`\hyxmp@suppress@pdf@metadata` Suppress hyperref from writing Author, Title, Subject, and other PDF metadata into the Info dictionary. This prevents conflicts between the PDF metadata and the XMP metadata, which cause PDF/A validation to fail. The PDF metadata can be restored by passing the `keeppdfinfo` option to `\hypersetup`.

```

122 \def\hyxmp@suppress@pdf@metadata{%
123   \global\let\PDF@FinishDoc=\@empty
124 }
125 \define@key{Hyp}{keeppdfinfo}[true]{%
126   \gdef\hyxmp@suppress@pdf@metadata{}}%
127 }

```

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 128 \def\hyxmp@pdfauthor{}
129 \def\hyxmp@pdfkeywords{}

\hyxmp@redefine@Hyp If not already redefined, redefine hyperref's pdfauthor and pdfkeywords options to
properly handle \xmpcomma and \xmpquote.
130 \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.
131 \@ifundefined{KV@Hyp@pdfauthor}{}{%
132   \@ifundefined{hyxmp@Hyp@pdfauthor}{%
133     \expandafter\let\expandafter\hyxmp@Hyp@pdfauthor
134     \csname KV@Hyp@pdfauthor\endcsname
135   }{}%
136 }%

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

```

to `pdfauthor`, we temporarily redefine `\and` as the list separator when producing a structured list and as “and” when producing an unstructured list.

```

137 \define@key{Hyp}{pdfauthor}{%
138   \let\xmpcomma=\hyxmp@comma
139   \def\xmpquote####1{####1}%
140   \let\hyxmp@and=\and
141   \def\and{,}%
142   \hyxmp@Hyp@pdfauthor{##1}%
143   \global\let\hyxmp@pdfauthor=\@pdfauthor
144   \def\and{and\space}%
145   \def\xmpcomma{,}%
146   \def\xmpquote####1{"####1"%
147   \hyxmp@Hyp@pdfauthor{##1}%
148   \def\xmpcomma{,}%
149   \let\xmpquote=\relax
150   \let\and=\hyxmp@and
151 }%

```

`\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`.

```

152 \ifundefined{KV@Hyp@pdfkeywords}{}{%
153   \ifundefined{hyxmp@Hyp@pdfkeywords}{%
154     \expandafter\let\expandafter\hyxmp@Hyp@pdfkeywords
155     \csname KV@Hyp@pdfkeywords\endcsname
156   }{}%
157 }%

```

`\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).

```

158 \define@key{Hyp}{pdfkeywords}{%
159   \let\xmpcomma=\hyxmp@comma
160   \def\xmpquote####1{####1}%
161   \hyxmp@Hyp@pdfkeywords{##1}%
162   \global\let\hyxmp@pdfkeywords=\@pdfkeywords
163   \def\xmpcomma{,}%
164   \def\xmpquote####1{"####1"%
165   \hyxmp@Hyp@pdfkeywords{##1}%
166   \def\xmpcomma{,}%
167   \let\xmpquote=\relax
168 }%
169 }

```

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

```

170 \let\hyxmp@ProcessKeyvalOptions=\ProcessKeyvalOptions
171 \renewcommand*{\ProcessKeyvalOptions}{%
172   \hyxmp@redefine@Hyp
173   \hyxmp@ProcessKeyvalOptions
174 }

```

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

```

175 \let\hyxmp@hypersetup=\hypersetup
176 \def\hypersetup{%
177   \hyxmp@redefine@Hyp
178   \hyxmp@hypersetup
179 }

```

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

```

180 \newcommand*{\hyxmp@find@metadata}{%
181   \edef\hyxmp@concat@metadata{%
182     \@baseurl
183     \@pdfauthor
184     \@pdfauthortitle
185     \@pdfbookedition
186     \@pdfbytes
187     \@pdfcaptionwriter
188     \@pdfcontactaddress
189     \@pdfcontactcity
190     \@pdfcontactcountry
191     \@pdfcontactemail
192     \@pdfcontactphone
193     \@pdfcontactpostcode
194     \@pdfcontactregion
195     \@pdfcontacturl
196     \@pdfcopyright
197     \@pdfcreationdate
198     \@pdfdatetime
199     \@pdfdoi
200     \@pdfeissn
201     \@pdfisbn
202     \@pdfissn
203     \@pdfissuenum
204     \@pdfkeywords
205     \@pdflang
206     \@pdflicenseurl
207     \@pdfmetadatettime

```



```

208     \@pdfmoddate
209     \@pdfnumpages
210     \@pdfpagerange
211     \@pdfpublication
212     \@pdfpubtype
213     \@pdfsubject
214     \@pdfsubtitle
215     \@pdftitle
216     \@pdftype
217     \@pdfurl
218     \@pdfversionid
219     \@pdfvolumenum
220 }%
221 \ifx\hyxmp@concat@metadata\@empty
222   \PackageWarningNoLine{hyperxmp}{%
223     \jobname.tex did not specify any metadata to\MessageBreak
224     include in the XMP packet.\space\space Please see the\MessageBreak
225     hyperxmp documentation for instructions on how to\MessageBreak
226     provide metadata values to hyperxmp}%
227 \fi
228 }

```

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.

```

229 \AtBeginDocument{%
230   \@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`.

```

231   \ifx\@pdflang\relax
232     \let\@pdflang=\@empty
233   \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.

```

234   \ifx\@pdfmetalang\@empty
235     \ifx\@pdflang\@empty
236       \let\@pdfmetalang=\hyxmp@x@default
237     \else
238       \edef\@pdfmetalang{\@pdflang}%
239     \fi
240   \fi
241   \hyxmp@xmlify\@pdfmetalang

```

If the author explicitly specified the document date, override the compilation timestamp with the specified date.

```

242   \ifx\@pdfdatetime\@empty
243   \else
244     \edef\hyxmp@today{\@pdfdatetime}%
245   \fi

```

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.

```

246   \@ifmtargexp{\@pdftitle}{%
247     \ifnotmtargexp{\@title}{%
248       \hypersetup{pdftitle={\@title}}%
249     }%
250   }%
251   {%
252     \@ifmtargexp{\@pdfauthor}{%
253       \ifnotmtargexp{\@author}{%
254         \hypersetup{pdfauthor={\@author}}%
255       }%
256     }%
257   }%

```

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.

```

258   \hyxmp@suppress@pdf@metadata

```

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.

```

259   \hyxmp@at@end{%
260     \hyxmp@find@metadata
261     \hyxmp@embed@packet
262   }%
263 }{%
264   \PackageWarningNoLine{hyperxmp}{%
265     \jobname.tex failed to include a\MessageBreak
266     \string\usepackage\string{hyperref\string}
267     in the preamble.\MessageBreak
268     Consequently, all hyperxmp functionality will be\MessageBreak
269     disabled}%
270   }%
271 }

```

3.3 Manipulating author-supplied data

The author provides metadata information to `hyperxmp` via package options to `hyperref` or via `hyperref`'s `\hypersetup` command. The functions in this section convert author-supplied lists (e.g., `pdfkeywords={foo, bar, baz}`) into L^AT_EX lists (e.g., `\@elt {foo} \@elt {bar} \@elt {baz}`) that can be more easily manipulated (Section 3.3.1); trim spaces off the ends of strings (Section 3.3.3); and, in Section 3.3.4, convert text to XML (e.g., from `<scott+hyxmp@pakin.org>` to `<scott+hyxmp@pakin.org>`).

3.3.1 List manipulation

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

```
\hyxmp@commas@to@list@i 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.)
272 \newcommand*\hyxmp@commas@to@list}[2]{%
273   \gdef#1{%
274     \expandafter\hyxmp@commas@to@list@i\expandafter#1#2,,%
275   }

\hyxmp@commas@to@list@i Recursively construct macro #1 from comma-separated list #2. Stop if #2 is empty.
  \next 276 \def\hyxmp@commas@to@list@i#1#2,{%
277   \gdef\hyxmp@sublist{#2}%
278   \ifx\hyxmp@sublist\@empty
279     \let\next=\relax
280   \else
281     \hyxmp@trimspaces\hyxmp@sublist
282     \@cons{#1}{\hyxmp@sublist}%
283     \def\next{\hyxmp@commas@to@list@i{#1}}%
284   \fi
285   \next
286 }

\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.
287 \def\xmpcomma{,}%

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

288 \bgroup
289 \catcode'\^C=11
```

```

290 \gdef\hyxmp@comma{^~C}
291 \egroup

\hyxmp@uscore This is what \_ temporarily maps to during packet construction. Because un-
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.

292 \bgroup
293 \catcode'\^~U=11
294 \gdef\hyxmp@uscore{^~U}
295 \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.

296 \let\xmpquote=\relax

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

297 \bgroup
298 \catcode'\~=12%
299 \gdef\xmptilde{~}%
300 \egroup

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

301 \newcommand{\XMPTruncateList}[1]{%
302 \PackageWarning{hyperxmp}{%
303 \noexpand\XMPTruncateList has been deprecated since\MessageBreak
304 hyperxmp 4.0 and may be removed in future\MessageBreak
305 versions of the package. \noexpand\XMPTruncateList\MessageBreak
306 was found}%
307 \edef\hyxmp@temp@str{\csname hyxmp@#1\endcsname}%
308 \hyxmp@commas@to@list{\hyxmp@temp@list}{\hyxmp@temp@str}%
309 \def\@elt##1{%
310 \expandafter\gdef\csname @#1\endcsname{##1}%
311 \let\@elt=\@gobble
312 }
313 \hyxmp@temp@list
314 }}

```

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:20190405065102-06’00’) [3], while XMP timestamps are of the form “YYYY-MM-DDThh:mm:ss+TT:tt” (e.g., 2019-04-05T06:51:02-06:00) [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 315 \def\hyxmp@first@char#1{\hyxmp@first@char@i#1\relax}
316 \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.
- ```
317 \def\hyxmp@as@xmp@date#1{%
318 \expandafter\ifx\hyxmp@first@char@i#1\relax D%
319 \hyxmp@pdf@to@xmp@date{#1}%
320 \else
321 #1%
322 \fi
323 }
```
- `\hyxmp@pdf@to@xmp@date` Convert a timestamp from PDF format to XMP format. This macro is fully expandable.
- ```
324 \def\hyxmp@pdf@to@xmp@date#1:#2#3#4#5#6#7#8#9{%
325 #2#3#4#5-#6#7-#8#9%
326 \hyxmp@parse@time
327 }
```
- `\hyxmp@parse@time` This is a helper function for `\hyxmp@pdf@to@xmp@date`. `\hyxmp@pdf@to@xmp@date` properly 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`.
- ```
328 \def\hyxmp@parse@time#1#2#3#4#5#6{%
329 T#1#2:#3#4:#5#6%
330 \hyxmp@parse@tz@char
331 }
```
- `\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.

```

332 \def\hyxmp@parse@tz@char#1{%
333   #1%
334   \ifx#1-%
335     \expandafter\hyxmp@parse@tz
336   \else
337     \ifx#1+%
338       \expandafter\hyxmp@parse@tz
339     \fi
340   \fi
341 }

```

`\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.

```

342 \def\hyxmp@parse@tz#1'#2'{%
343   #1:#2%
344 }

```

`\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.

```

345 \def\hyxmp@as@pdf@date#1{%
346   \expandafter\ifx\hyxmp@first@char@i#1\relax D%
347   #1%
348   \else
349     \hyxmp@xmp@to@pdf@date{#1}%
350   \fi
351 }

```

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

```

352 \def\hyxmp@xmp@to@pdf@date#1{%
353   D:\hyxmp@xmp@to@pdf@date@i#1\relax\relax
354 }

```

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

```

355 \def\hyxmp@xmp@to@pdf@date@i#1#2#3#4#5#6{%
356   #1#2#3#4%
357   \ifx#5-%
358     \expandafter\hyxmp@xmp@to@pdf@date@ii\expandafter#6%
359   \fi
360 }

```

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

```

361 \def\hyxmp@xmp@to@pdf@date@ii#1#2#3#4{%
362   #1#2%
363   \ifx#3-%
364     \expandafter\hyxmp@xmp@to@pdf@date@iii\expandafter#4%
365   \fi
366 }

```

```

\hyxmp@xmp@to@pdf@date@iii Parse the day for \hyxmp@xmp@to@pdf@date.
367 \def\hyxmp@xmp@to@pdf@date@iii#1#2#3#4{%
368 #1#2%
369 \ifx#3T%
370 \expandafter\hyxmp@xmp@to@pdf@date@iv\expandafter#4%
371 \fi
372 }

\hyxmp@xmp@to@pdf@date@iv Parse the hour for \hyxmp@xmp@to@pdf@date.
373 \def\hyxmp@xmp@to@pdf@date@iv#1#2#3#4{%
374 #1#2%
375 \ifx#3:%
376 \expandafter\hyxmp@xmp@to@pdf@date@v\expandafter#4%
377 \fi
378 }

\hyxmp@xmp@to@pdf@date@v Parse the minute for \hyxmp@xmp@to@pdf@date.
379 \def\hyxmp@xmp@to@pdf@date@v#1#2#3#4{%
380 #1#2%
381 \ifx#3:%
382 \expandafter\hyxmp@xmp@to@pdf@date@vi\expandafter#4%
383 \fi
384 }

\hyxmp@gobbletwo This is exactly the same as LATEX 2ε's \@gobbletwo but needs to be a different
literal for \hyxmp@xmp@to@pdf@date@vii's pattern-matching to work.
385 \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).
386 \def\hyxmp@xmp@to@pdf@date@vi#1#2#3#4{%
387 #1#2%
388 \ifx#3+%
389 +\expandafter\hyxmp@xmp@to@pdf@date@vii
390 \fi
391 \ifx#3-%
392 -\expandafter\hyxmp@xmp@to@pdf@date@vii
393 \fi
394 \ifx#3Z%
395 Z%
396 \fi
397 \ifx#3\relax
398 \expandafter\hyxmp@gobbletwo
399 \fi
400 \@gobbletwo #4%
401 }

```

```

\hyxmp@xmp@to@pdf@date@vii Parse the time-zone hours for \hyxmp@xmp@to@pdf@date.
402 \def\hyxmp@xmp@to@pdf@date@vii#1\@gobbletwo#2#3#4#5{%
403 #2#3%
404 \ifx#4:%
405 \expandafter\hyxmp@xmp@to@pdf@date@viii\expandafter#5%
406 \fi
407 }

\hyxmp@xmp@to@pdf@date@viii Parse the time-zone minutes for \hyxmp@xmp@to@pdf@date.
408 \def\hyxmp@xmp@to@pdf@date@viii#1#2#3#4{%
409 '#1#2'%
410 }

\hyxmp@today@define Use TeX primitives to define a given macro as today's date in YYYY-MM-DDThh:mm
format.
411 \def\hyxmp@today@define#1{%
The date is a straightforward representation of TeX's \year, \month, and \day
primitives, with the latter two zero-padded to two digits apiece.
412 \xdef#1{\the\year}%
413 \ifnum\month<10
414 \xdef#1{#1-0\the\month}%
415 \else
416 \xdef#1{#1-\the\month}%
417 \fi
418 \ifnum\day<10
419 \xdef#1{#1-0\the\day}%
420 \else
421 \xdef#1{#1-\the\day}%
422 \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.
423 \@tempcnta=\time
424 \divide\@tempcnta by 60%
425 \ifnum\@tempcnta<10%
426 \xdef#1{#1T0\the\@tempcnta}%
427 \else
428 \xdef#1{#1T\the\@tempcnta}%
429 \fi
430 \multiply\@tempcnta by -60%
431 \advance\@tempcnta by \time
432 \ifnum\@tempcnta<10%
433 \xdef#1{#1:0\the\@tempcnta}%
434 \else
435 \xdef#1{#1:\the\@tempcnta}%
436 \fi
437 }

```


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

```

438 \ifundefined{pdffeedback}{%
439 \ifundefined{pdfcreationdate}{%
    Case 1: Neither \pdffeedback nor \pdfcreationdate is defined (XeLaTeX and regular LATEX).
440 \hyxmp@today@define\hyxmp@today
441 }{%
    Case 2: \pdfcreationdate is defined (pdfLATEX and pre-0.85 LuaLATEX).
442 \edef\hyxmp@today{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
443 }%
444 }{%
    Case 3: \pdffeedback is defined (LuaLATEX 0.85+).
445 \edef\hyxmp@today{\expandafter\hyxmp@pdf@to@xmp@date\pdffeedback creationdate}%
446 }

```

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.

```

447 \catcode'\Q=3
\hyxmp@trimspaces\x redefines \x to have the same replacement text sans leading
and trailing space tokens.
448 \newcommand{\hyxmp@trimspaces}[1]{%
    Use grouping to emulate a multi-token afterassignment queue.
449 \begingroup
    Put “\toks 0 {” into the afterassignment queue.
450 \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.
451 \expandafter\hyxmp@trimb\expandafter\noexpand#1Q Q}%
    Transfer the trimmed text back into #1.
452 \edef#1{\the\toks0}%
453 }

```

`\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.

```

454 \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.

```
455 \def\hyxmp@trimc#1Q#2{\afterassignment\endgroup \vfuzz\the\vfuzz#1}
456 \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.

```
457 \newif\ifhyxmp@unicodetex
458 \ifnum64='\^^^0040\relax
459 \hyxmp@unicodetextrue
460 \else
461 \hyxmp@unicodetexfalse
462 \fi
```

`\hyxmp@reencode` This is now a placeholder macro needed only for `\@pdfmetalang` in the `\begin{document}`.

```
463 \newcommand*\hyxmp@reencode}[1]{} 
```

`\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.

```
464 \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.

```
465 \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 `&`.

```
466 \newcommand*\hyxmp@xmlify}[1]{%
467 \gdef\hyxmp@xmlified{}
```

```

Escaped PDF string → PDFDocEncoding/Unicode
468 \EdefUnescapeString\hyxmp@text{#1}%
469 \ifhyxmp@unicodetex
PDFDocEncoding/Unicode → UTF-32BE
470 \hyxmp@is@unicode\hyxmp@text{%
471 \StringEncodingConvert
472 \hyxmp@text\hyxmp@text{utf16be}{utf32be}%
473 }{%
474 \ifxetex
475 \hyxmp@xetex@crap
476 \else
477 \StringEncodingConvert
478 \hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
479 \fi
480 }%
UTF-32BE → UTF-32BE as hex string
481 \EdefEscapeHex\hyxmp@text{\hyxmp@text}%
UTF-32BE → XML in ASCII
482 \edef\hyxmp@text{%
483 \expandafter
484 }\expandafter\hyxmp@toxml@unicodetex\hyxmp@text
485 \relax\relax\relax\relax\relax\relax\relax\relax
486 \else
PDFDocEncoding/Unicode → UTF-8
487 \hyxmp@is@unicode\hyxmp@text{%
488 \StringEncodingConvert
489 \hyxmp@text\hyxmp@text{utf16be}{utf8}%
490 }{%
491 \StringEncodingConvert
492 \hyxmp@text\hyxmp@text{pdfdoc}{utf8}%
493 }%
UTF-8 → UTF-8 as hex string
494 \EdefEscapeHex\hyxmp@text{\hyxmp@text}%
UTF-8 as hex string → XML in UTF-8 as hex string
495 \edef\hyxmp@text{%
496 \expandafter\hyxmp@toxml\hyxmp@text\@empty\@empty
497 }%
XML in UTF-8 as hex string → XML in UTF-8
498 \EdefUnescapeHex\hyxmp@text{\hyxmp@text}%
499 \fi
500 \global\let\hyxmp@xmllified\hyxmp@text
501 }

```

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

```

502 \begingroup
503 \lccode'\<=254 %
504 \lccode'\>=255 %
505 \catcode254=12 %
506 \catcode255=12 %
507 \lowercase{\endgroup
508 \def\hyxmp@is@unicode#1{%
509   \expandafter\hyxmp@@is@unicode#1<>\@nil
510 }%
511 \def\hyxmp@@is@unicode#1<>#2\@nil{%
512   \ifx\#1\%
513     \expandafter\@firstoftwo
514   \else
515     \expandafter\@secondoftwo
516   \fi
517 }%
518 }

```

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

```

519 \def\hyxmp@toxml#1#2{%
520   \ifx#1\@empty
521   \else
522     \ifnum"#1#2='\& %
523       26616D703B% &#1#2;
524     \else\ifnum"#1#2='\< %
525       266C743B% &lt;#1#2;
526     \else\ifnum"#1#2='\> %
527       2667743B% &gt;#1#2;
528     \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.

```

529   \@ifundefined{pdfmark}{%
530     #1#2%
531   }{%
532     \ifnum"#1#2='\( %
533       5C28% \ (
534     \else\ifnum"#1#2='\) %
535       5C29% \ )

```

```

536     \else
537     #1#2%
538     \fi\fi
539     }%
540     \fi\fi\fi
541     \expandafter\hyxmp@toxml
542     \fi
543 }

\hyxmp@toxml@unicodetex Replace the characters “<”, “&”, and “>” with XML entities when using a native-
\hyxmp@text Unicode TEX (XƎTEX or LuaTEX).
544 \def\hyxmp@toxml@unicodetex#1#2#3#4#5#6#7#8{%
545     \ifx#1\relax
546     \else
547     \ifnum"#1#2#3#4#5#6#7#8>127 %
548     \uccode'\*="#1#2#3#4#5#6#7#8\relax
549     \uppercase{%
550     \edef\hyxmp@text{\hyxmp@text *}%
551     }%
552     \else\ifnum"#7#8='< %
553     \edef\hyxmp@text{\hyxmp@text &lt;};}%
554     \else\ifnum"#7#8='& %
555     \edef\hyxmp@text{\hyxmp@text &amp;};}%
556     \else\ifnum"#7#8='> %
557     \edef\hyxmp@text{\hyxmp@text &gt;};}%
558     \else\ifnum"#7#8='\ %
559     \edef\hyxmp@text{\hyxmp@text\space}%
560     \else
561     \uccode'\*="#7#8\relax
562     \uppercase{%
563     \edef\hyxmp@text{\hyxmp@text *}%
564     }%
565     \fi\fi\fi\fi\fi
566     \expandafter\hyxmp@toxml@unicodetex
567     \fi
568 }

\hyxmp@skipzeros Skip over leading zeroes in the input argument.
569 \def\hyxmp@skipzeros#1{%
570     \ifx#10%
571     \expandafter\hyxmp@skipzeros
572     \fi
573 }

\lx In the case of XƎTEX, the strings defined by \pdfstringdef can contain big
\hyxmp@xetex@crap characters. In this case, the string is treated as Unicode.
\hyxmp@try 574 \begingroup
\hyxmp@crap@result 575 \def\x#1{\endgroup
\hyxmp@text 576 \def\hyxmp@xetex@crap{%

```

```

577 \edef\hyxmp@try{%
578   \expandafter\hyxmp@SpaceOther\hyxmp@text#1\@nil
579 }%
580 \let\hyxmp@crap@result=N%
581 \expandafter\hyxmp@crap@test\hyxmp@try\relax
582 \ifx\hyxmp@crap@result Y%
583   \let\hyxmp@text\@empty
584   \expandafter\hyxmp@crap@convert\hyxmp@try\relax
585 \else
586   \StringEncodingConvert\hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
587 \fi
588 }%
589 }
590 \x{ }

```

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

```

591 \begingroup
592 \catcode'\~=12 %
593 \lccode'\~=\' %
594 \lowercase{\endgroup
595 \def\hyxmp@SpaceOther#1 #2\@nil{%
596   #1%
597   \ifx\relax#2\relax
598     \expandafter\@gobble
599   \else
600     ~%
601     \expandafter\@firstofone
602   \fi
603   {\hyxmp@SpaceOther#2\@nil}%
604 }%
605 }

```

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

```

606 \def\hyxmp@crap@test#1{%
607   \ifx#1\relax
608   \else
609     \ifnum'#1>127 %
610       \let\hyxmp@crap@result=Y%
611       \expandafter\expandafter\expandafter\hyxmp@skiptorelax
612     \else
613       \expandafter\expandafter\expandafter\hyxmp@crap@test
614     \fi
615 \fi
616 }

```

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

```

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

```

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

```

\hyxmp@num
\hyxmp@text

```

```

618 \def\hyxmp@crap@convert#1{%
619   \ifx#1\relax
620   \else
621     \edef\hyxmp@num{\number'#1}%
622     \ifnum\hyxmp@num>"FFFFFF %
623       \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"1000000}\relax
624       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
625       \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"1000000}}%
626     \else
627       \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
628     \fi
629     \ifnum\hyxmp@num>"FFFF %
630       \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"10000}\relax
631       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
632       \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"10000}}%
633     \else
634       \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
635     \fi
636     \ifnum\hyxmp@num>"FF %
637       \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"100}\relax
638       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
639       \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"100}}%
640     \else
641       \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
642     \fi
643     \ifnum\hyxmp@num>0 %
644       \lccode'\!=\hyxmp@num\relax
645       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
646     \else
647       \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
648     \fi
649     \expandafter\hyxmp@crap@convert
650   \fi
651 }

```

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

```

652 \begingroup
653   \catcode0=12 %
654   \gdef\hyxmp@zero{^^00}%
655 \endgroup

```

3.3.5 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}`”.

```
656 \def\hyxmp@alt@title{}
657 \def\hyxmp@alt@description{}
658 \def\hyxmp@alt@rights{}
```

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

```
659 \newcommand{\hyxmp@LA@accept}[2]{%
660   \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`.

```
661   \hyxmp@pdfstringdef\hyxmp@value{##1}%
662   \xdef#2{#2\noexpand\do {\hyxmp@cur@lang} {\hyxmp@value}}%
663 }
664 }
```

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

```
665 \hyxmp@LA@accept{pdftitle}{\hyxmp@alt@title}
666 \hyxmp@LA@accept{pdfsubject}{\hyxmp@alt@description}
667 \hyxmp@LA@accept{pdfcopyright}{\hyxmp@alt@rights}
```

`\XMPLangAlt` Argument `#1` is a language expressed as a two-letter country code and optional two-
letter region code. Argument `#2` is a list of `<key>=<value>` pairs. Keys correspond to
`\hypersetup` options such as “`pdftitle`”, “`pdfsubject`”, and “`pdfcopyright`”.
Values are the alternative-language form of the text provided for the corresponding
option.

```
668 \newcommand{\XMPLangAlt}[2]{%
669   \let\do=\relax
```

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

```
670   \edef\hyxmp@cur@lang{#1}%
671   \setkeys{hyxmp@LA}{#2}%
672 }
```

3.4 UUID generation

We use a linear congruential generator to produce pseudorandom version 4
UUIDs [11]. True, this method has its flaws but it’s simple to implement in T_EX and
is good enough for producing the XMP `xmpMM:DocumentID` and `xmpMM:InstanceID`
fields.

`\hyxmp@modulo@a` Replace the contents of `\@tempcnta` with the contents modulo `#1`. Note that `\@tempcntb` is overwritten in the process.

```

673 \def\hyxmp@modulo@a#1{%
674   \@tempcntb=\@tempcnta
675   \divide\@tempcntb by #1
676   \multiply\@tempcntb by #1
677   \advance\@tempcnta by -\@tempcntb
678 }

```

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

```

\hyxmp@big@prime@ii 679 \def\hyxmp@big@prime{536870923}
680 \def\hyxmp@big@prime@ii{536870027}

```

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

```

\hyxmp@one@token 681 \def\hyxmp@seed@rng#1{%
682   \@tempcnta=\hyxmp@big@prime
683   \futurelet\hyxmp@one@token\hyxmp@seed@rng@i#1\@empty
684 }

```

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

```

\hyxmp@one@token \next 685 \def\hyxmp@seed@rng@i{%
686   \ifx\hyxmp@one@token\@empty
687     \let\next=\relax
688   \else
689     \def\next##1{%
690       \multiply\@tempcnta by 3
691       \advance\@tempcnta by '##1
692       \hyxmp@modulo@a{\hyxmp@big@prime}%
693       \futurelet\hyxmp@one@token\hyxmp@seed@rng@i
694     }%
695   \fi
696 \next
697 }

```

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

```

\hyxmp@rand@num 698 \def\hyxmp@set@rand@num{%
699   \@tempcnta=\hyxmp@rand@num
700   \multiply\@tempcnta by 3
701   \advance\@tempcnta by \hyxmp@big@prime@ii
702   \hyxmp@modulo@a{\hyxmp@big@prime}%
703   \xdef\hyxmp@rand@num{\the\@tempcnta}%
704 }

```

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

```

705 \def\hyxmp@append@hex#1{%
706   \hyxmp@set@rand@num
707   \@tempcnta=\hyxmp@rand@num
708   \hyxmp@modulo@a{16}%
709   \ifnum\@tempcnta<10
710     \xdef#1{#1\the\@tempcnta}%
711   \else

```

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

```

712   \advance\@tempcnta by -10
713   \ifcase\@tempcnta
714     \xdef#1{#1a}%
715     \or\xdef#1{#1b}%
716     \or\xdef#1{#1c}%
717     \or\xdef#1{#1d}%
718     \or\xdef#1{#1e}%
719     \or\xdef#1{#1f}%
720   \fi
721 \fi
722 }

```

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

```

723 \def\hyxmp@append@hex@iii#1{%
724   \hyxmp@append@hex#1%
725   \hyxmp@append@hex#1%
726   \hyxmp@append@hex#1%
727 }

```

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

```

728 \def\hyxmp@append@hex@iv#1{%
729   \hyxmp@append@hex@iii#1%
730   \hyxmp@append@hex#1%
731 }

```

`\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`.

```

732 \def\hyxmp@create@uuid#1{%
733   \def#1{uuid:}%
734   \hyxmp@append@hex@iv#1%
735   \hyxmp@append@hex@iv#1%
736   \g@addto@macro#1{-}%
737   \hyxmp@append@hex@iv#1%
738   \g@addto@macro#1{-4}%
739   \hyxmp@append@hex@iii#1%
740   \g@addto@macro#1{-}%

```

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

```
741 \hyxmp@set@rand@num
742 \@tempcnta=\hyxmp@rand@num
743 \hyxmp@modulo@a{4}%
744 \ifcase\@tempcnta
745   \g@addto@macro#1{8}%
746   \or\g@addto@macro#1{9}%
747   \or\g@addto@macro#1{a}%
748   \or\g@addto@macro#1{b}%
749 \fi
750 \hyxmp@append@hex@iii#1%
751 \g@addto@macro#1{-}%
752 \hyxmp@append@hex@iv#1%
753 \hyxmp@append@hex@iv#1%
754 \hyxmp@append@hex@iv#1%
755 }
```

```
\hyxmp@def@DocumentID Seed the random-number generator with a function of the current filename, PDF
  \hyxmp@DocumentID document title, and PDF author, then invoke \hyxmp@create@uuid to define
  \hyxmp@seed@string \hyxmp@DocumentID as a random UUID.
756 \newcommand*{\hyxmp@def@DocumentID}{%
757   \edef\hyxmp@seed@string{\jobname:\@pdftitle:\@pdfauthor:}%
758   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
759   \edef\hyxmp@rand@num{\the\@tempcnta}%
760   \hyxmp@create@uuid\hyxmp@DocumentID
761 }
```

```
\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).
762 \newcommand*{\hyxmp@def@InstanceID}{%
763   \hyxmp@today@define{\hyxmp@seed@string}%
764   \edef\hyxmp@seed@string{%
765     \jobname:\@pdftitle:\@pdfauthor:\hyxmp@today:\hyxmp@seed@string
766   }%
767   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
768   \edef\hyxmp@rand@num{\the\@tempcnta}%
769   \hyxmp@create@uuid\hyxmp@InstanceID
770 }
```

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/A 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.

```

771 \newcommand*{\hyxmp@add@to+xml}[1]{%
772   \bgroup
773     \@tempcnta=0
774     \ifhyxmp@unicodetex
775       \@tempcntb=65536%
776     \else
777       \@tempcntb=256%
778     \fi
779     \loop
780       \lccode\@tempcnta=\@tempcnta
781       \advance\@tempcnta by 1
782       \ifnum\@tempcnta<\@tempcntb
783         \repeat
784         \lccode'\_='\ \relax
785         \lccode'\^C='\,\relax
786         \lccode'\^U='\_\relax
787         \lowercase{\xdef\hyxmp@new+xml{#1}}%
788         \xdef\hyxmp+xml{\hyxmp+xml\hyxmp@new+xml}%
789     \egroup
790 }
```

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

```

791 \bgroup
792 \catcode'\#=11
793 \gdef\hyxmp@hash{#}
794 \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@padding` is defined to contain 32 lines of 63 spaces and a newline apiece for a total of 2048 characters of whitespace.

```

795 \bgroup
796 \xdef\hyxmp+xml{}
```

```

797 \hyxmp@add@to+xml{%
798 -----^^J%
799 }
800 \xdef\hyxmp@padding{\hyxmp+xml}%
801 \egroup
802 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
803 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
804 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
805 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
806 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}

```

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

```

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

```

3.5.2 The Adobe PDF schema

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

```

808 \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.

```

809 \hyxmp@add@simple@var{pdf:Keywords}{@pdfkeywords}%
810 \hyxmp@add@simple@var{pdf:Producer}{@pdfproducer}%
811 \@ifundefined{pdfvariable}{%
812   \@ifundefined{pdfminorversion}{%

```

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

```

813   }{%

```

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

```

814     \hyxmp@add@simple{pdf:PDFVersion}{1.\the\pdfminorversion}%
815   }%
816 }{%

```

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

```

817   \hyxmp@add@simple{pdf:PDFVersion}{1.\the\pdfvariable minorversion}%
818 }%
819 }

```

`\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.

```

820 \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.

```
821 \newcommand*{\hyxmp@add@simple}[2]{%
822   \@ifnotmtargexp{#2}{%
823     \hyxmp@xmlify{#2}%
824     \hyxmp@add@to@xml{%
825       \hyxmp@extra@indent_____<#1>\hyxmp@xmllified</#1>^^J%
826     }%
827   }%
828 }
```

`\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.

```
829 \newcommand*{\hyxmp@add@simple@var}[2]{%
830   \expandafter\ifx\csname#2\endcsname\relax
831   \else
832     \hyxmp@xmlify{\csname#2\endcsname}%
833     \hyxmp@add@to@xml{%
834       \hyxmp@extra@indent_____<#1>\hyxmp@xmllified</#1>^^J%
835     }%
836   \fi
837 }
```

`\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.

```
838 \newcommand*{\hyxmp@add@simple@lang}[2]{%
839   \@ifnotmtarg{#2}{%
840     \hyxmp@xmlify{#2}%
841     \expandafter\hyxmp@add@simple@lang@i\hyxmp@xmllified\relax{#1}%
842   }%
843 }
```

`\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.

```
844 \newcommand*{\hyxmp@add@simple@lang@i}{%
845   \@ifnextchar[\hyxmp@add@simple@lang@ii{\hyxmp@add@simple@lang@ii []}%
846 }
```

`\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.

```

847 \def\hyxmp@add@simple@lang@ii[#1]#2\relax#3{%
848   \@ifnotmtarg{#2}{%
849     \hyxmp@xmlify{#2}%
850     \@ifmtarg{#1}{%
851       \hyxmp@add@to@xml{%
852         <#3>\hyxmp@xmlified</#3>^^J%
853       }%
854     }{%
855       \hyxmp@add@to@xml{%
856         <#3 xml:lang="#1">\hyxmp@xmlified</#3>^^J%
857       }%
858     }%
859   }%
860 }

```

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.

```

861 \newcommand*{\hyxmp@rdf@dc}[3][\iffalse]{%
Set \@tempswattrue only if the given text is nonempty or the provided conditional
evaluates to TRUE.
862   \@ifmtargexp{#3}{\@tempswafalse}{\@tempswattrue}%
863   #1
864   \@tempswattrue
865   \fi
Append the corresponding XML only if \@tempswattrue.
866   \if@tempswa
867     \hyxmp@xmlify{#3}%

```

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

```

868   \let\hyxmp@value=\hyxmp@xmlified
869   \hyxmp@add@to@xml{%
870     <dc:#2>^^J%
871     <rdf:Alt>^^J%
872   }%
873   \ifx\@pdfmetalang\hyxmp@x@default
874   \else
875     \hyxmp@xmlify{\@pdfmetalang}%
876     \hyxmp@add@to@xml{%
877     <rdf:li xml:lang="\hyxmp@xmlified">\hyxmp@value</rdf:li>^^J%
878   }%

```

```

879 \fi
880 \hyxmp@add@to@xml{%
881 -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@value</rdf:li>^^J%
882 }%

```

Include variants of the text expressed in other languages, as specified by the author using \XMPLangAlt (Section 3.3.5).

```

883 \def\do##1##2{
884 \hyxmp@xmlify{##2}%
885 \hyxmp@add@to@xml{%
886 -----<rdf:li xml:lang="##1">\hyxmp@xmlified</rdf:li>^^J%
887 }%
888 }%
889 \csname hyxmp@alt@#2\endcsname

```

Complete this XMP element.

```

890 \hyxmp@add@to@xml{%
891 -----</rdf:Alt>^^J%
892 -----</dc:#2>^^J%
893 }%
894 \fi
895 }%

```

`\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.

```

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

```

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

```

897 \@ifmargexp{#4}{\@tempwafalse}{\@tempwattrue}%
898 #1
899 \@tempwattrue
900 \fi

```

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

```

901 \if@tempwa
902 \hyxmp@add@to@xml{%
903 -----<dc:#2>^^J%
904 -----<rdf:#3>^^J%
905 }%
906 \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`.

```

907 \hyxmp@xmlify{#4}%
908 \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
909 \def\@elt##1{%
910 \hyxmp@add@to@xml{%
911 -----<rdf:li>##1</rdf:li>^^J%
912 }%

```



```

913     }%
914     \hyxmp@list
915     \egroup
916     \hyxmp@add@to@xml{%
917     -----</rdf:#3>^^J%
918     -----</dc:#2>^^J%
919     }%
920     \fi
921 }

```

`\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.

```

922 \newcommand{\hyxmp@singleton@dc}[3][Bag]{%
923   \ifnotmtarg{#3}{%
924     \hyxmp@xmlify{#3}%
925     \hyxmp@add@to@xml{%
926     -----<dc:#2>^^J%
927     -----<rdf:#1>^^J%
928     -----<rdf:li>\hyxmp@xmlified</rdf:li>^^J%
929     -----</rdf:#1>^^J%
930     -----</dc:#2>^^J%
931     }%
932   }
933 }

```

`\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.

```

934 \newcommand*{\hyxmp@dc@schema}{%
935   \hyxmp@add@simple{dc:format}{application/pdf}%
936   \hyxmp@rdf@dc[\ifHy@pdfa]{title}{\@pdftitle}%
937   \hyxmp@rdf@dc[\ifHy@pdfa]{description}{\@pdfsubject}%
938   \hyxmp@rdf@dc{rights}{\@pdfcopyright}%
939   \hyxmp@singleton@dc{publisher}{\@pdfpublisher}%
940   \hyxmp@singleton@dc[Seq]{date}{\hyxmp@today}%
941   \hyxmp@singleton@dc{language}{\@pdflang}%
942   \hyxmp@singleton@dc{type}{\@pdftype}%
943   \hyxmp@list@to@xml[\ifHy@pdfa]{creator}{Seq}{\hyxmp@pdfauthor}%
944   \hyxmp@list@to@xml{subject}{Bag}{\hyxmp@pdfkeywords}%
945   \ifx\@pdfsource\empty
946     \else
947     \hyxmp@add@simple{dc:source}{\@pdfsource}%
948   \fi

```

949 }

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.

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

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

```
951 \let\hyxmp@rights=\@empty
952 \ifx\@pdflicenseurl\@empty
953 \else
954 \def\hyxmp@rights{YES}%
955 \fi
956 \ifx\@pdfcopyright\@empty
957 \else
958 \def\hyxmp@rights{YES}%
959 \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.

```
960 \ifx\hyxmp@rights\@empty
961 \else
962 \ifx\@pdfcopyright\@empty
963 \else
964 \hyxmp@add@simple{xmpRights:Marked}{True}%
965 \fi
966 \hyxmp@add@simple{xmpRights:WebStatement}{\@pdflicenseurl}%
967 \fi
968 }
```

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 TeX-based workflow. We additionally support the `xmpMM:VersionID` property, whose value is supplied by the author using `pdfversionid`.

```
969 \gdef\hyxmp@mm@schema{%
970 \@ifmtargexp{\hyxmp@DocumentID}{\hyxmp@def@DocumentID}{}%
971 \@ifmtargexp{\hyxmp@InstanceID}{\hyxmp@def@InstanceID}{}%
972 \hyxmp@add@simple{xmpMM:DocumentID}{\hyxmp@DocumentID}%
973 \hyxmp@add@simple{xmpMM:InstanceID}{\hyxmp@InstanceID}%
```

```

974 \hyxmp@add@simple{xmpMM:VersionID}{\@pdfversionid}%
975 }

```

3.5.6 The XMP Basic schema

`\hyxmp@define@createdate` Define `\hyxmp@createdate` as the document's creation date but in XMP date format, not PDF date format. We use `\hyxmp@createdate` for the `xmp:CreateDate`, `xmp:ModifyDate`, and `xmp:MetadataDate` fields.

```

976 \newcommand*{\hyxmp@define@createdate}{%
977 \@ifundefined{pdffeedback}{%
978 \@ifundefined{pdfcreationdate}{%

```

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

```

979 \hyxmp@today@define\hyxmp@createdate
980 }{%

```

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

```

981 \edef\hyxmp@createdate{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
982 }%
983 }{%

```

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

```

984 \edef\hyxmp@createdate{\expandafter\hyxmp@pdf@to@xmp@date\pdffeedback creationdate}%
985 }%
986 }

```

`\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`.

```

987 \newcommand*{\hyxmp@xmp@basic@schema}{%
988 \hyxmp@define@createdate

```

For the document's creation date, use the user-specified `\@pdfcreationdate` if defined and non-empty. Otherwise use our fabricated `\hyxmp@createdate`.

```

989 \@ifundefined{@pdfcreationdate}{%
990 \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@createdate}%
991 }{%
992 \ifx\@pdfcreationdate\@empty
993 \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@createdate}%
994 \else
995 \hyxmp@add@simple{xmp:CreateDate}{%
996 \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfcreationdate}}%
997 \fi
998 }%

```

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

```

999 \@ifundefined{@pdfmoddate}{%
1000 \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@createdate}%
1001 }{%

```

```

1002 \ifx\@pdfmoddate\@empty
1003   \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@createdate}%
1004 \else
1005   \hyxmp@add@simple{xmp:ModifyDate}{%
1006     \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfmoddate}}%
1007 \fi
1008 }%

```

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

```

1009 \ifx\@pdfmetadatetime\@empty
1010   \hyxmp@add@simple{xmp:MetadataDate}{\hyxmp@createdate}%
1011 \else
1012   \hyxmp@add@simple{xmp:MetadataDate}{\@pdfmetadatetime}%
1013 \fi

```

Define the creation tool and the base URL.

```

1014 \hyxmp@add@simple{xmp:CreatorTool}{\@pdfcreator}%
1015 \hyxmp@add@simple{xmp:BaseURL}{\@baseurl}%
1016 }

```

3.5.7 The Photoshop schema

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

```

1017 \gdef\hyxmp@photoshop@schema{%
1018 \edef\hyxmp@photoshop@data{\@pdfauthortitle\@pdfcaptionwriter}%
1019 \hyxmp@add@simple{photoshop:AuthorsPosition}{\@pdfauthortitle}%
1020 \hyxmp@add@simple{photoshop:CaptionWriter}{\@pdfcaptionwriter}%
1021 }

```

3.5.8 The PDF/A Identification schema

`\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`.

```

1022 \newcommand*{\hyxmp@pdfa@id@schema}{%
1023 \ifHy@pdfa
1024   \hyxmp@add@simple{pdfaid:part}{\@pdfapart}%
1025   \hyxmp@add@simple{pdfaid:conformance}{\@pdfaconformance}%
1026 \fi
1027 }

```

3.5.9 The IPTC Photo Metadata schema

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

consistency across operating systems.

```
1028 \begingroup
1029 \catcode'\&=12
1030 \catcode'\#=12
1031 \gdef\xmplinesep{&#xA;}
1032 \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.

```
1033 \newcommand*\hyxmp@list@to@lines}[2]{%
1034 \ifnotmtargexp{#2}{%
1035 \bgroup
1036 \hyxmp@add@to+xml{%
1037 \hyxmp@extra@indent_____<#1>%
1038 }%
```

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

```
1039 \def\@elt@first##1{%
1040 \hyxmp@add@to+xml{##1}%
1041 \let\@elt=\@elt@rest
1042 }%
```

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

```
1043 \def\@elt@rest##1{%
1044 \hyxmp@add@to+xml{\xmplinesep##1}%
1045 }%
```

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

```
1046 \let\@elt=\@elt@first
1047 \hyxmp@xmlify{#2}%
1048 \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
1049 \hyxmp@list
1050 \hyxmp@add@to+xml{</#1>^^J}%
1051 \egroup
1052 }%
1053 }
```

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

```
1054 \gdef\hyxmp@iptc@schema{%
```

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

```
1055 \ifx\hyxmp@iptc@data\@empty
1056 \else
```

```

1057 \hyxmp@add@to+xml{%
1058 -----<Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">^^J%
1059 }%

```

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

```

1060 \bgroup
1061 \edef\hyxmp@extra@indent{\hyxmp@extra@indent\space\space}%
1062 \hyxmp@list@to@lines{Iptc4xmpCore:CiAdrExtadr}{\@pdfcontactaddress}%
1063 \hyxmp@add@simple{Iptc4xmpCore:CiAdrCity}{\@pdfcontactcity}%
1064 \hyxmp@add@simple{Iptc4xmpCore:CiAdrRegion}{\@pdfcontactregion}%
1065 \hyxmp@add@simple{Iptc4xmpCore:CiAdrPcode}{\@pdfcontactpostcode}%
1066 \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.

```

1067 \def\xmplinesep{,}%
1068 \hyxmp@list@to@lines{Iptc4xmpCore:CiTelWork}{\@pdfcontactphone}%
1069 \hyxmp@list@to@lines{Iptc4xmpCore:CiEmailWork}{\@pdfcontactemail}%
1070 \hyxmp@list@to@lines{Iptc4xmpCore:CiUrlWork}{\@pdfcontacturl}%
1071 \egroup
1072 \hyxmp@add@to+xml{%
1073 -----</Iptc4xmpCore:CreatorContactInfo>^^J%
1074 }%
1075 \fi
1076 }

```

3.5.10 The PRISM Basic Metadata schema

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

```

1077 \newcommand*{\hyxmp@prism@schema}{%
1078 \ifx\hyxmp@prism@data\@empty
1079 \else
1080 \hyxmp@add@simple{prism:complianceProfile}{three}%
1081 \fi
1082 \hyxmp@add@simple@lang{prism:subtitle}{\@pdfsubtitle}%
1083 \hyxmp@add@simple@lang{prism:publicationName}{\@pdfpublication}%
1084 \hyxmp@add@simple@lang{prism:aggregationType}{\@pdfpubtype}%
1085 \hyxmp@add@simple@lang{prism:bookEdition}{\@pdfbookedition}%
1086 \hyxmp@add@simple{prism:volume}{\@pdfvolumenum}%
1087 \hyxmp@add@simple{prism:number}{\@pdfissuenum}%
1088 \hyxmp@add@simple{prism:pageRange}{\@pdfpagerange}%
1089 \hyxmp@add@simple{prism:isbn}{\@pdfisbn}%
1090 \hyxmp@add@simple{prism:issn}{\@pdfissn}%

```

```

1091 \hyxmp@add@simple{prism:eIssn}{\@pdfeissn}%
1092 \hyxmp@add@simple{prism:doi}{\@pdfdoi}%
1093 \hyxmp@add@simple{prism:url}{\@pdfurl}%
1094 \hyxmp@add@simple{prism:byteCount}{\@pdfbytes}%
1095 \hyxmp@add@simple{prism:pageCount}{\@pdfnumpages}%
1096 }

```

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.

```
1097 \newcommand*{\hyxmp@check@iptc@data}{%
```

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

```

1098 \edef\hyxmp@iptc@data{%
1099   \@pdfcontactaddress
1100   \@pdfcontactcity
1101   \@pdfcontactregion
1102   \@pdfcontactpostcode
1103   \@pdfcontactcountry
1104   \@pdfcontactphone
1105   \@pdfcontactemail
1106   \@pdfcontacturl
1107 }%
1108 }%

```

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

```
1109 \newcommand*{\hyxmp@check@prism@data}{%
```

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

```

1110 \edef\hyxmp@prism@data{%
1111   \@pdfbookedition
1112   \@pdfbytes
1113   \@pdfdoi
1114   \@pdfeissn
1115   \@pdfisbn
1116   \@pdfissn
1117   \@pdfissuenum
1118   \@pdfnumpages
1119   \@pdfpagerange
1120   \@pdfpublication
1121   \@pdfpubtype

```

```

1122 \pdfsubtitle
1123 \pdfurl
1124 \pdfvolumenum
1125 }%
1126 }%

```

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

```

1127 \newcommand*{\hyxmp@begin@extension@decls}{%
1128 \hyxmp@add@to@xml{%
1129 _____<pdfaExtension:schemas>^^J%
1130 _____<rdf:Bag>^^J%
1131 }%
1132 }

```

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

```

1133 \newcommand*{\hyxmp@end@extension@decls}{%
1134 \hyxmp@add@to@xml{%
1135 _____</rdf:Bag>^^J%
1136 _____</pdfaExtension:schemas>^^J%
1137 }%
1138 }

```

`\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.

```

1139 \newcommand*{\hyxmp@begin@ext@decl}[3]{%
1140 \hyxmp@add@to@xml{%
1141 _____<rdf:li rdf:parseType="Resource">^^J%
1142 _____<pdfaSchema:schema>#1</pdfaSchema:schema>^^J%
1143 _____<pdfaSchema:prefix>#2</pdfaSchema:prefix>^^J%
1144 _____<pdfaSchema:namespaceURI>#3</pdfaSchema:namespaceURI>^^J%
1145 _____<pdfaSchema:property>^^J%
1146 _____<rdf:Seq>^^J%
1147 }%
1148 }%

```

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

```

1149 \newcommand*{\hyxmp@end@ext@decl}{%
1150 \hyxmp@add@to@xml{%
1151 _____</rdf:Seq>^^J%
1152 _____</pdfaSchema:property>^^J%
1153 _____</rdf:li>^^J%
1154 }%
1155 }%

```

`\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.

```

1156 \newcommand{\hyxmp@declare@property}[4][Text]{%

```



```

1157 \hyxmp@add@to+xml{%
1158 -----<rdf:li rdf:parseType="Resource">^^J%
1159 -----<pdfaProperty:name>#2</pdfaProperty:name>^^J%
1160 -----<pdfaProperty:valueType>#1</pdfaProperty:valueType>^^J%
1161 -----<pdfaProperty:category>#3</pdfaProperty:category>^^J%
1162 -----<pdfaProperty:description>#4</pdfaProperty:description>^^J%
1163 -----</rdf:li>^^J%
1164 }%
1165 }%

```

`\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.

```

1166 \newcommand{\hyxmp@declare@field}[3][Text]{%
1167 \hyxmp@add@to+xml{%
1168 -----<rdf:li rdf:parseType="Resource">^^J%
1169 -----<pdfaField:name>#2</pdfaField:name>^^J%
1170 -----<pdfaField:valueType>#1</pdfaField:valueType>^^J%
1171 -----<pdfaField:description>#3</pdfaField:description>^^J%
1172 -----</rdf:li>^^J%
1173 }%
1174 }

```

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

```

1175 \newcommand*{\hyxmp@mm@extensions}{%
1176 \hyxmp@begin@ext@decl
1177 {XMP Media Management Schema}%
1178 {xmpMM}%
1179 {http://ns.adobe.com/xap/1.0/mm/}%
1180 \hyxmp@declare@property
1181 [URI]
1182 {DocumentID}%
1183 {internal}%
1184 {UUID based identifier for all versions and renditions of a document}%
1185 \hyxmp@declare@property
1186 [URI]
1187 {InstanceID}%
1188 {internal}%
1189 {UUID based identifier for specific incarnation of a document}%
1190 \hyxmp@declare@property
1191 {VersionID}%
1192 {internal}%
1193 {Document version identifier}%
1194 \hyxmp@end@ext@decl
1195 }%

```

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

```

1196 \newcommand*{\hyxmp@pdfa@id@extensions}{%
1197 \hyxmp@begin@ext@decl

```

```

1198     {PDF/A Identification Schema}%
1199     {pdfaid}%
1200     {http://www.aiim.org/pdfa/ns/id/}%
1201 \hyxmp@declare@property
1202     [Integer]%
1203     {part}%
1204     {internal}%
1205     {Part of PDF/A standard}%
1206 \hyxmp@declare@property
1207     {conformance}%
1208     {internal}%
1209     {Conformance level of PDF/A standard}%
1210 \hyxmp@end@ext@decl
1211 }%

```

`\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.

```

1212 \newcommand*{\hyxmp@iptc@extensions}{%
1213   \hyxmp@begin@ext@decl
1214     {IPTC Core Schema}%
1215     {Iptc4xmpCore}%
1216     {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}%
1217   \hyxmp@declare@property
1218     [ContactInfo]
1219     {CreatorContactInfo}
1220     {external}
1221     {Document creator's contact information}

```

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

```

1222   \hyxmp@add@to+xml{%
1223     -----</rdf:Seq>^^J%
1224     -----</pdfaSchema:property>^^J%
1225     -----<pdfaSchema:valueType>^^J%
1226     -----<rdf:Seq>^^J%
1227     -----<rdf:li rdf:parseType="Resource">^^J%
1228     -----<pdfaType:type>ContactInfo</pdfaType:type>^^J%
1229     -----<pdfaType:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaType:
1230     -----<pdfaType:prefix>Iptc4xmpCore</pdfaType:prefix>^^J%
1231     -----<pdfaType:description>%
1232         Basic set of information to get in contact with a person%
1233     -----</pdfaType:description>^^J%
1234     -----<pdfaType:field>^^J%
1235     -----<rdf:Seq>^^J%
1236   }%
1237   \hyxmp@declare@field
1238     {CiAdrCity}%
1239     {Contact information city}%

```

```

1240 \hyxmp@declare@field
1241     {CiAdrCtry}%
1242     {Contact information country}%
1243 \hyxmp@declare@field
1244     {CiAdrExtadr}%
1245     {Contact information address}%
1246 \hyxmp@declare@field
1247     {CiAdrPcode}%
1248     {Contact information local postal code}%
1249 \hyxmp@declare@field
1250     {CiAdrRegion}%
1251     {Contact information regional information such as state or province}%
1252 \hyxmp@declare@field
1253     {CiEmailWork}%
1254     {Contact information email address(es)}%
1255 \hyxmp@declare@field
1256     {CiTelWork}%
1257     {Contact information telephone number(s)}%
1258 \hyxmp@declare@field
1259     {CiUrlWork}%
1260     {Contact information Web URL(s)}%
1261 \hyxmp@add@to+xml{%
1262 -----</rdf:Seq>^^J%
1263 -----</pdfaType:field>^^J%
1264 -----</rdf:li>^^J%
1265 -----</rdf:Seq>^^J%
1266 -----</pdfaSchema:valueType>^^J%
1267 -----</rdf:li>^^J%
1268 }%
1269 }

```

\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.

```

1270 \newcommand*{\hyxmp@prism@extensions}{%
1271 \hyxmp@begin@ext@decl
1272     {PRISM Basic Metadata}%
1273     {prism}%
1274     {http://prismstandard.org/namespaces/basic/2.1/}%
1275 \hyxmp@declare@property
1276     {complianceProfile}%
1277     {internal}%
1278     {PRISM specification compliance profile to which this document adheres}%
1279 \hyxmp@declare@property
1280     {publicationName}%
1281     {external}%
1282     {Publication name}%
1283 \hyxmp@declare@property
1284     {aggregationType}%

```

```

1285     {external}%
1286     {Publication type}%
1287 \hyxmp@declare@property
1288     {bookEdition}%
1289     {external}%
1290     {Edition of the book in which the document was published}%
1291 \hyxmp@declare@property
1292     {volume}%
1293     {external}%
1294     {Publication volume number}%
1295 \hyxmp@declare@property
1296     {number}%
1297     {external}%
1298     {Publication issue number within a volume}%
1299 \hyxmp@declare@property
1300     {pageRange}%
1301     {external}%
1302     {Page range for the document within the print version of its publication}%
1303 \hyxmp@declare@property
1304     {issn}%
1305     {external}%
1306     {ISSN for the printed publication in which the document was published}%
1307 \hyxmp@declare@property
1308     {eIssn}%
1309     {external}%
1310     {ISSN for the electronic publication in which the document was published}%
1311 \hyxmp@declare@property
1312     {isbn}%
1313     {external}%
1314     {ISBN for the publication in which the document was published}%
1315 \hyxmp@declare@property
1316     {doi}%
1317     {external}%
1318     {Digital Object Identifier for the document}%
1319 \hyxmp@declare@property
1320     [URL]
1321     {url}%
1322     {external}%
1323     {URL at which the document can be found}%
1324 \hyxmp@declare@property
1325     [Integer]
1326     {byteCount}%
1327     {internal}%
1328     {Approximate file size in octets}%
1329 \hyxmp@declare@property
1330     [Integer]
1331     {pageCount}%
1332     {internal}%
1333     {Number of pages in the print version of the document}%
1334 \hyxmp@declare@property

```

```

1335     {subtitle}%
1336     {external}%
1337     {Document's subtitle}%
1338 \hyxmp@end@ext@decl
1339 }%

```

`\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.

```

1340 \newcommand*{\hyxmp@declare@extensions}{%
1341 \hyxmp@begin@extension@decls
    Declare the XMP Media Management extensions (always present).
1342 \hyxmp@mm@extensions
    Declare the PDF/A Identification extensions, but only when generating a PDF/A
    document.
1343 \ifHy@pdfa
1344     \hyxmp@mm@extensions
1345 \fi
    Conditionally declare IPTC photo metadata extensions.
1346 \ifx\hyxmp@iptc@data\@empty
1347 \else
1348     \hyxmp@iptc@extensions
1349 \fi
    Conditionally declare PRISM basic metadata extensions.
1350 \ifx\hyxmp@prism@data\@empty
1351 \else
1352     \hyxmp@prism@extensions
1353 \fi
1354 \hyxmp@end@extension@decls
1355 }

```

3.5.12 Combining schemata into an XMP packet

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

```

1356 \begingroup
1357 \ifhyxmp@unicodetex
1358     \lccode'\!="FEFF %
1359     \lowercase{%
1360         \gdef\hyxmp@bom{!}
1361     }%
1362 \else
1363     \catcode'\^^ef=12
1364     \catcode'\^^bb=12
1365     \catcode'\^^bf=12
1366     \gdef\hyxmp@bom{^^ef^^bb^^bf}%
1367 \fi
1368 \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.

```

1369 \def\hyxmp@construct@packet{%
1370 \gdef\hyxmp+xml{%
1371 \hyxmp@add@to+xml{<?xpacket begin="\hyxmp@bom" %
1372 id="W5M0MPCehiHzreSzNTczkc9d"?>^^J%
1373 <x:xmpmeta xmlns:x="adobe:ns:meta/">^^J%
1374 __<rdf:RDF %
1375 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\hyxmp@hash">^^J%
1376 ____<rdf:Description rdf:about="">^^J%

```

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

```

1377 _____xmlns:pdf="http://ns.adobe.com/pdf/1.3/"^^J%
1378 _____xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/"^^J%
1379 _____xmlns:dc="http://purl.org/dc/elements/1.1/"^^J%
1380 _____xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/"^^J%
1381 _____xmlns:xmp="http://ns.adobe.com/xap/1.0/"^^J%
1382 _____xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"^^J%
1383 _____xmlns:stEvt="http://ns.adobe.com/xap/1.0/sType/ResourceEvent\hyxmp@hash"^^J%
1384 _____xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/"^^J%
1385 _____xmlns:prism="http://prismstandard.org/namespaces/basic/2.1/"^^J%
1386 _____xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"^^J%
1387 _____xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"^^J%
1388 _____xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema\hyxmp@hash"^^J%
1389 _____xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property\hyxmp@hash"^^J%
1390 _____xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type\hyxmp@hash"^^J%
1391 _____xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field\hyxmp@hash">^^J%
1392 }%

```

Declare non-standard schemata.

```

1393 \hyxmp@check@iptc@data
1394 \hyxmp@check@prism@data
1395 \hyxmp@declare@extensions

```

Insert all the metadata we know how to insert.

```

1396 \hyxmp@pdf@schema
1397 \hyxmp@xmpRights@schema
1398 \hyxmp@dc@schema
1399 \hyxmp@photoshop@schema
1400 \hyxmp@xmp@basic@schema
1401 \hyxmp@pdfa@id@schema
1402 \hyxmp@mm@schema
1403 \hyxmp@iptc@schema
1404 \hyxmp@prism@schema
1405 \hyxmp@add@to+xml{%
1406 ____</rdf:Description>^^J%
1407 __</rdf:RDF>^^J%
1408 </x:xmpmeta>^^J%
1409 \hyxmp@padding
1410 <?xpacket end="w"?>^^J%

```

```

1411 }%
1412 }

```

3.6 Embedding the XMP packet

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

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

```

1413 \newcommand*{\hyxmp@embed@packet}{%
1414   \hyxmp@construct@packet
1415   \def\hyxmp@driver{hpdfTeX}%
1416   \ifx\hyxmp@driver\Hy@driver
1417     \hyxmp@embed@packet@pdfTeX
1418   \else
1419     \def\hyxmp@driver{hLuaTeX}%
1420     \ifx\hyxmp@driver\Hy@driver
1421       \hyxmp@embed@packet@LuaTeX
1422     \else
1423       \def\hyxmp@driver{hdvipdfm}%
1424       \ifx\hyxmp@driver\Hy@driver
1425         \hyxmp@embed@packet@dvipdfm
1426       \else
1427         \def\hyxmp@driver{hXeTeX}%
1428         \ifx\hyxmp@driver\Hy@driver
1429           \hyxmp@embed@packet@XeTeX
1430         \else
1431           \@ifundefined{pdfmark}{%
1432             \PackageWarningNoLine{hyperxmp}{%
1433               Unrecognized hyperref driver ‘\Hy@driver’.\MessageBreak
1434               \jobname.tex’s XMP metadata will *not* be\MessageBreak
1435               embedded in the resulting file}%
1436           }{%
1437             \hyxmp@embed@packet@pdfmark
1438           }%
1439         \fi
1440       \fi
1441     \fi
1442   \fi
1443 }

```

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.

```
1444 \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.

```
1445 \newcommand*{\hyxmp@embed@packet@pdftex}{%
1446   \bgroup
1447   \ifluatex
1448   \else
1449     \pdfcompresslevel=0
1450   \fi
1451   \immediate\pdfobj \ifluatex uncompressed\fi stream attr {%
1452     /Type /Metadata
1453     /Subtype /XML
1454   }{\hyxmp@xml}%
1455   \pdfcatalog {/Metadata \the\pdflastobj\space 0 R}%
1456   \egroup
1457 }
```

3.6.2 Embedding using LuaTeX 0.85+

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

```
1458 \newcommand*{\hyxmp@embed@packet@luatex}{%
1459   \immediate\pdfextension obj uncompressed stream attr {%
1460     /Type /Metadata
1461     /Subtype /XML
1462   }{\hyxmp@xml}%
1463   \pdfextension catalog {/Metadata \the\numexpr\pdffeedback lastobj\relax\space 0 R}%
1464 }
```

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.

```
1465 \newcommand*{\hyxmp@embed@packet@pdfmark}{%
1466   \pdfmark{%
1467     pdfmark=/NamespacePush
```



```

1468 }%
1469 \pdfmark{%
1470   pdfmark=/OBJ,
1471   Raw={/_objdef \string{hyxmp@Metadata\string} /type /stream}%
1472 }%
1473 \pdfmark{%
1474   pdfmark=/PUT,
1475   Raw={\string{hyxmp@Metadata\string}
1476     2 dict begin
1477       /Type /Metadata def
1478       /Subtype /XML def
1479       currentdict
1480     end
1481 }%
1482 }%
1483 \pdfmark{%
1484   pdfmark=/PUT,
1485   Raw={\string{hyxmp@Metadata\string} (\hyxmp@xml)}%
1486 }%
1487 \pdfmark{%
1488   pdfmark=/Metadata,
1489   Raw={\string{Catalog\string} \string{hyxmp@Metadata\string}}%
1490 }%
1491 \pdfmark{%
1492   pdfmark=/NamespacePop
1493 }%
1494 }

```

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.

```

1495 \newcommand*{\hyxmp@embed@packet@dvipdfm}{%
1496   \hyxmp@string@len{\hyxmp@xml}%
1497   \special{pdf: object @hyxmp@Metadata
1498     <<
1499       /Type /Metadata
1500       /Subtype /XML
1501       /Length \the\@tempcnta
1502     >>
1503     stream^^J\hyxmp@xml endstream%
1504 }%
1505   \special{pdf: docview
1506     <<
1507       /Metadata @hyxmp@Metadata
1508     >>
1509 }%
1510 }

```

`\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.

```
1511 \newcommand*{\hyxmp@string@len}[1]{%
1512   \@tempcnta=0
1513   \expandafter\hyxmp@count@spaces#1 {} %
1514   \expandafter\hyxmp@count@non@spaces#1{}%
1515 }
```

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

```
1516 \def\hyxmp@count@spaces#1 {%
1517   \def\hyxmp@one@token{#1}%
1518   \ifx\hyxmp@one@token\empty
1519     \advance\@tempcnta by -1
1520   \else
1521     \advance\@tempcnta by 1
1522     \expandafter\hyxmp@count@spaces
1523   \fi
1524 }
```

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

```
1525 \newcommand*{\hyxmp@count@non@spaces}[1]{%
1526   \def\hyxmp@one@token{#1}%
1527   \ifx\hyxmp@one@token\empty
1528     \else
1529       \advance\@tempcnta by 1
1530       \expandafter\hyxmp@count@non@spaces
1531     \fi
1532 }
```

3.6.5 Embedding using X_qTeX

`\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.

```
1533 \newcommand*{\hyxmp@embed@packet@xetex}{%
1534   \special{pdf:stream @hyxmp@Metadata (\hyxmp@xml)
1535     <<
1536       /Type /Metadata
1537       /Subtype /XML
1538     >>
1539   }%
```

```

1540 \special{pdf:put @catalog
1541   <<
1542     /Metadata @hyxmp@Metadata
1543   >>
1544 }%
1545 }

```

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.

```

1546 \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 8–9. 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:prism="http://prismstandard.org/namespaces/basic/3.0/"
      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 100 lines of boilerplate definitions not shown]
    :
  </rdf:Bag>
</pdfaExtension:schemas>
<pdf:Keywords>
  energy quanta, Hertz effect, quantum physics
</pdf:Keywords>
<pdf:Producer>pdfTeX-1.40.19</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>
<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>56884</prism:byteCount>
<prism:pageCount>17</prism:pageCount>
</rdf:Description>
</rdf:RDF>
</x:xmpmeta>
<?xpacket end="w"?>

```

References

- [1] Adobe Systems, Inc., San Jose, California. *Adobe Acrobat X SDK Help, pdfmark Reference*. Available from <http://www.adobe.com/devnet/acrobat/documentation.html>.
- [2] Adobe Systems, Inc. *PostScript Language Reference Manual*. Addison-Wesley, 2nd edition, January 1996, ISBN: 0-201-18127-4.
- [3] Adobe Systems, Inc., San Jose, California. *Document Management—Portable Document Format—Part 1: PDF 1.7*, July 2008. ISO 32000-1 standard document. Available from http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000_2008.pdf.
- [4] Adobe Systems, Inc., San Jose, California. *XMP Specification Part 1: Data model, Serialization, and Core Properties*, April 2012. Available from <http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/xmp/pdfs/cc-201306/XMPSpecificationPart1.pdf>.
- [5] DCMI Usage Board *DCMI Metadata Terms*, June 14, 2012. Available from <http://dublincore.org/documents/dcmi-terms/>.
- [6] Michael Downes. Around the bend #15, answers, 4th (last) installment. `comp.text.tex` newsgroup posting, January 3, 1994. Archived by Google at <http://groups.google.com/group/comp.text.tex/msg/7da7643b9e8f3b48>.
- [7] International Digital Enterprise Alliance, Inc. *Publishing Requirements for Industry Standard Metadata, Version 3.0: PRISM Basic Metadata Specification*, October 12, 2012. Available from http://www.prismstandard.org/specifications/3.0/PRISM_Basic_Metadata_3.0.htm.
- [8] International Digital Enterprise Alliance, Inc. *Publishing Requirements for Industry Standard Metadata, Version 3.0: PRISM Controlled Vocabularies Specification*, October 4, 2012. Available from http://www.prismstandard.org/specifications/3.0/PRISM_CV_Spec_3.0.pdf.
- [9] International Press Telecommunications Council. *IPTC Photo Metadata: Core 1.1/Extension 1.1*, July 2010. Revision 1. Available from http://www.iptc.org/std/photometadata/specification/IPTC-PhotoMetadata-201007_1.pdf.
- [10] Internet Assigned Numbers Authority. Language subtag registry, January 11, 2011. Available from <http://www.iana.org/assignments/language-subtag-registry>.
- [11] Paul J. Leach, Michael Mealling, and Rich Salz. A Universally Unique Identifier (UUID) URN namespace. Request for Comments 4122, Internet Engineering Task Force, Network Working Group, July 2005. Category: Standards Track. Available from <http://www.ietf.org/rfc/rfc4122.txt>.

- [12] PDF/A Competence Center, Berlin, Germany. *TechNote 0008: Predefined XMP Properties in PDF/A-1*, March 20, 2008. Available from http://www.pdfa.org/wp-content/uploads/2011/08/tn0008_predefined_xmp_properties_in_pdfa-1_2008-03-20.pdf.
- [13] PDF/A Competence Center, Berlin, Germany. *TechNote 0009: XMP Extension Schemas in PDF/A-1*, March 20, 2008. Available from http://www.pdfa.org/wp-content/uploads/2011/08/tn0009_xmp_extension_schemas_in_pdfa-1_2008-03-20.pdf.
- [14] Misha Wolf and Charles Wicksteed. Date and time formats. Note NOTE-datetime, World Wide Web Consortium (W3C), September 15, 1997. Available from <http://www.w3.org/TR/NOTE-datetime>.

Change History

v1.0		wrote the document’s metadata	25
	General: Initial version	1 v1.4
v1.1	<code>\hyxmp@construct@packet:</code>		
	Explicitly set the category		
	codes of characters $\langle EF \rangle$, $\langle BB \rangle$,	<code>\hyxmp@rdf@dc:</code> Included metadata	
	and $\langle BF \rangle$ to “letter”. Thanks	in the <code>x-default</code> language	
	to Daniel Schömer for the bug	regardless of the specified	
	report	metadata language	47
		62
v1.2	<code>\hyxmp@xmpRights@schema:</code>		
	General: Added support for the	Renamed the <code>xapRights</code>	
	<code>X_YT_EX</code> backend (<code>xdvipdfmx</code>)	namespace prefix to <code>xmpRights</code>	50
		1
	Added support for the	v1.5	
	Photoshop schema	General: Made the XMP inclusion	
	more robust. Thanks to Heiko	
	Made the package compatible	Oberdiek for the bug report	
	with <code>ngerman</code> . Thanks to	and suggested modifications.	16
	Tobias Mueller for the bug		
	report.	v2.0	
		16
v1.3	<code>\ProcessKeyvalOptions:</code> Added		
	<code>\hyxmp@reencode:</code> Introduced this	this macro	23
	macro to re-encode Unicode		
	strings as 8-bit strings before	<code>\XMPTruncateList:</code> Added this	
	manipulating them into XMP	macro	28
	schema. This change addresses		
	a bug reported by Martin	<code>\hyxmp@ProcessKeyvalOptions:</code>	
	Münch	Added this macro	23
		34
	General: Introduced the	<code>\hyxmp@SpaceOther:</code> Added by	
	<code>pdfmetalang</code> package option,	Heiko Oberdiek	38
	which enables an author to		
	specify the language in which he	<code>\hyxmp@add@simple:</code> Added this	
		macro	46
		<code>\hyxmp@add@to@xml:</code> Updated also	
		to replace commas	44

<code>\hyxmp@bom</code> : Added by Heiko Oberdiek	61	<code>\hyxmp@zero</code> : Added by Heiko Oberdiek	39
<code>\hyxmp@comma</code> : Added this macro	27	<code>\ifhyxmp@unicodetex</code> : Added by Heiko Oberdiek	34
<code>\hyxmp@construct@packet</code> : Modified by Heiko Oberdiek to use an appropriate BOM representation via <code>\hyxmp@bom</code>	62	<code>\xmpcomma</code> : Added this macro . . .	27
<code>\hyxmp@crap@convert</code> : Added by Heiko Oberdiek	38	<code>\xmpquote</code> : Added this macro . . .	28
<code>\hyxmp@crap@test</code> : Added by Heiko Oberdiek	38	General: Added support for the XMP Basic schema and miscellaneous other bits of metadata	1
<code>\hyxmp@dc@schema</code> : Added support for <code>dc:language</code> and <code>dc:source</code> .	49	Heiko Oberdiek's major rewrite of the code to better support native-Unicode <code>T_EX</code> implementations (<code>X_YT_EX</code> and <code>LuaT_EX</code>)	1
<code>\hyxmp@is@unicode</code> : Added by Heiko Oberdiek	35	New <code>\AtBeginDocument</code> code from Heiko Oberdiek to properly encode <code>\@pdfmetalang</code>	25
<code>\hyxmp@list@to+xml</code> : Modified by Heiko Oberdiek to use the new Unicode-processing macros . .	48	v2.1	
<code>\hyxmp@photoshop@schema</code> : Simplified using <code>\hyxmp@add@simple</code>	52	<code>\hypersetup</code> : Added this macro .	24
<code>\hyxmp@reencode</code> : Replaced with an empty macro by Heiko Oberdiek	34	<code>\hyxmp@hypersetup</code> : Added this macro	24
<code>\hyxmp@skiptorelax</code> : Added by Heiko Oberdiek	38	<code>\hyxmp@redefine@Hyp</code> : Added this macro	22
<code>\hyxmp@skipzeros</code> : Added by Heiko Oberdiek	37	General: Enabled <code>hyperxmp</code> and <code>hyperref</code> to be loaded in either order. This addresses a bug report by Yury Donskoy	22
<code>\hyxmp@toxml</code> : Added by Heiko Oberdiek	36	v2.2	
Escaped parentheses written with <code>pdfmarks</code> to prevent <code>dvips</code> from line-wrapping the XMP packet	36	<code>\hyxmp@iptc@extensions</code> : Added this macro to support PDF/A generation	58
<code>\hyxmp@toxml@unicodetex</code> : Added by Heiko Oberdiek	37	<code>\hyxmp@iptc@schema</code> : Added this macro	53
<code>\hyxmp@xetex@crap</code> : Added by Heiko Oberdiek	37	<code>\hyxmp@list@to@lines</code> : Added this macro	53
<code>\hyxmp@xmlify</code> : Completely rewritten by Heiko Oberdiek to better support Unicode-enabled <code>T_EX</code> programs	34	<code>\xmpcomma</code> : Changed the default from <code>\relax</code> to an ordinary comma	27
<code>\hyxmp@xmp@basic@schema</code> : Added this macro	51	<code>\xmplinesep</code> : Added this macro .	53
<code>\hyxmp@xmpRights@schema</code> : Modified to include <code>xmpRights:Marked</code> only when <code>pdfcopyright</code> is specified and <code>xmpRights:WebStatement</code> only when <code>pdflicenseurl</code> is specified .	50	General: Added support for the IPTC Photo Metadata schema .	1
		v2.3	
		<code>\hyxmp@iptc@extensions</code> : Gave the <code>lptc4xmpCore:CreatorContactInfo</code> fields a unique <code>pdfaType:prefix</code> to better support conversion of the document to PDF/A	58

v2.3a	General: Bug fix: Redefine <code>\pdflang</code> as <code>\empty</code> when <code>hyperref</code> has set it to <code>\relax</code>	25	General: Added support for the PDF/A Identification schema, as requested by Florian Breitwieser	1
v2.3b	<code>\XMPTruncateList</code> : Made all definitions local to avoid spurious <code>Too many</code> <code>unprocessed floats</code> errors when running with <code>memoir</code>	28	v2.5 <code>\hyxmp@add@to@xml</code> : Updated also to replace underscores and to modify only the text being added, not the already-modified text	44
v2.4	<code>\hyxmp@add@simple@var</code> : Added this macro	46	<code>\hyxmp@textunderscore</code> : Added this macro	17
	<code>\hyxmp@create@uuid</code> : Modified this macro to produce a proper version 4 (random or pseudorandom) UUID	42	<code>\hyxmp@uscore</code> : Added this macro	28
	<code>\hyxmp@dc@schema</code> : Made <code>dc:language</code> a <code>Bag</code> instead of an individual item so as to conform to the latest XMP specifications, a detail identified by Florian Breitwieser	49	General: Enabled “ <code>_</code> ” to work within email addresses, as requested by Leonid Sinev	1
	<code>\hyxmp@parse@time</code> : Added this macro	29	v2.6 General: Added support for a new <code>pdfdate</code> key to explicitly specify the document date (and optionally time)	1
	<code>\hyxmp@parse@tz</code> : Added this macro	30	v2.7 General: Automatically use <code>\title</code> and <code>\author</code> if <code>pdftitle</code> and <code>pdfauthor</code> are left unspecified. Thanks to Maciej Radziejewski for the suggestion	26
	<code>\hyxmp@parse@tz@char</code> : Added this macro	29	v2.8 <code>\hyxmp@add@to@xml</code> : Corrected inadvertent lowercasing of non-Latin characters when run under <code>X_YL^AT_EX</code> or <code>Lua^AT_EX</code> (bug reported by Leonid Sinev)	44
	<code>\hyxmp@pdf@schema</code> : Made <code>\hyxmp@pdf@schema</code> <i>always</i> include the Adobe PDF schema, even when empty. Florian Breitwieser noted that this is necessary for PDF/A-1b compliance	45	v2.9 <code>\hyxmp@iptc@schema</code> : Use <code>lptc4xmpCore</code> instead of <code>lptc4ContInfo</code> as the contact-information metadata prefix. Leonid Sinev reports that Acrobat’s PDF/A validator seems to prefer <code>lptc4xmpCore</code>	53
	<code>\hyxmp@pdf@to@xmp@date</code> : Added this macro	29	<code>\hyxmp@pdfa@id@schema</code> : Let the author specify the PDF/A part and conformance IDs, as requested by Leonid Sinev	52
	<code>\hyxmp@pdfa@id@schema</code> : Added this macro	52	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	
	<code>\hyxmp@today</code> : Modified the code to parse the time and timezone from <code>\pdfcreationdate</code> , as proposed by Florian Breitwieser	33		
	<code>\hyxmp@today@define</code> : Added this macro	32		
	<code>\hyxmp@xmp@to@pdf@date</code> : Added this macro	30		
	<code>\xmptilde</code> : Added this macro	28		

	with the <code>pdfa</code> option (suggested by Leonid Sinev)	1		and the corresponding <code>pdfsource</code> option, at Niklas Beisert's request	19
	Introduced the <code>pdftype</code> package option, which enables an author to specify the type of document being produced	1		<code>\XMLLangAlt</code> : Added this macro based on a request—and some code—by Niklas Beisert to support metadata expressed in multiple languages	40
v3.0	<code>\hyxmp@embed@packet@luatex</code> : Added this macro	64		<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	47
	<code>\hyxmp@today@define</code> : Modified to accept the name of a macro to define	32		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	25
	<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>	51			
	General: Made the code compatible with LuaTeX 0.85. Thanks to Robert Schlicht, Leonid Sinev, and David Carlisle for bug reports and to Leonid Sinev for helping test the new <code>hyperxmp</code> code	1	v3.4	<code>\hyxmp@seed@string</code> : Correctly handle an author field of all spaces. Bug reported by Gaëtan Leurent	43
v3.1	<code>\hyxmp@embed@packet@luatex</code> : Updated to use <code>\pdfextension obj uncompressed</code> as suggested by Hans Hagen	64		General: Use <code>ifmtarg</code> to test for empty arguments, including non-empty but all spaces	1
	<code>\hyxmp@embed@packet@pdftex</code> : Leave the XMP packet—and only the XMP packet—uncompressed in both pdfTeX and pre-0.85 LuaTeX	64	v3.5	<code>\hyxmp@DocumentID</code> : Added the <code>pdfdocumentid</code> option, at Michael Osipov's request	19
v3.2	<code>\hyxmp@as@pdf@date</code> : Added this macro	30		<code>\hyxmp@InstanceID</code> : Added the <code>pdfinstanceid</code> option, at Michael Osipov's request	19
	<code>\hyxmp@as@xmp@date</code> : Added this macro	29		<code>\hyxmp@mm@schema</code> : Generate <code>\hyxmp@DocumentID</code> and <code>\hyxmp@InstanceID</code> only if the document does not already define these using the <code>pdfdocumentid</code> and <code>pdfinstanceid</code> options	50
	<code>\hyxmp@today@define</code> : Modified to include hours and minutes	32		<code>\hyxmp@seed@string</code> : Seed with the TeX timestamp in addition to the document-specified timestamp	43
	<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	51	v4.0	<code>\XMPTruncateList</code> : Deprecated this macro	28
v3.3	<code>\@pdfsource</code> : Added this macro			<code>\hyxmp@add@simple@lang</code> : Added this macro	46

108, 189, 1063, 1100	\@pdfpubtype	CiUrlWork 3
\@pdfcontactcountry	72, 212, 1084, 1121	CreationDate 75
114, 190, 1066, 1103	\@pdfsource 62, 945, 947	
\@pdfcontactemail . .	\@pdfsubject . . 213, 937	D
118, 191, 1069, 1105	\@pdfsubtitle	Date 33
\@pdfcontactphone . .	98, 214, 1082, 1122	\day 418, 419, 421
116, 192, 1068, 1104	\@pdftitle 215,	dc:creator . . 2, 10, 49, 74
\@pdfcontactpostcode	246, 757, 765, 936	dc:date 2, 49
112, 193, 1065, 1102	\@pdftype . . 48, 216, 942	dc:description
\@pdfcontactregion .	\@pdfurl 3, 10, 39, 49, 74
110, 194, 1064, 1101	96, 217, 1093, 1123	dc:format 2
\@pdfcontacturl . . .	\@pdfversionid	dc:language . . 3, 49, 73, 74
120, 195, 1070, 1106 68, 218, 974	dc:publisher 3
\@pdfcopyright . 46,	\@pdfvolumenum	dc:rights . . . 2, 15, 39, 49
196, 938, 956, 962	88, 219, 1086, 1124	dc:source 2, 49, 73
\@pdfcreationdate . .	\@tempwafalse 862, 897	dc:subject 2, 49
. . . . 197, 992, 996	\@tempwatrue	dc:title . . 3, 10, 39, 49, 74
\@pdfcreator 1014	. 862, 864, 897, 899	dc:type 3
\@pdfdatetimestamp	\@title 247, 248	DCMI 7
. 24, 198, 242, 244	\^ 289, 293, 458,	\define@key . . 25, 36,
\@pdfdoi	785, 786, 1363–1365	47, 49, 51, 53, 55,
94, 199, 1092, 1113	_ 784, 786	57, 59, 61, 63, 65,
\@pdfeissn	\~ 298, 592, 593	67, 69, 71, 73, 75,
80, 200, 1091, 1114		77, 79, 81, 83, 85,
\@pdfisbn	\sq 558, 593, 784	87, 89, 91, 93, 95,
82, 201, 1089, 1115		97, 99, 101, 109,
\@pdfissn	A	111, 113, 115,
78, 202, 1090, 1116	\and 137	117, 119, 121,
\@pdfissuenumber	ASCII 17, 35	125, 137, 158, 660
90, 203, 1087, 1117	\AtBeginDocument . . . 229	\do 662, 669, 883
\@pdfkeywords . 158, 204	\AtEndDocument 5	DOI 2, 6, 20
\@pdflang . . 205, 231,	\AtEndDvi 8	dvipdf (option) 64
232, 235, 238, 941	atenddvi 16	dvipdfm 65
\@pdflicenseurl . . .	Author 9, 10, 21	dvips (option) 64
. 50, 206, 952, 966		dvips 9, 36, 73
\@pdfmetadatetimestamp . .	B	dvipsone (option) 64
35, 207, 1009, 1012	Bag 74	dviwindo (option) . . . 64
\@pdfmetalang	baseurl (option)	
. . 56, 234, 236,	4, 6, 13, 17, 20, 51	E
238, 241, 873, 875	BOM 61, 73	\EdefEscapeHex 481, 494
\@pdfmoddate		\EdefUnescapeHex . . . 498
. . 208, 1002, 1006	C	\EdefUnescapeString 468
\@pdfnumpages	CiAdrCity 2	ETX 27, 34
76, 209, 1095, 1118	CiAdrCtry 2	
\@pdfpagerange	CiAdrExtadr 2	G
92, 210, 1088, 1119	CiAdrPcode 2	Ghostsript 9
\@pdfpublication	CiAdrRegion 2	gitver 6
70, 211, 1083, 1120	CiEmailWork 2	
\@pdfpublisher . 86, 939	CiTelWork 3	

H	
\Hy@driver	\hyxmp@alt@title 656, 665
. 4, 1416, 1420,	\hyxmp@and 137
1424, 1428, 1433	\hyxmp@append@hex . .
\Hy@unicodedefalse . 27, 38	. 705, 724–726, 730
hyperref 1, 4, 5, 7, 8, 10,	\hyxmp@append@hex@iii
13, 17, 21, 22, 24–	. 723, 729, 739, 750
27, 45, 63, 64, 73–76	\hyxmp@append@hex@iv
\hypersetup 175, 248, 254 728, 734,
hyperxmp 1, 2, 4, 5, 7, 8,	735, 737, 752–754
10, 12–18, 21, 22,	\hyxmp@as@pdf@date . 345
25–27, 29, 34, 41,	\hyxmp@as@xmp@date .
45, 55, 63, 67, 73, 75 30,
\hyxmp@is@unicode . 502	41, 317, 996, 1006
\hyxmp@add@simple . .	\hyxmp@at@end . . . 3, 259
. 814, 817,	\hyxmp@begin@ext@decl
821, 935, 947,	. . . 1139, 1176,
964, 966, 972–	1197, 1213, 1271
974, 990, 993,	\hyxmp@begin@extension@decls
995, 1000, 1003, 1127, 1341
1005, 1010, 1012,	\hyxmp@big@prime . . .
1014, 1015, 1019,	. 679, 682, 692, 702
1020, 1024, 1025,	\hyxmp@big@prime@ii
1063–1066, 1080, 679, 701
1084, 1086–1095	\hyxmp@bom . . . 1356, 1371
\hyxmp@add@simple@lang	\hyxmp@check@iptc@data
. 838, 1097, 1393
1082, 1083, 1085	\hyxmp@check@prism@data
\hyxmp@add@simple@lang@i 1109, 1394
. 841, 844	\hyxmp@comma
\hyxmp@add@simple@lang@ii	. 102, 138, 159, 288
. 845, 847	\hyxmp@commas@to@list
\hyxmp@add@simple@var	272, 308, 908, 1048
. . . . 809, 810, 829	\hyxmp@commas@to@list@i
\hyxmp@add@to+xml 274, 276
. 771,	\hyxmp@concat@metadata
797, 824, 833, 180
851, 855, 869,	\hyxmp@construct@packet
876, 880, 885, 1369, 1414
890, 902, 910,	\hyxmp@count@non@spaces
916, 925, 1036, 1514, 1525
1040, 1044, 1050,	\hyxmp@count@spaces
1057, 1072, 1128, 1513, 1516
1134, 1140, 1150,	\hyxmp@crap@convert
1157, 1167, 1222, 584, 618
1261, 1371, 1405	\hyxmp@crap@result .
\hyxmp@alt@description 574, 610
. 656, 666	\hyxmp@crap@test 581, 606
\hyxmp@alt@rights . .	\hyxmp@create@uuid .
. 656, 667 732, 760, 769
	\hyxmp@createdate . .
	. 976, 990, 993,
	1000, 1003, 1010
	\hyxmp@cur@lang 662, 670
	\hyxmp@dc@schema . . .
 934, 1398
	\hyxmp@declare@extensions
 1340, 1395
	\hyxmp@declare@field
 1166,
	1237, 1240, 1243,
	1246, 1249,
	1252, 1255, 1258
	\hyxmp@declare@property
 1156,
	1180, 1185, 1190,
	1201, 1206, 1217,
	1275, 1279, 1283,
	1287, 1291, 1295,
	1299, 1303, 1307,
	1311, 1315, 1319,
	1324, 1329, 1334
	\hyxmp@def@DocumentID
 756, 970
	\hyxmp@def@InstanceID
 762, 971
	\hyxmp@define@createdate
 976, 988
	\hyxmp@DocumentID . .
	. 64, 756, 970, 972
	\hyxmp@dq@code . 1, 1546
	\hyxmp@driver . . 3, 1413
	\hyxmp@embed@packet
 261, 1413
	\hyxmp@embed@packet@dvi
 1425, 1495
	\hyxmp@embed@packet@luatex
 1421, 1458
	\hyxmp@embed@packet@pdfmark
 1437, 1465
	\hyxmp@embed@packet@pdftex
 1417, 1445
	\hyxmp@embed@packet@xetex
 1429, 1533
	\hyxmp@end@ext@decl
 1149,
	1194, 1210, 1338
	\hyxmp@end@extension@decls
 1133, 1354

<code>\hyxmp@extra@indent</code>	<code>\hyxmp@parse@tz</code> . . .	<code>\hyxmp@seed@rng@i</code> . .
. 820 , 335 , 338 , 342 683 , 685
825 , 834 , 1037 , 1061	<code>\hyxmp@parse@tz@char</code>	<code>\hyxmp@seed@string</code> .
<code>\hyxmp@find@metadata</code> 330 , 332 756 , 762
. 180 , 260	<code>\hyxmp@pdf@schema</code> . .	<code>\hyxmp@set@rand@num</code>
<code>\hyxmp@first@char</code> . . 315 808 , 1396 698 , 706 , 741
<code>\hyxmp@first@char@i</code>	<code>\hyxmp@pdf@to@xmp@date</code>	<code>\hyxmp@singleton@dc</code>
. . . . 315 , 318 , 346 319 , 324 , 922 , 939 – 942
<code>\hyxmp@gobbletwo</code> 385 , 398	442 , 445 , 981 , 984	<code>\hyxmp@skiptorelax</code> .
<code>\hyxmp@hash</code> 791 , 1375 ,	<code>\hyxmp@pdfa@id@extensions</code> 611 , 617
1383 , 1388 – 1391 1196	<code>\hyxmp@skipzeros</code> . . . 569
<code>\hyxmp@Hyp@pdfauthor</code> 131	<code>\hyxmp@pdfa@id@schema</code>	<code>\hyxmp@SpaceOther</code> . .
<code>\hyxmp@Hyp@pdfkeywords</code> 1022 , 1401 578 , 591
. 152	<code>\hyxmp@pdfauthor</code> . . .	<code>\hyxmp@string@len</code> . .
<code>\hyxmp@hypersetup</code> . . 175 128 , 137 , 943 1496 , 1511
<code>\hyxmp@InstanceID</code> . .	<code>\hyxmp@pdfkeywords</code> .	<code>\hyxmp@sublist</code>
. . 66 , 762 , 971 , 973 128 , 158 , 944	. . 277 , 278 , 281 , 282
<code>\hyxmp@iptc@data</code> . . .	<code>\hyxmp@pdfstringdef</code>	<code>\hyxmp@suppress@pdf@metadata</code>
. . 1055 , 1098 , 1346 18 , 29 , 40 , 122 , 258
<code>\hyxmp@iptc@extensions</code>	47 , 49 , 51 , 53 , 55 ,	<code>\hyxmp@temp@list</code> . . . 301
. 1212 , 1348	57 , 59 , 61 , 63 , 65 ,	<code>\hyxmp@temp@str</code> . . . 301
<code>\hyxmp@iptc@schema</code> .	67 , 69 , 71 , 73 , 75 ,	<code>\hyxmp@text</code>
. 1054 , 1403	77 , 79 , 81 , 83 , 85 ,	. . 466 , 544 , 574 , 618
<code>\hyxmp@is@unicode</code> . .	87 , 89 , 91 , 93 , 95 ,	<code>\hyxmp@textunderscore</code>
. . . . 470 , 487 , 502	97 , 99 , 104 , 109 , 18
<code>\hyxmp@LA@accept</code> . . .	111 , 113 , 115 ,	<code>\hyxmp@today</code>
. . . . 659 , 665 – 667	117 , 119 , 121 , 661	. . 244 , 438 , 765 , 940
<code>\hyxmp@legal</code> 951	<code>\hyxmp@photoshop@data</code>	<code>\hyxmp@today@define</code>
<code>\hyxmp@list</code> 1017	. . 411 , 440 , 763 , 979
908 , 914 , 1048 , 1049	<code>\hyxmp@photoshop@schema</code>	<code>\hyxmp@toxml</code> . . 496 , 519
<code>\hyxmp@list@to@lines</code> 1017 , 1399	<code>\hyxmp@toxml@unicodetex</code>
. 1033 ,	<code>\hyxmp@prism@data</code> 484 , 544
1062 , 1068 – 1070	. . 1078 , 1110 , 1350	<code>\hyxmp@trimb</code> . . 451 , 454
<code>\hyxmp@list@to@xml</code> .	<code>\hyxmp@prism@extensions</code>	<code>\hyxmp@trimc</code> . . 454 , 455
. . . . 896 , 943 , 944 1270 , 1352	<code>\hyxmp@trimspace</code> . .
<code>\hyxmp@mm@extensions</code>	<code>\hyxmp@prism@schema</code> 281 , 447
. . 1175 , 1342 , 1344 1077 , 1404	<code>\hyxmp@try</code> 574
<code>\hyxmp@mm@schema</code> . . .	<code>\hyxmp@ProcessKeyvalOptions</code>	<code>\hyxmp@unicodetexfalse</code>
. 969 , 1402 170 457
<code>\hyxmp@modulo@a</code> 673 ,	<code>\hyxmp@rand@num</code> 698 ,	<code>\hyxmp@unicodetextrue</code>
692 , 702 , 708 , 743	707 , 742 , 759 , 768 457
<code>\hyxmp@new@xml</code> 787 , 788	<code>\hyxmp@rdf@dc</code>	<code>\hyxmp@uscore</code> . . 20 , 292
<code>\hyxmp@num</code> 618 861 , 936 – 938	<code>\hyxmp@value</code> . . 661 , 868
<code>\hyxmp@one@token</code> . . .	<code>\hyxmp@redefine@Hyp</code>	<code>\hyxmp@x@default</code> . . .
. . 681 , 685 , 1517 , 130 , 172 , 177	. . 236 , 807 , 873 , 881
1518 , 1526 , 1527	<code>\hyxmp@reencode</code> . . . 463	<code>\hyxmp@xetex@crap</code> . .
<code>\hyxmp@padding</code> 795 , 1409	<code>\hyxmp@rights</code> 475 , 574
<code>\hyxmp@parse@time</code> 951 , 954 , 958 , 960	<code>\hyxmp@xml</code>
. 326 , 328	<code>\hyxmp@seed@rng</code> 788 , 795 , 1369 ,
 681 , 758 , 767	

1454, 1462, 1485, 1496, 1503, 1534	Info 9, 10, 12, 21, 26	dvipdf 64
<code>\hyxmp@xmlified</code>	<code>intcalc</code> 17	<code>dvips</code> 64
. 466 , 825, 834, 841, 852, 856, 868, 877, 886, 908, 928, 1048	<code>\intcalcDiv</code> 623, 630, 637	<code>dvipsone</code> 64
<code>\hyxmp@xmlify</code>	<code>\intcalcMod</code> 625, 632, 639	<code>dviwindo</code> 64
. 241, 466 , 823, 832, 840, 849, 867, 875, 884, 907, 924, 1047	IPTC 12, 20, 44, 53–55, 58, 61, 73	<code>keeppdfinfo</code> . . . 12, 21
<code>\hyxmp@xmp@basic@schema</code> 987 , 1400	<code>lptc4xmpCore:ContactInfo</code> 53, 58	<code>nativepdf</code> 64
<code>\hyxmp@xmp@to@pdf@date</code> 349, 352	<code>lptc4xmpCore:CreatorContactInfo</code> . . . 2, 3, 53, 54, 73	<code>pdfa</code> 8, 75
<code>\hyxmp@xmp@to@pdf@date@i</code> 353, 355	ISBN 2, 6, 20	<code>pdfaconformance</code> 4, 8, 52
<code>\hyxmp@xmp@to@pdf@date@ii</code> 358, 361	ISO 5, 6, 14, 18, 40	<code>pdfapart</code> 4, 8, 52
<code>\hyxmp@xmp@to@pdf@date@iii</code> 364, 367	ISSN 2, 6, 19	<code>pdfauthor</code> 4, 10, 13, 14, 17, 22, 23, 26, 49, 74
<code>\hyxmp@xmp@to@pdf@date@iv</code> 370, 373		<code>pdfauthortitle</code> . 4, 5, 13
<code>\hyxmp@xmp@to@pdf@date@v</code> 376, 379	J	<code>pdfbookedition</code> . . 4, 6
<code>\hyxmp@xmp@to@pdf@date@vi</code> 382, 386	<code>\jobname</code> . . . 62, 223, 265, 757, 765, 1434	<code>pdfbytes</code> 4, 7
<code>\hyxmp@xmp@to@pdf@date@vii</code> 389, 392, 402	K	<code>pdfcaptionwriter</code> . 4, 5
<code>\hyxmp@xmp@to@pdf@date@viii</code> 405, 408	<code>keeppdfinfo</code> (option) 12, 21	<code>pdfcontactaddress</code> 4, 5, 12
<code>\hyxmp@xmpRights@schema</code> 950 , 1397	Keywords 9, 10, 45	<code>pdfcontactcity</code> . . 4, 5
<code>\hyxmp@zero</code> 627, 634, 641, 647, 652	<code>\KV@Hyp@pdfauthor</code> . . 137	<code>pdfcontactcountry</code> 4, 5
	<code>\KV@Hyp@pdfkeywords</code> 158	<code>pdfcontactemail</code> . 4, 5
I	<code>kvoptions</code> 17, 24	<code>pdfcontactphone</code> . 4, 5
IETF 5	L	<code>pdfcontactpostcode</code> 4, 5
<code>\if@tempswa</code> . . . 866, 901	LF 52	<code>pdfcontactregion</code> . 4, 5
<code>\iffalse</code> 861, 896	Lua \LaTeX 9, 12, 33, 45, 51, 74	<code>pdfcontacturl</code> . 4, 5, 13
<code>\ifHy@pdfa</code> 936, 937, 943, 1023, 1343	Lua \TeX 34, 37, 63, 64, 67, 73, 75	<code>pdfcopyright</code> 4, 5, 49, 50, 73
<code>\ifhyxmp@unicodetex</code> 457 , 469, 774, 1357	M	<code>pdfcreationdate</code> 4, 7, 75
<code>ifluatex</code> 64	memoir 74	<code>pdfdate</code> 4, 7, 14, 18, 43, 74
<code>\ifluatex</code> 1447, 1451	Metadata 9, 63, 66	<code>pdfdocumentid</code> 4, 6, 75
<code>ifmtarg</code> 17, 75	<code>\month</code> 413, 414, 416	<code>pdfdoi</code> 4, 6
<code>ifxetex</code> 17	N	<code>pdffeissn</code> 4, 6
<code>\ifxetex</code> 474	NAK 17, 28, 34	<code>pdfinstanceid</code> . 4, 6, 75
	<code>nativepdf</code> (option) . . . 64	<code>pdfisbn</code> 4, 6
	<code>\newif</code> 457	<code>pdfissn</code> 4, 6
	<code>\next</code> 28 , 39 , 276 , 685	<code>pdfkeywords</code> . . . 4, 10, 13, 17, 22, 49
	<code>ngerman</code> 16, 72	<code>pdflang</code> 4–6, 14, 17, 25, 39, 49, 75
	<code>\number</code> 621, 623, 625, 630, 632, 637, 639	<code>pdflicenseurl</code> 4, 5, 13, 50, 73
	<code>\numexpr</code> 1463	<code>pdfmark</code> 64
	O	<code>pdfmetadate</code> 4, 7, 18, 75
	options	<code>pdfmetalang</code> 4, 5, 14, 39, 72, 75
	<code>baseurl</code>	<code>pdfmoddate</code> . . 4, 7, 75
	4, 6, 13, 17, 20, 51	

pdfnumpages	4, 7, 15, 16	pdfbookedition (option)	4, 6	pdfL ^A T _E X	4, 9, 12, 33, 45, 51
pdfpagerange	5, 6, 15, 16	pdfbytes (option)	4, 7	pdflicenseurl (option)	4, 5, 13, 50, 73
pdfproducer	4, 17	pdfcaptionwriter (option)	4, 5	pdfmark (option)	64
pdfpublication	5, 6	\pdfcatalog	1455	\pdfmark	1466, 1469, 1473, 1483, 1487, 1491
pdfpublisher	5, 6	\pdfcompresslevel	1449	pdfmetadate (option)	4, 7, 18, 75
pdfpubtype	5, 6	pdfcontactaddress (option)	4, 5, 12	pdfmetalang (option)	4, 5, 14, 39, 72, 75
pdfsource	5, 7, 75	pdfcontactcity (option)	4, 5	\pdfminorversion	814
pdfsubject	4, 10, 17, 49	pdfcontactcountry (option)	4, 5	pdfmoddate (option)	4, 7, 75
pdfsubtitle	5	pdfcontactemail (option)	4, 5	pdfnumpages (option)	4, 7, 15, 16
pdftitle	4, 5, 10, 14, 17, 26, 49, 74	pdfcontactphone (option)	4, 5	\pdfobj	1451
pdftype	5, 7, 75	pdfcontactpostcode (option)	4, 5	pdfpagerange (option)	5, 6, 15, 16
pdfurl	5, 6	pdfcontactregion (option)	4, 5	pdfproducer (option)	4, 17
pdfversionid	5, 6, 50	pdfcontacturl (option)	4, 5, 13	pdfpublication (option)	5, 6
pdfvolumenum	5, 6	pdfcopyright (option)	4, 5, 49, 50, 73	pdfpublisher (option)	5, 6
ps2pdf	64	pdfcreationdate (option)	4, 7, 75	pdfpubtype (option)	5, 6
textures	64	\pdfcreationdate	442, 981	pdfsource (option)	5, 7, 75
unicode	13, 75	pdfdate (option)	4, 7, 14, 18, 43, 74	\pdfstringdef	21
vtexpdfmark	64	PDFDocEncoding	22, 34, 35	pdfsubject (option)	4, 10, 17, 49
P					
\PackageWarning	302	pdfdocumentid (option)	4, 6, 75	pdfsubtitle (option)	5
\PackageWarningNoLine	222, 264, 1432	pdfdoi (option)	4, 6	pdfT _E X	7, 16, 36, 63, 64, 67, 75
PDF 1–4, 7, 9–12, 15, 17, 18, 21, 26, 27, 29, 30, 35, 36, 43–45, 51, 62–64, 66, 74, 75		pdfdoi (option)	4, 6	pdftitle (option)	4, 5, 10, 14, 17, 26, 49, 74
PDF/A	3, 8, 10–12, 19, 21, 44, 45, 52, 55, 57–59, 61, 73, 74	pdfescape	17	pdftype (option)	5, 7, 75
pdf:Keywords	2, 10, 45	\pdfextension	1459, 1463	pdfurl (option)	5, 6
pdf:PDFVersion	3, 45	\pdffeedback	445, 984, 1463	\pdfvariable	817
pdf:Producer	3, 45	pdfinstanceid (option)	4, 6, 75	pdfversionid (option)	5, 6, 50
\PDF@FinishDoc	123	pdfisbn (option)	4, 6	pdfvolumenum (option)	5, 6
pdfa (option)	8, 75	pdfissn (option)	4, 6	photoshop:AuthorsPosition	3, 52
pdfa:conformance (option)	4, 8, 52	pdfissuenum (option)	4, 6	photoshop:CaptionWriter	3, 52
pdfaid:conformance	3	pdfkeywords (option)	4, 10, 13, 17, 22, 49	PI	43, 44
pdfaid:part	3	pdflang (option)	4–6, 14, 17, 25, 39, 49, 75	PRISM	6, 54, 55, 59, 61
pdfapart (option)	4, 8, 52	\pdflastobj	1455	prism:aggregationType	3
pdfaType:prefix	73			prism:bookEdition	2
pdfauthor (option)	4, 10, 13, 14, 17, 22, 23, 26, 49, 74			prism:byteCount	2
pdfauthor:conformance	3				
pdfauthor:part	3				
pdfauthor:title (option)	4, 5, 13				

prism:doi	2	prism:number	2	Subject	9, 10, 21
prism:elssn	2	prism:pageCount	3	T	
prism:isbn	2	prism:pageRange	3	TeX	14,
prism:issn	2	prism:publicationName	3		15, 32, 34, 36, 37,
prism:number	2	prism:subtitle	3		40, 41, 43, 50, 66, 67
prism:pageCount	3	prism:url	3	texdate	14
prism:pageRange	3	prism:volume	3	Text	56, 57
prism:publicationName	3	xmp:BaseURL	2	\textunderscore	...
prism:subtitle	3	xmp:CreateDate	2, 51, 75		19, 20, 22
prism:url	3	xmp:CreatorTool	3	textures (option)	64
prism:volume	3	xmp:MetadataDate	2, 51, 75	\time	423, 431
\ProcessKeyValueOptions	170	xmp:ModifyDate	2, 51, 75	Title	9, 10, 21
Producer	45	xmpMM:DocumentID	3, 40, 50, 61	totpages	15, 16
properties, XMP		xmpMM:InstanceID	3, 40, 50, 61	U	
dc:creator	2, 10, 49, 74	xmpMM:VersionID	3, 50, 76	Unicode	13, 17, 34–
dc:date	2, 49	xmpRights:Marked	2, 50, 73		38, 48, 53, 61, 67, 73
dc:description	3, 10, 39, 49, 74	xmpRights:WebStatement	3, 50, 73	unicode (option)	13, 75
dc:format	2	ps2pdf (option)	64	URL	2, 3, 5, 6, 13,
dc:language	3, 49, 73, 74	Q			18, 20, 21, 50–52, 54
dc:publisher	3	\Q	447, 456	UTF-16BE	35
dc:rights	2, 15, 39, 49	R		UTF-32BE	35
dc:source	2, 49, 73	RDF	48	UTF-8	35
dc:subject	2, 49	rdf:Description	76	UUID	3, 5,
dc:title	3, 10, 39, 49, 74	rdf:li	2		6, 19, 40, 42, 43, 74
dc:type	3	rdf:Seq	2	V	
lptc4xmpCore:ContactInfo	53, 58	\renewcommand	171	\vfuzz	455
lptc4xmpCore:CreatorContactInfo	2, 3, 53, 54, 73	\RequirePackage	7, 10–15, 1444	vtexpdfmark (option)	64
pdf:Keywords	2, 10, 45	S		X	
pdf:PDFVersion	3, 45	\SE->pdfdoc@03	464	\x	574
pdf:Producer	3, 45	\SE->pdfdoc@15	465	xdvipdfmx	12, 66
pdfaid:conformance	3	\setkeys	671	X _q ^L A _T E _X	9,
pdfaid:part	3	\special	1497,		12, 33, 45, 51, 74
pdfaType:prefix	73		1505, 1534, 1540	X _q I _T E _X	17,
photoshop:AuthorsPosition	3, 52	stringenc	17		34, 37, 66, 67, 72, 73
photoshop:CaptionWriter	3, 52	\StringEncodingConvert	471,	XML	1, 2,
prism:aggregationType	3		477, 488, 491, 586		12, 27, 34–37, 44–
prism:bookEdition	2	X			49, 52, 54, 56, 62, 75
prism:byteCount	2	xmp:BaseURL	2	XMP	1, 2, 4, 5, 7,
prism:doi	2	xmp:CreateDate	2, 51, 75		9–18, 21, 25–30,
prism:elssn	2	U			33, 36, 39, 40, 43–
prism:isbn	2	V			46, 48, 50, 51, 55,
prism:issn	2	W			57, 61, 64–67, 72–76

xmp:CreatorTool 3	.. <u>1028</u> , 1044, <u>1067</u>	xmpRights:Marked	2, 50, 73
xmp:MetadataDate	2, 51, 75	xmpMM:DocumentID	.	xmpRights:WebStatement
xmp:ModifyDate	2, 51, 75 3, 40, 50, 61	3, 50, 73
\xmpcomma 102,	xmpMM:InstanceID	..	\xmptilde
	105, <u>137</u> , <u>158</u> , <u>287</u> 3, 40, 50, 61	<u>297</u>
xmpincl 4	xmpMM:VersionID	3, 50, 76	\XMPTruncateList
\XMPLangAlt <u>668</u>	\xmpquote 103,	... <u>301</u>
\xmplinesep	106, <u>137</u> , <u>158</u> , <u>296</u>		Y
			\year 412