

# The hyperxmp package\*

Scott Pakin  
scott+hyxmp@pakin.org

May 12, 2020

## Abstract

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

## 1 Introduction

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

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

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

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

---

\*This document corresponds to `hyperxmp` v5.2, dated 2020/05/12.

```

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

```

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

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

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

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

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

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

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

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

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

## 2 Usage

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

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

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

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

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

## 2.1 Option descriptions

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

```
pdftitle={Frankenstein},
pdfsubtitle={The Modern Prometheus},
```

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

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

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

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

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

pdflang	language is the same as the document language (hyperref's pdflang option). If neither pdfmetalang nor pdflang is specified, hyperxmp uses only "x-default" as the metadata language. Note that "x-default" metadata are always included in addition to the specified metadata language, as the user reading the document may not have specified a language preference.
pdfdocumentid pdfinstanceid	XMP can include a universally unique identifier (UUID) for each document and for each instance of a given document. By default, hyperxmp assigns a version 4 (i.e., pseudorandom) UUID [11] for each of these. However, a document can alternatively specify a particular document identifier using pdfdocumentid and (not normally recommended) a particular instance identifier using pdfinstanceid. These should be of the form <code>uuid:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx</code> , where "x" is a lowercase hexadecimal number. For example, <code>uuid:53ab7f19-a48c-5177-8bb2-403ad907f632</code> is a valid argument to pdfdocumentid (or pdfinstanceid). See Leach, Mealling, and Salz's UUID specification document for details on how to produce the various forms of UUIDs [11]. A more freeform mechanism than pdfinstanceid for versioning documents is available via pdfversionid. The version specified by pdfversionid can be incremented as 1, 2, 3, ...; identified with a hierarchical numbering scheme (e.g., this document is versioned 5.2 to match the package version); or labeled using any other approach. One possibility is to use a revision number or commit hash from the version-control software maintaining the document. For example, the <code>\gitVer</code> macro from the <code>gitver</code> package is an expandable (see Note 8 on page 15) version of the current Git hash that can suitably be passed to pdfversionid. If not specified, pdfversionid defaults to 1.
pdfisbn pdfissn pdfeissn pdfdoi	Already-published documents can be identified in a number of ways. pdfisbn specifies the ISBN. pdfissn refers to the ISSN of the <i>print</i> version of the document. pdfeissn refers to the ISSN of the <i>electronic</i> version of the document. pdfdoi specifies the DOI and should include only the DOI name without any URL prefix. For example, specify <code>pdfdoi={10.1145/3149526.3149532}</code> , <i>not</i> <code>pdfdoi={https://doi.org/10.1145/3149526.3149532}</code> .
pdfurl baseurl	pdfurl points to the complete URL for the document. In contrast, baseurl points one level up and is used to resolve relative URLs.
pdfidentifier	pdfidentifier provides an alternative mechanism to uniquely identify a document. Its advantage relative to pdfisbn, pdfissn, pdfdoi, etc. is its flexibility; any of a wide variety of identification types can be used. <sup>1</sup> pdfidentifier's disadvantage is that it allows only a single identifier per document. For example, a document could use <code>pdfidentifier=urn:iso:std:32000:ed-1:v1:en</code> to identify itself as version 1 of English-language ISO standard 32000-1, but then this same document could not also use pdfidentifier to identify itself by DOI ( <code>info:doi/...</code> ), ISBN ( <code>urn:ISSN:...</code> ), etc. (It can still use the options described in the previous paragraph, though.) If pdfidentifier is not specified explicitly, hyperxmp will use the first non-empty value out of the DOI, electronic ISSN, print ISSN, and ISBN or skip the identifier entirely if all of those are empty.

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

pdfpublication	<p>Already-published documents can further be identified by the publication in which they appear. <code>pdfpublication</code> specifies the title of the journal, magazine, or other parent document. The title language is assumed to be the same as the document language (<code>pdflang</code>) but can be overridden by preceding the text with a bracketed ISO 639-1 two-letter language code and an optional ISO 3166-1 two-letter region code. For example, <code>pdfpublication={ [fr]Charlie Hedbo }</code> indicates a French-language title. Were the language or pronunciation differences significant, <code>fr-FR</code> would indicate specifically the French spoken in France, as opposed to that spoken in, say, Canada (<code>fr-CA</code>) or Belgium (<code>fr-BE</code>). The publisher itself can be named using <code>pdfpublisher</code>.</p>
pdfpublisher pdfpubtype	<p><code>pdfpubtype</code> indicates the type of publication in which the document was published. This should be one of the PRISM aggregation types [8] such as <code>book</code>, <code>journal</code>, <code>magazine</code>, <code>manual</code>, <code>report</code>, or <code>whitepaper</code>. For publications in journals, magazines, and similar periodicals, a document can specify the volume number with <code>pdfvolumenum</code> and the issue number within the volume with <code>pdfissuenum</code>.</p>
pdfvolumenum pdfissuenum pdfpagerange	<p><code>pdfpagerange</code> indicates the page numbers at which the document appears within the publication. The intention is that this be a comma-separated list of dash-separated ranges, as in <code>pdfpagerange={1,4-5}</code>. See Note 9 on page 16 for advice on how to assign <code>pdfpagerange</code> semi-automatically. For books, <code>pdfbookedition</code> names the edition of the book. This is specified as text, not a number. As with <code>pdfpublication</code> (above), <code>pdfbookedition</code> accepts a bracketed language code, as in <code>pdfbookedition={ [en]Second edition }</code>.</p>
pdfbookedition	
pdfnumpages	<p>The number of pages in the published, print version of the document can be expressed with <code>pdfnumpages</code>. Note 9 on page 16 explains how to automatically assign a value to <code>pdfnumpages</code>.</p>
pdfdate	<p>XMP metadata can include a number of dates (in fact, timestamps, as they include both date and time components). <code>pdfdate</code> specifies the document date. It is analogous to the <math>\text{\LaTeX}</math> <code>\date</code> command, and, like <code>\date</code>, defaults to the date the document was built. It must be specified in either XMP format [4] or PDF format [3]. XMP dates are written in the form <code>YYYY-MM-DDThh:mm:ss+TT:tt</code>.<sup>2</sup> A W3C recommendation [14] discusses this format in more detail, but as an example, 14 hours, 15 minutes, 9 seconds past midnight U.S. Mountain Daylight Time (UTC-6) on the 23rd day of September in the year 2014 should be written as <code>2014-09-23T14:15:09-06:00</code>. This can be truncated (with loss of information) to <code>2014-09-23T14:15:09</code>, <code>2014-09-23T14:15</code>, <code>2014-09-23</code>, <code>2014-09</code>, or <code>2014</code> but no other subsets. PDF dates are written in the form <code>D:YYYYMMDDhhmmss+TT'tt'</code>. The same date in the preceding example would be written as <code>D:20140923141509-06'00'</code> in PDF format.</p>
pdfcreationdate pdfmoddate pdfmetadate	<p>The document's creation date, modification date, and metadata date are normally set automatically, but <code>pdfcreationdate</code>, <code>pdfmoddate</code>, and <code>pdfmetadate</code> can be used to override the defaults. Like <code>pdfdate</code>, <code>pdfmetadate</code> can be specified in either XMP or PDF format. However, because <code>hyperref</code> defines <code>pdfcreationdate</code> and <code>pdfmoddate</code> and expects these to be written as PDF dates, <code>hyperxmp</code> concomitantly</p>

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

	accepts these two dates only in PDF format as well. Note that it's rare that a document would need to specify any of <code>pdfcreationdate</code> , <code>pdfmoddate</code> , or <code>pdfmetadate</code> .
<code>pdftype</code>	<code>pdftype</code> describes the type of document being produced. This refers to “the nature or genre of the resource” [4] such as <code>poem</code> , <code>novel</code> or <code>working paper</code> , as opposed to the file format (always <code>application/pdf</code> when generated by <code>hyperxmp</code> ). Although <code>pdftype</code> can be assigned an arbitrary piece of text, the XMP specification recommends selecting types from a “controlled vocabulary” such as the DCMI Type Vocabulary [5]. The DCMI Type Vocabulary currently consists of only <code>Collection</code> , <code>Dataset</code> , <code>Event</code> , <code>Image</code> , <code>InteractiveResource</code> , <code>MovingImage</code> , <code>PhysicalObject</code> , <code>Service</code> , <code>Software</code> , <code>Sound</code> , <code>StillImage</code> , and <code>Text</code> . <code>pdftype</code> defaults to <code>Text</code> , which refers to “books, letters, dissertations, poems, newspapers, articles, archives of mailing lists,” [5] and other forms of text—all things $\text{\LaTeX}$ is commonly used to typeset.
<code>pdfrendition</code>	Sometimes a base document is rendered in different forms. <code>pdfrendition</code> indicates the particular rendition the current document instance represents. The value should come from the following controlled vocabulary [4]: <code>default</code> , <code>draft</code> , <code>low-res</code> , <code>proof</code> , <code>screen</code> , and <code>thumbnail</code> . <code>hyperxmp</code> 's default value is <code>default</code> , which indicates the master document, unless the <code>draft</code> option is passed to <code>\documentclass</code> , in which case <code>hyperxmp</code> defaults to <code>draft</code> .
<code>pdfbytes</code>	The <code>pdfbytes</code> option expresses the document's file size in bytes. The intention is for this to be used to display an estimate of download time to a user or to serve as a quick check on whether a file was transmitted correctly between systems. This feature is easiest to use in conjunction with <code>pdfTeX</code> 's <code>\pdffilesize</code> primitive: “ <code>pdfbytes={\pdffilesize{\jobname.pdf}}</code> ”. Note that this requires a second run of <code>pdftex</code> because it queries the size of the PDF file from the <i>previous</i> run.
<code>pdftrapped</code>	<code>hyperxmp</code> honors <code>hyperref</code> 's <code>pdftrapped</code> option. A document can indicate whether it employs color trapping by specifying <code>pdftrapped=True</code> or <code>pdftrapped=False</code> . ( <code>pdftrapped=Unknown</code> is also allowed.) A current limitation of <code>hyperxmp</code> is that if a value other than <code>False</code> is provided, a document will additionally need to specify <code>keeppdfinfo</code> (page 13) to ensure that the PDF Info dictionary specifies the correct trapping value.
<code>pdfapart</code> <code>pdfaconformance</code>	<code>pdfapart</code> and <code>pdfaconformance</code> , are used in conjunction with <code>hyperref</code> 's <code>pdfa</code> option to claim a particular PDF/A standard by which the document abides. They default to <code>pdfapart=1</code> and <code>pdfaconformance=B</code> , indicating the PDF/A-1b standard. These can be changed (with caution) to assert that the document abides by a different standard (e.g., PDF/A-2u). A document that conforms to the PDF/UA standard can use <code>pdfuapart</code> to indicate the PDF/UA conformance level. For example, <code>pdfuapart=1</code> asserts that the document respects PDF/UA-1. <code>pdfxstandard</code> indicates the particular PDF/X standard by which the document abides. Unlike <code>pdfapart</code> and <code>pdfaconformance</code> , which accept a number and a letter, respectively, <code>pdfxstandard</code> expects a textual identification of a standard name. The following are the PDF/X standard names that are considered acceptable at the time of this writing.
<code>pdfuapart</code> <code>pdfxstandard</code>	



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

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

`pdfsource` A rarely needed option, `pdfsource`, overrides the name of the L<sup>A</sup>T<sub>E</sub>X source file. It defaults to `\jobname.tex` but can be replaced by any other string. If `pdfsource` is given an empty argument, no document source will be specified at all.

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

## 2.2 A complete example

The following is a sample L<sup>A</sup>T<sub>E</sub>X document that provides values for most of the metadata options that `hyperxmp` recognizes:

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

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

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

```

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

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

```

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

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

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

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

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

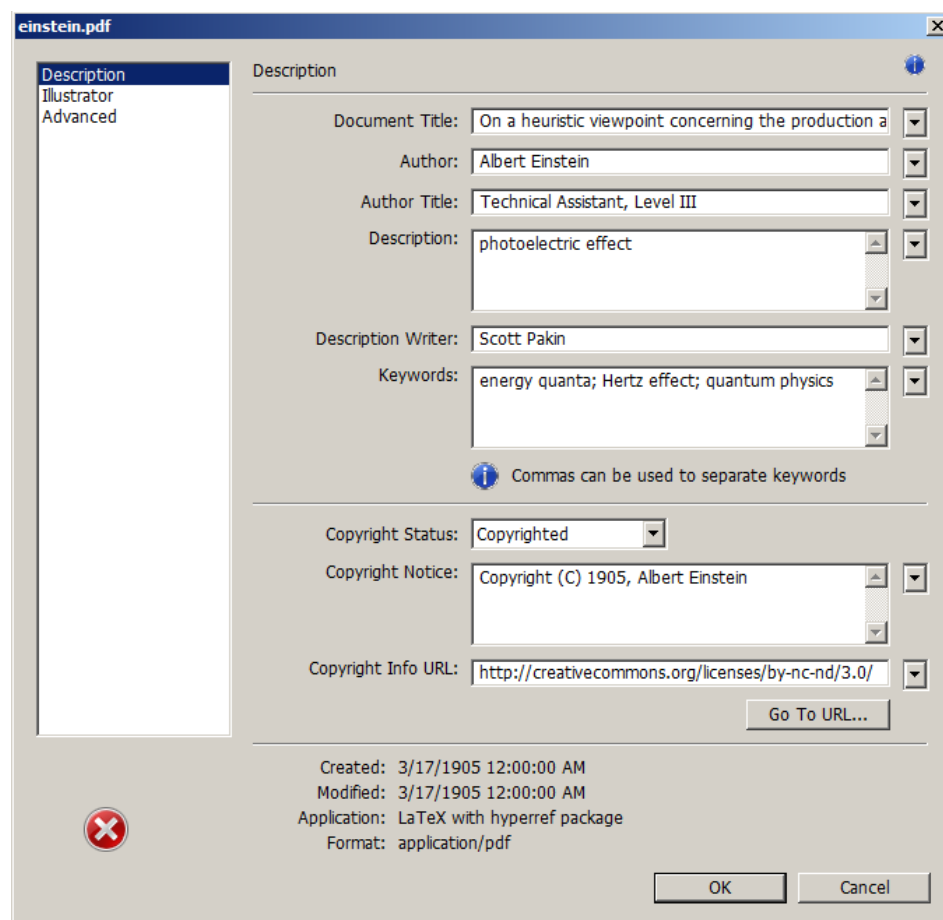


Figure 1: XMP metadata as it appears in Adobe Acrobat

## 2.3 Usage notes

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

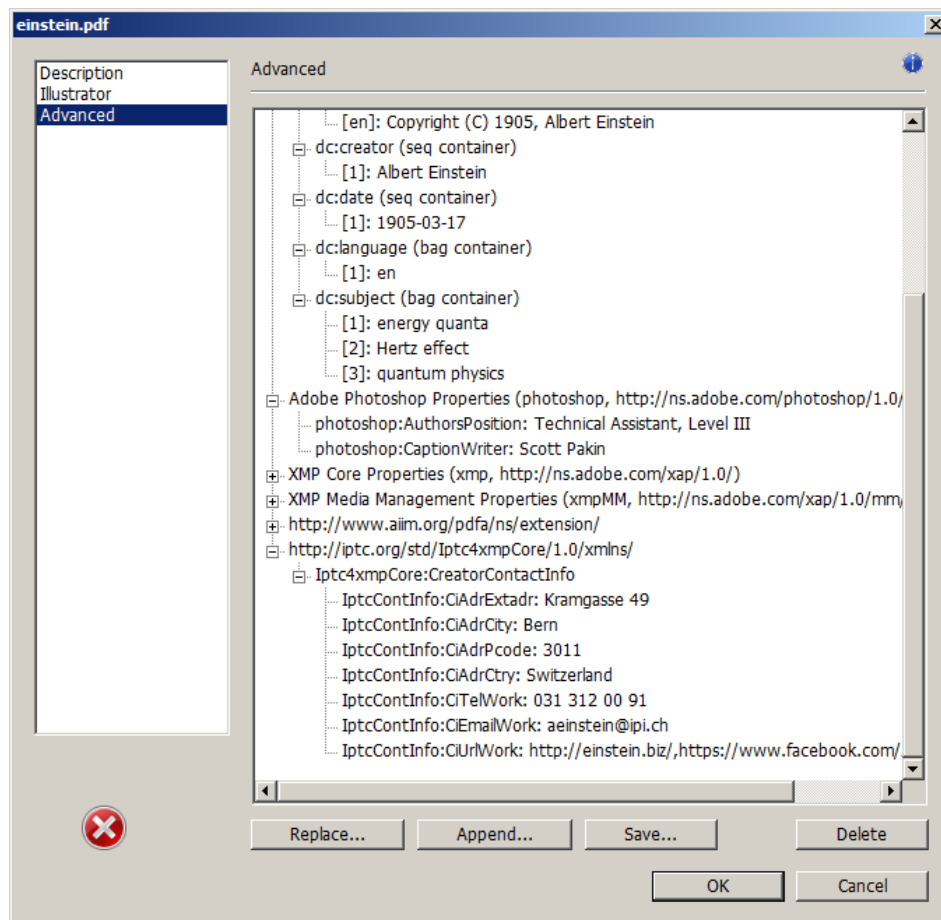


Figure 2: Additional XMP metadata as it appears in Adobe Acrobat

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**Note 6: Automatically specified metadata** `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 `TEX` terminology. This implies that any macros that are used in arguments are limited to a relatively small set of operations (such as conditionals and macro expansion) and must produce a string of text. Code (such as macro definitions and arithmetic operations) will be written to XMP as code, not as the result of executing the code.

By way of example, the macros provided by the `texdate` package for typesetting dates are not expandable (at least at the time of this writing). Hence, the

`\printdate{Y}` in the following code snippet is not replaced by the current year, as one might expect:

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

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

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

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

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

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

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

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



```

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

```

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

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

```

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

```

### 3 Implementation

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

#### 3.1 Initial preparation

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

```

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

```

`\hyxmp@at@end` The `\hyxmp@at@end` macro includes code at the end of the document. For pdfT<sub>E</sub>X, `\hyxmp@driver` 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<sup>A</sup>T<sub>E</sub>X 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 XMP metadata from `hyperref`'s `baseurl`, `pdfauthor`, `pdfkeywords`, `pdflang`, `pdfproducer`, `pdfsubject`, `pdftrapped`, and `pdftitle` options. It also introduces a number of new options, which are listed on pages 4–5. For consistency with `hyperref`'s document-metadata naming conventions (which are in turn based on L<sup>A</sup>T<sub>E</sub>X's document-metadata naming conventions), we do not prefix metadata-related macro names with our package-specific `\hyxmp@` prefix. That is, we use names like `\pdfcopyright` instead of `\hyxmp@pdfcopyright`.

We load a bunch of helper packages: `kvoptions` for package-option processing, `pdfescape` and `stringenc` for re-encoding Unicode strings, `intcalc` for performing integer calculations (division and modulo), `iftex` for determining which T<sub>E</sub>X engine is being used, `ifmtarg` for testing if a macro argument is empty or all spaces, `etoolbox` for dynamically patching existing commands (specifically, `hyperref`'s `\PDF@FinishDoc`), and `ifthen` for convenient string comparisons.

```

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

```

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

```

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

```

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

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

```

20 \newcommand{\hyxmp@pdfstringdef}[2]{%
21   \let\hyxmp@textunderscore=\textunderscore

```

```

22 \let\textunderscore=\hyxmp@uscore
23 \pdfstringdef{#1}{#2}%
24 \let\textunderscore=\hyxmp@textunderscore
25 }

\@pdfdatetime Prepare to store the document's date and (optionally) time. Whether specified
by the author in XMP format or PDF format (see Section 3.3.2) we always store
\@pdfdatetime as an XMP-format string.
26 \def\@pdfdatetime{}
27 \define@key{Hyp}{pdfdate}{%
28 \begingroup
29 \Hy@unicodedefalse

\next Expand pdfdate's argument and convert it to XMP format.
30 \edef\next{%
31 \noexpand\hyxmp@pdfstringdef\noexpand\@pdfdatetime{%
32 \noexpand\hyxmp@as@xmp@date{#1}}%
33 }%
34 \next
35 \endgroup
36 }

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

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

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

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

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

```

<code>\@pdfauthortitle</code>	<p>Prepare to store the author's position/title (e.g., Staff Writer).</p> <pre> 54 \def\@pdfauthortitle{} 55 \define@key{Hyp}{pdfauthortitle}{\hyxmp@pdfstringdef\@pdfauthortitle{#1}} </pre>
<code>\@pdfcaptionwriter</code>	<p>Prepare to store the name of the person who inserted the hyperxmp metadata.</p> <pre> 56 \def\@pdfcaptionwriter{} 57 \define@key{Hyp}{pdfcaptionwriter}{\hyxmp@pdfstringdef\@pdfcaptionwriter{#1}} </pre>
<code>\@pdfmetalang</code>	<p>Prepare to store the natural language of the document's metadata, typically as an ISO 639-1 two-letter abbreviation.</p> <pre> 58 \def\@pdfmetalang{} 59 \define@key{Hyp}{pdfmetalang}{\hyxmp@pdfstringdef\@pdfmetalang{#1}} </pre>
<code>\hyxmp@no@bad@parts</code>	<p>Complain about a bad pdfapart or pdfuapart if given trailing non-digits after a part number.</p> <pre> 60 \def\hyxmp@no@bad@parts#1\relax{% 61   \ifnotmtarg{#1}{% 62     \PackageWarning{hyperxmp}{pdfapart and pdfuapart must be numeric}% 63   }% 64 } </pre>
<code>\@pdfapart</code>	<p>Prepare to store the PDF/A part ID, which defaults to “1” if pdfa is passed to hyperref.</p> <pre> 65 \def\@pdfapart{} 66 \define@key{Hyp}{pdfapart}{% 67   \afterassignment\hyxmp@no@bad@parts\@tempcnta=0#1\relax 68   \hyxmp@pdfstringdef\@pdfapart{the\@tempcnta}% 69 } </pre>
<code>\@pdfaconformance</code>	<p>Prepare to store the PDF/A conformance ID, which defaults to “b” if pdfa is passed to hyperref and \@pdfapart is empty.</p> <pre> 70 \def\@pdfaconformance{} 71 \define@key{Hyp}{pdfaconformance}{% 72   \uppercase{\hyxmp@pdfstringdef\@pdfaconformance{#1}}% 73 } </pre>
<code>\@pdfuapart</code>	<p>Prepare to store the PDF/UA part ID.</p> <pre> 74 \def\@pdfuapart{} 75 \define@key{Hyp}{pdfuapart}{% 76   \afterassignment\hyxmp@no@bad@parts\@tempcnta=0#1\relax 77   \hyxmp@pdfstringdef\@pdfuapart{the\@tempcnta}% 78 } </pre>
<code>\hyxmp@set@pdfx@major</code>	<p>Parse pdfxstandard as “PDF/X-<i>&lt;major&gt;</i>”, setting \hyxmp@pdfx@major to <i>&lt;major&gt;</i>.</p> <pre> 79 \newcommand*{\hyxmp@set@pdfx@major}[1]{\hyxmp@set@pdfx@major@i#1!} </pre>

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

```

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

```

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

```

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

```

`\hyxmp@check@std` Compare a user-provided string to a fixed string. (Assumption: Both are names of PDF/X standard versions.) If they match, undefine `\next`, which we assume was previously defined to issue an “unrecognized standard” warning message.

```

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

```

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

```

92 \def\@pdfxstandard{}
93 \def\hyxmp@pdfx@major{}
94 \define@key{Hyp}{pdfxstandard}{%
95   \hyxmp@pdfstringdef\@pdfxstandard{#1}%

```

`\next` Issue a warning message if the PDF/X standard named by the user does not appear in a list of known PDF/X standards. This is to caution the user that `hyperxmp` generates standard-specific XMP metadata and it can only guess at the correct format for new standard versions. (See the comments on page 60 above the definition of `\hyxmp@pdfx@id@schema`, for example.)

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

115 \def\hyxmp@InstanceID{}
116 \define@key{Hyp}{pdfinstanceid}{\hyxmp@pdfstringdef\hyxmp@InstanceID{#1}}

```

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

```

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

```

`\ifdraft` Use the `ifdraft` package to determine if this is a draft or final document.

```

119 \begingroup
120 \let\ifdraft=\relax
121 \RequirePackage{ifdraft}

```

`\@pdfrendition` Prepare to store a tag describing how this rendition of the document differs from the master. The default value is `default`, which indicates the master document, except in the case of `\documentclass[draft]`, for which `\@pdfrendition` defaults to `draft`.

```

122 \ifdraft{%
123   \gdef\@pdfrendition{draft}%
124 }{%
125   \gdef\@pdfrendition{default}%
126 }
127 \endgroup
128 \define@key{Hyp}{pdfrendition}{\hyxmp@pdfstringdef\@pdfrendition{#1}}

```

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

```

129 \def\@pdfpublication{}
130 \define@key{Hyp}{pdfpublication}{\hyxmp@pdfstringdef\@pdfpublication{#1}}

```

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

```

131 \def\@pdfpubtype{}
132 \define@key{Hyp}{pdfpubtype}{\hyxmp@pdfstringdef\@pdfpubtype{#1}}

```

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

```

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

```

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

```

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

```

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

```

137 \def\@pdfissn{}
138 \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.

```

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

```

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

```

141 \def\@pdfisbn{}
142 \define@key{Hyp}{pdfisbn}{\hyxmp@pdfstringdef\@pdfisbn{#1}}

```

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

```

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

```

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

```

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

```

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

```

147 \def\@pdfvolumenum{}
148 \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.

```

149 \def\@pdfissuenum{}
150 \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.

```

151 \def\@pdfpagerange{}
152 \define@key{Hyp}{pdfpagerange}{\hyxmp@pdfstringdef\@pdfpagerange{#1}}

```

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

```

153 \def\@pdfdoi{}
154 \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.

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

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

```
157 \def\@pdfidentifier{}
158 \define@key{Hyp}{pdfidentifier}{\hyxmp@pdfstringdef\@pdfidentifier{#1}}
```

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

```
159 \def\@pdfsubtitle{}
160 \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.

```
161 \def\@pdfcontactaddress{}
162 \define@key{Hyp}{pdfcontactaddress}{%
163   \let\xmpcomma=\hyxmp@comma
164   \def\xmpquote##1{##1}%
165   \hyxmp@pdfstringdef\@pdfcontactaddress{#1}%
166   \def\xmpcomma{,}%
167   \let\xmpquote=\relax
168 }
```

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

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

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

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

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

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



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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

`\hyxmp@no@info@lists` Suppress hyperref from writing Author and Keywords into the Info dictionary. This prevents conflicts between the PDF metadata and the XMP metadata that cause PDF/A validation to fail. The PDF metadata can be restored by passing the `keeppdfinfo` option to `\hypersetup`.

```

183 \def\hyxmp@no@info@lists{%

```

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

```

184 \def\hyxmp@suppress@pdf@info{%
185   \global\let\PDF@FinishDoc=\@empty
186   \PackageWarningNoLine{hyperxmp}{%
187     Suppressing the _entire_ PDF Info dictionary.\MessageBreak
188     Please notify the hyperxmp maintainer%
189   }%
190 }%
191 \let\next=\relax
192 \patchcmd
193   {\PDF@FinishDoc}%
194   {/Author(\@pdfauthor)}%
195   {}%
196   {}%
197   {\let\next=\hyxmp@suppress@pdf@info}%
198 \patchcmd
199   {\PDF@FinishDoc}%
200   {/Keywords(\@pdfkeywords)}%
201   {}%
202   {}%
203   {\let\next=\hyxmp@suppress@pdf@info}%
204 \next
205 }

```

```

206 \define@key{Hyp}{keeppdfinfo}[true]{%
207   \gdef\hyxmp@no@info@lists{}%
208 }

```

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 209 \def\hyxmp@pdfauthor{}
210 \def\hyxmp@pdfkeywords{}

\hyxmp@redefine@Hyp If not already redefined, redefine hyperref's pdfauthor and pdfkeywords options to
properly handle \xmpcomma and \xmpquote.
211 \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.
212 \ifundefined{KV@Hyp@pdfauthor}{}%
213   \ifundefined{hyxmp@Hyp@pdfauthor}{%
214     \expandafter\let\expandafter\hyxmp@Hyp@pdfauthor
215       \csname KV@Hyp@pdfauthor\endcsname
216   }{}%
217 }%

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

```

```

218 \define@key{Hyp}{pdfauthor}{%
219   \let\xmpcomma=\hyxmp@comma
220   \def\xmpquote####1{####1}%
221   \let\hyxmp@and=\and
222   \def\and{,}%
223   \hyxmp@Hyp@pdfauthor{##1}%
224   \global\let\hyxmp@pdfauthor=\@pdfauthor
225   \def\and{and\space}%
226   \def\xmpcomma{,%}
227   \def\xmpquote####1{"####1"%}
228   \hyxmp@Hyp@pdfauthor{##1}%
229   \def\xmpcomma{,%}
230   \let\xmpquote=\relax
231   \let\and=\hyxmp@and
232 }%

```

`\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`.

```

233 \ifundefined{KV@Hyp@pdfkeywords}{%
234   \ifundefined{hyxmp@Hyp@pdfkeywords}{%
235     \expandafter\let\expandafter\hyxmp@Hyp@pdfkeywords
236     \csname KV@Hyp@pdfkeywords\endcsname
237   }{%
238 }%

```

`\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).

```

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

```

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

```

251 \let\hyxmp@ProcessKeyvalOptions=\ProcessKeyvalOptions

```

```

252 \renewcommand*{\ProcessKeyvalOptions}{%
253   \hyxmp@redefine@Hyp
254   \hyxmp@ProcessKeyvalOptions
255 }

\hyxmp@hypersetup  Redefine hyperref's \hypersetup command to invoke \hyxmp@redefine@Hyp before
\hypersetup        performing its normal option processing.
256 \let\hyxmp@hypersetup=\hypersetup
257 \def\hypersetup{%
258   \hyxmp@redefine@Hyp
259   \hyxmp@hypersetup
260 }

\hyxmp@find@metadata  Issue a warning message if the author failed to specify any metadata at all. This
\hyxmp@concat@metadata excludes metadata that is included automatically such as the current timestamp.
                        Note that we don't consider \pdfmetlang as metadata as that value is meaningful
                        only when used in conjunction with other information. We also don't examine
                        \pdfapart or \pdfaconformance because those have nonempty default values.
261 \newcommand*{\hyxmp@find@metadata}{%
262   \edef\hyxmp@concat@metadata{%
263     \@baseurl
264     \@pdfauthor
265     \@pdfauthortitle
266     \@pdfbookedition
267     \@pdfbytes
268     \@pdfcaptionwriter
269     \@pdfcontactaddress
270     \@pdfcontactcity
271     \@pdfcontactcountry
272     \@pdfcontactemail
273     \@pdfcontactphone
274     \@pdfcontactpostcode
275     \@pdfcontactregion
276     \@pdfcontacturl
277     \@pdfcopyright
278     \@pdfcreationdate
279     \@pdfdatetime
280     \@pdfdoi
281     \@pdfeissn
282     \@pdfidentifier
283     \@pdfisbn
284     \@pdfissn
285     \@pdfissuenum
286     \@pdfkeywords
287     \@pdflang
288     \@pdflicenseurl
289     \@pdfmetadatetitle
290     \@pdfmoddate
291     \@pdfnumpages

```

```

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

```

`\hyxmp@check@standards` Most PDF standards require that certain metadata be present. If compliance with a PDF standard is claimed but any of the metadata it requires are absent, issue a warning message.

```

311 \newcommand*{\hyxmp@check@standards}{%

```

If the `pdfa` option was passed to `hyperref` but `\pdfapart` is not set, set it to 1 and `\pdfaconformance` to B.

```

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

```

`\hyxmp@standards` We define `\hyxmp@standards` to be non-empty if *any* PDF standard is claimed (currently, PDF/A, PDF/X, or PDF/UA).

```

324 \edef\hyxmp@standards{%
325 \pdfapart
326 \pdfxstandard
327 \pdfuapart
328 }%

```

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

```

329 \ifnotmtargexp{\hyxmp@standards}{%

```

```

330 \ifmtargexp{\@pdftitle}{%
331 \PackageWarningNoLine{hyperxmp}{%
332 Missing pdftitle (required for PDF standards\MessageBreak
333 compliance)%
334 }%
335 }%
336 {}%
337 }%
338 }

```

Rather than load `hyperref` ourself 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.

```

339 \AtBeginDocument{%
340 \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`.

```

341 \ifx\@pdflang\relax
342 \let\@pdflang=\@empty
343 \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.

```

344 \ifx\@pdfmetalang\@empty
345 \ifx\@pdflang\@empty
346 \let\@pdfmetalang=\hyxmp@x@default
347 \else
348 \edef\@pdfmetalang{\@pdflang}%
349 \fi
350 \fi
351 \hyxmp@xmlify\@pdfmetalang

```

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

```

352 \ifmtargexp{\@pdftitle}{%
353 \ifnotmtargexp{\@title}{%
354 \hypersetup{pdftitle={\@title}}}%
355 }%
356 }%
357 {}%
358 \ifmtargexp{\@pdfauthor}{%
359 \ifnotmtargexp{\@author}{%

```

```

360     \hypersetup{pdfauthor={\@author}}}%
361     }%
362   }%
363   {}%

```

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

```

364   \@ifundefined{XeTeXversion}{}{}%
365   \@ifmtargexp{\@pdfcreationdate}{}%
366     \let\@pdfcreationdate=\hyxmp@today@pdf
367   }%
368   {}%
369   }%

```

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

```

370   \hyxmp@check@standards

```

Older versions of `hyperref` write the Info dictionary to the PDF file at the end of the document. New versions of `hyperref` write the Info dictionary to the PDF file at the *beginning* of the document. For compatibility with both old and new `hyperref` implementations we suppress writing the Info dictionary here, at the beginning of the document.

```

371   \hyxmp@no@info@lists

```

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

```

372   \hyxmp@at@end{%
373     \hyxmp@find@metadata
374     \hyxmp@embed@packet
375   }%
376 }{%
377   \PackageWarningNoLine{hyperxmp}{}%
378   \jobname.tex failed to include a\MessageBreak
379   \string\usepackage\string{hyperref}\string{
380   in the preamble.\MessageBreak
381   Consequently, all hyperxmp functionality will be\MessageBreak
382   disabled}%
383 }%
384 }

```

### 3.3 Manipulating author-supplied data

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

#### 3.3.1 List manipulation

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

```
\hyxmp@commas@to@list  Given a macro name (#1) and a comma-separated list (#2), define the macro name
                        as the elements of the list, each preceded by \@elt. (Executing the macro therefore
                        applies \@elt to each element in turn.)
385 \newcommand*{\hyxmp@commas@to@list}[2]{%
386   \gdef#1{%
387     \expandafter\hyxmp@commas@to@list@i\expandafter#1#2,,%
388   }

\hyxmp@commas@to@list@i  Recursively construct macro #1 from comma-separated list #2. Stop if #2 is empty.
\next 389 \def\hyxmp@commas@to@list@i#1#2,{%
390   \gdef\hyxmp@sublist{#2}%
391   \ifx\hyxmp@sublist\@empty
392     \let\next=\relax
393   \else
394     \hyxmp@trimspaces\hyxmp@sublist
395     \@cons{#1}{\hyxmp@sublist}%
396     \def\next{\hyxmp@commas@to@list@i{#1}}%
397   \fi
398   \next
399 }

\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.
400 \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.
```



```

401 \bgroup
402 \catcode'\^^C=11
403 \gdef\hyxmp@comma{^^C}
404 \egroup

```

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

```

405 \bgroup
406 \catcode'\^^U=11
407 \gdef\hyxmp@uscore{^^U}
408 \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.

```

409 \let\xmpquote=\relax

```

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

```

410 \bgroup
411 \catcode'\~=12%
412 \gdef\xmptilde{~}%
413 \egroup

```

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

```

414 \newcommand{\XMPTruncateList}[1]{%
415   \PackageWarning{hyperxmp}{%
416     \noexpand\XMPTruncateList has been deprecated since\MessageBreak
417     hyperxmp 4.0 and may be removed in future\MessageBreak
418     versions of the package. \noexpand\XMPTruncateList\MessageBreak
419     was found}%
420   \edef\hyxmp@temp@str{\csname hyxmp@#1\endcsname}%
421   \hyxmp@commas@to@list{\hyxmp@temp@list}{\hyxmp@temp@str}%
422   \def\@elt##1{%
423     \expandafter\gdef\csname @#1\endcsname{##1}%
424     \let\@elt=\@gobble
425   }
426   \hyxmp@temp@list
427 }%

```

### 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:YYMMDDhhmmss+TT’tt’” (e.g., D:20200512222355-06’00’) [3], while XMP timestamps are of the form “YYYY-MM-DDThh:mm:ss+TT:tt” (e.g., 2020-05-12T22:23:55-06:00) [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.

<code>\hyxmp@first@char</code>	Return the first character of a string. This macro is fully expandable.
<code>\hyxmp@first@char@i</code>	<pre> 428 \def\hyxmp@first@char#1{\hyxmp@first@char@i#1\relax} 429 \def\hyxmp@first@char@i#1#2\relax{#1} </pre>
<code>\hyxmp@as@xmp@date</code>	<p>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.</p> <pre> 430 \def\hyxmp@as@xmp@date#1{% 431   \expandafter\ifnum\expandafter'\hyxmp@first@char@i#1\relax='D 432     \hyxmp@pdf@to@xmp@date{#1}% 433   \else 434     #1% 435   \fi 436 } </pre>
<code>\hyxmp@pdf@to@xmp@date</code>	<p>Convert a timestamp from PDF format to XMP format. This macro is fully expandable.</p> <pre> 437 \def\hyxmp@pdf@to@xmp@date#1:#2#3#4#5#6#7#8#9{% 438   #2#3#4#5-#6#7-#8#9% 439   \hyxmp@parse@time 440 } </pre>
<code>\hyxmp@parse@time</code>	<p>This is a helper function for <code>\hyxmp@pdf@to@xmp@date</code>. <code>\hyxmp@pdf@to@xmp@date</code> proper parses only the year, month, and day then calls <code>\hyxmp@parse@time</code>. <code>\hyxmp@parse@time</code> parses the hours, minutes, and seconds then calls <code>\hyxmp@parse@tz@char</code>.</p> <pre> 441 \def\hyxmp@parse@time#1#2#3#4#5#6{% 442   T#1#2:#3#4:#5#6% 443   \hyxmp@parse@tz@char 444 } </pre>
<code>\hyxmp@parse@tz@char</code>	<p>This is another helper function for <code>\hyxmp@pdf@to@xmp@date</code>. So far, the date and time have been parsed. <code>\hyxmp@parse@tz@char</code> parses the first character of the timezone descriptor. This can be one of “+” for eastern timezones (UTC+<math>x</math>, including Asia, Oceania, and most of Europe), “-” for western timezones (UTC-<math>x</math>, 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 <code>\hyxmp@parse@tz</code>; timezones beginning with “Z” are not.</p>

```

445 \def\hyxmp@parse@tz@char#1{%
446   #1%
447   \ifx#1-%
448     \expandafter\hyxmp@parse@tz
449   \else
450     \ifx#1+%
451       \expandafter\hyxmp@parse@tz
452     \fi
453   \fi
454 }

```

`\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.

```

455 \def\hyxmp@parse@tz#1'#2'{%
456   #1:#2%
457 }

```

`\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.

```

458 \def\hyxmp@as@pdf@date#1{%
459   \expandafter\ifx\hyxmp@first@char@i#1\relax D%
460   #1%
461   \else
462     \hyxmp@xmp@to@pdf@date{#1}%
463   \fi
464 }

```

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

```

465 \def\hyxmp@xmp@to@pdf@date#1{%
466   D:\hyxmp@xmp@to@pdf@date@i#1\relax\relax
467 }

```

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

```

468 \def\hyxmp@xmp@to@pdf@date@i#1#2#3#4#5#6{%
469   #1#2#3#4%
470   \ifx#5-%
471     \expandafter\hyxmp@xmp@to@pdf@date@ii\expandafter#6%
472   \fi
473 }

```

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

```

474 \def\hyxmp@xmp@to@pdf@date@ii#1#2#3#4{%
475   #1#2%
476   \ifx#3-%
477     \expandafter\hyxmp@xmp@to@pdf@date@iii\expandafter#4%
478   \fi
479 }

```

```

\hyxmp@xmp@to@pdf@date@iii Parse the day for \hyxmp@xmp@to@pdf@date.
480 \def\hyxmp@xmp@to@pdf@date@iii#1#2#3#4{%
481   #1#2%
482   \ifx#3T%
483     \expandafter\hyxmp@xmp@to@pdf@date@iv\expandafter#4%
484   \fi
485 }

\hyxmp@xmp@to@pdf@date@iv Parse the hour for \hyxmp@xmp@to@pdf@date.
486 \def\hyxmp@xmp@to@pdf@date@iv#1#2#3#4{%
487   #1#2%
488   \ifx#3:%
489     \expandafter\hyxmp@xmp@to@pdf@date@v\expandafter#4%
490   \fi
491 }

\hyxmp@xmp@to@pdf@date@v Parse the minute for \hyxmp@xmp@to@pdf@date.
492 \def\hyxmp@xmp@to@pdf@date@v#1#2#3#4{%
493   #1#2%
494   \ifx#3:%
495     \expandafter\hyxmp@xmp@to@pdf@date@vi\expandafter#4%
496   \fi
497 }

\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.
498 \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).
499 \def\hyxmp@xmp@to@pdf@date@vi#1#2#3#4{%
500   #1#2%
501   \ifx#3+%
502     +\expandafter\hyxmp@xmp@to@pdf@date@vii
503   \fi
504   \ifx#3-%
505     -\expandafter\hyxmp@xmp@to@pdf@date@vii
506   \fi
507   \ifx#3Z%
508     Z%
509   \fi
510   \ifx#3\relax
511     \expandafter\hyxmp@gobbletwo
512   \fi
513   \@gobbletwo #4%
514 }

```

```

\hyxmp@xmp@to@pdf@date@vii Parse the time-zone hours for \hyxmp@xmp@to@pdf@date.
515 \def\hyxmp@xmp@to@pdf@date@vii#1\@gobbletwo#2#3#4#5{%
516   #2#3%
517   \ifx#4:%
518     \expandafter\hyxmp@xmp@to@pdf@date@viii\expandafter#5%
519   \fi
520 }

\hyxmp@xmp@to@pdf@date@viii Parse the time-zone minutes for \hyxmp@xmp@to@pdf@date.
521 \def\hyxmp@xmp@to@pdf@date@viii#1#2#3#4{%
522   '#1#2'%
523 }

\hyxmp@today@xmp@define Use TeX primitives to define a given macro as today's date in YYYY-MM-DDThh:mmZ format.
524 \def\hyxmp@today@xmp@define#1{%
    The date is a straightforward representation of TeX's \year, \month, and \day
    primitives, with the latter two zero-padded to two digits apiece.
525   \xdef#1{\the\year}%
526   \ifnum\month<10
527     \xdef#1{#1-0\the\month}%
528   \else
529     \xdef#1{#1-\the\month}%
530   \fi
531   \ifnum\day<10
532     \xdef#1{#1-0\the\day}%
533   \else
534     \xdef#1{#1-\the\day}%
535   \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.
536   \@tempcnta=\time
537   \divide\@tempcnta by 60
538   \ifnum\@tempcnta<10
539     \xdef#1{#1T0\the\@tempcnta}%
540   \else
541     \xdef#1{#1T\the\@tempcnta}%
542   \fi
543   \multiply\@tempcnta by -60
544   \advance\@tempcnta by \time
545   \ifnum\@tempcnta<10
546     \xdef#1{#1:0\the\@tempcnta}%
547   \else
548     \xdef#1{#1:\the\@tempcnta}%
549   \fi
550   \xdef#1{#1Z}%

```

```

551 }

\hyxmp@try@today  If \hyxmp@today@xmp is still empty and #1 is defined, evaluate #2. Otherwise, do
                    nothing.
552 \def\hyxmp@try@today#1#2{%
553   \ifmtargexp{\hyxmp@today@xmp}{%
554     \ifundefined{#1}{#2}%
555   }%
556   {%
557 }

\hyxmp@today@xmp  Define \hyxmp@today@xmp as the current date and (if available) time and timezone
                    in XMP Date format [4].
558 \def\hyxmp@today@xmp{%
    Case 1: \pdfcreationdate is defined (pdfLATEX and pre-0.85 LuaLATEX).
559 \hyxmp@try@today{\pdfcreationdate}{%
560   \edef\hyxmp@today@xmp{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
561 }
    Case 2: \pdffeedback is defined (LuaLATEX 0.85+).
562 \hyxmp@try@today{\pdffeedback}{%
563   \edef\hyxmp@today@xmp{\expandafter\hyxmp@pdf@to@xmp@date\pdffeedback creationdate}%
564 }

\hyxmp@timestamp  Case 3: \filemoddate is defined (XYLATEX). In this case, we treat the timestamp
                    of the job's .log file as the current date/time.
565 \hyxmp@try@today{\filemoddate}{%
566   \edef\hyxmp@today@xmp{\filemoddate{\jobname.log}}%
567   \edef\next{%
568     \edef\noexpand\hyxmp@today@xmp{\noexpand\hyxmp@as@xmp@date{\hyxmp@today@xmp}}%
569   }%
570   \next
571 }%
    Case 4: None of the above. Do the best we can using the available TEX primitives
            (\year, \month, \day, and \time.
572 \hyxmp@try@today{\year}{%
573   \hyxmp@today@xmp\define\hyxmp@today@xmp
574 }

\hyxmp@today@pdf  Define \hyxmp@today@pdf as the current date and (if available) time and timezone
                    in PDF date format [3]. To do so we simply convert \hyxmp@today@xmp, defined
                    above, from XMP to PDF using \hyxmp@xmp@to@pdf@date.
575 \expandafter\edef\expandafter\hyxmp@today@pdf\expandafter{%
576   \expandafter\hyxmp@xmp@to@pdf@date\expandafter{\hyxmp@today@xmp}%
577 }

```

### 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.

```
578 \catcode'\Q=3
\hyxmp@trimspaces\x redefines \x to have the same replacement text sans leading
and trailing space tokens.
579 \newcommand{\hyxmp@trimspaces}[1]{%
  Use grouping to emulate a multi-token afterassignment queue.
580 \begingroup
  Put "\toks 0 {" into the afterassignment queue.
581 \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.
582 \expandafter\hyxmp@trimb\expandafter\noexpand#1Q Q}%
  Transfer the trimmed text back into #1.
583 \edef#1{\the\toks0}%
584 }
```

`\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.

```
585 \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.

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

### 3.3.4 Converting text to XML

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

`\ifhyxmp@unicodetex` X<sub>Y</sub>TeX and LuaTeX natively support Unicode. We define the conditional `\ifhyxmp@unicodetex` to check for these so we can properly handle encoding

`\hyxmp@unicodetextrue`

`\ifhyxmp@unicodetex`

`\hyxmp@unicodetexfalse`

conversions. The trick here is that Unicode T<sub>E</sub>X implementations compare decimal 64 to hexadecimal 40 (decimal 64), specified with four carets, and take the TRUE branch; non-Unicode T<sub>E</sub>X implementations compare decimal 64 to character “^” (decimal 94), ignore the “^^0040” and the rest of the TRUE branch, and take the FALSE branch.

```
588 \newif\ifhymp@unicodetex
589 \ifnum64='\^^^0040\relax
590 \hymp@unicodetextrue
591 \else
592 \hymp@unicodetexfalse
593 \fi
```

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

```
594 \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 \hymp@xmlify below) as a placeholder for an underscore character.

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

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

```
596 \newcommand*{\hymp@xmlify}[1]{%
597 \gdef\hymp@xmlified{}
```

Escaped PDF string → PDFDocEncoding/Unicode

```
598 \EdefUnescapeString\hymp@text{#1}%
599 \ifhymp@unicodetex
```

PDFDocEncoding/Unicode → UTF-32BE

```
600 \hymp@is@unicode\hymp@text{%
601 \StringEncodingConvert
602 \hymp@text\hymp@text{utf16be}{utf32be}%
603 }{%
604 \ifXeTeX
605 \hymp@xetex@crap
606 \else
607 \StringEncodingConvert
608 \hymp@text\hymp@text{pdfdoc}{utf32be}%
609 \fi
610 }%
```

UTF-32BE → UTF-32BE as hex string

```
611 \EdefEscapeHex\hymp@text{\hymp@text}%
```



UTF-32BE → XML in ASCII

```

612 \edef\hyxmp@text{%
613 \expandafter
614 }\expandafter\hyxmp@toxml@unicodetex\hyxmp@text
615 \relax\relax\relax\relax\relax\relax\relax
616 \else

```

PDFDocEncoding/Unicode → UTF-8

```

617 \hyxmp@is@unicode\hyxmp@text{%
618 \StringEncodingConvert
619 \hyxmp@text\hyxmp@text{utf16be}{utf8}%
620 }{%
621 \StringEncodingConvert
622 \hyxmp@text\hyxmp@text{pdfdoc}{utf8}%
623 }%

```

UTF-8 → UTF-8 as hex string

```

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

```

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

```

625 \edef\hyxmp@text{%
626 \expandafter\hyxmp@toxml\hyxmp@text\@empty\@empty
627 }%

```

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

```

628 \EdefUnescapeHex\hyxmp@text{\hyxmp@text}%
629 \fi
630 \global\let\hyxmp@xmlified\hyxmp@text
631 }

```

\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.

```

632 \begingroup
633 \lccode'\<=254 %
634 \lccode'\>=255 %
635 \catcode254=12 %
636 \catcode255=12 %
637 \lowercase{\endgroup
638 \def\hyxmp@is@unicode#1{%
639 \expandafter\hyxmp@@is@unicode#1<>\@nil
640 }%
641 \def\hyxmp@@is@unicode#1<>#2\@nil{%
642 \ifx\#1\%
643 \expandafter\@firstoftwo
644 \else
645 \expandafter\@secondoftwo
646 \fi
647 }%
648 }

```

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

```

649 \def\hyxmp@toxml#1#2{%
650   \ifx#1\@empty
651   \else
652     \ifnum"#1#2='\& %
653       26616D703B% &amp;
654     \else\ifnum"#1#2='\< %
655       266C743B% &lt;
656     \else\ifnum"#1#2='\> %
657       2667743B% &gt;
658     \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.

```

659   \@ifundefined{pdfmark}{%
660     #1#2%
661   }{%
662     \ifnum"#1#2='\( %
663       5C28% \(
664     \else\ifnum"#1#2='\) %
665       5C29% \)
666     \else
667       #1#2%
668     \fi\fi
669   }%
670   \fi\fi\fi
671   \expandafter\hyxmp@toxml
672 \fi
673 }

```

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

`\hyxmp@text`

```

674 \def\hyxmp@toxml@unicodetex#1#2#3#4#5#6#7#8{%
675   \ifx#1\relax
676   \else
677     \ifnum"#1#2#3#4#5#6#7#8>127 %
678     \uccode'\*"#1#2#3#4#5#6#7#8\relax
679     \uppercase{%
680       \edef\hyxmp@text{\hyxmp@text *}%

```

```

681     }%
682     \else\ifnum"#7#8='< %
683         \edef\hyxmp@text{\hyxmp@text &lt;}%
684     \else\ifnum"#7#8='& %
685         \edef\hyxmp@text{\hyxmp@text &amp;}%
686     \else\ifnum"#7#8='> %
687         \edef\hyxmp@text{\hyxmp@text &gt;}%
688     \else\ifnum"#7#8='\ %
689         \edef\hyxmp@text{\hyxmp@text\space}%
690     \else
691         \uccode'\*="#7#8\relax
692         \uppercase{%
693             \edef\hyxmp@text{\hyxmp@text *}%
694         }%
695     \fi\fi\fi\fi\fi
696     \expandafter\hyxmp@toxml@unicodetex
697 \fi
698 }

```

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

```

699 \def\hyxmp@skipzeros#1{%
700     \ifx#10%
701         \expandafter\hyxmp@skipzeros
702     \fi
703 }

```

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

```

\hyxmp@xetex@crap \hyxmp@try 704 \begingroup
\hyxmp@crap@result 705 \def\x#1{\endgroup
\hyxmp@text 706 \def\hyxmp@xetex@crap{%
707     \edef\hyxmp@try{%
708         \expandafter\hyxmp@SpaceOther\hyxmp@text#1@nil
709     }%
710     \let\hyxmp@crap@result=N%
711     \expandafter\hyxmp@crap@test\hyxmp@try\relax
712     \ifx\hyxmp@crap@result Y%
713         \let\hyxmp@text\@empty
714         \expandafter\hyxmp@crap@convert\hyxmp@try\relax
715     \else
716         \StringEncodingConvert\hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
717     \fi
718 }%
719 }
720 \x{ }

```

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

```

721 \begingroup
722 \catcode'\~=12 %

```

```

723 \lccode'\~=' \ %
724 \lowercase{\endgroup
725 \def\hyxmp@SpaceOther#1 #2\@nil{%
726   #1%
727   \ifx\relax#2\relax
728     \expandafter\@gobble
729   \else
730     ~%
731     \expandafter\@firstofone
732   \fi
733   {\hyxmp@SpaceOther#2\@nil}%
734 }%
735 }

```

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

```

736 \def\hyxmp@crap@test#1{%
737   \ifx#1\relax
738   \else
739     \ifnum'#1>127 %
740       \let\hyxmp@crap@result=Y%
741       \expandafter\expandafter\expandafter\hyxmp@skiptorelax
742     \else
743       \expandafter\expandafter\expandafter\hyxmp@crap@test
744     \fi
745   \fi
746 }

```

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

```

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

```

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

```

\hyxmp@num 748 \def\hyxmp@crap@convert#1{%
\hyxmp@text 749   \ifx#1\relax
750   \else
751     \edef\hyxmp@num{\number'#1}%
752     \ifnum\hyxmp@num>"FFFFFF %
753       \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"1000000}\relax
754       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
755       \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"1000000}}%
756     \else
757       \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
758     \fi
759     \ifnum\hyxmp@num>"FFFF %
760       \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"10000}\relax
761       \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
762       \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"10000}}%
763     \else
764       \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
765     \fi

```

```

766 \ifnum\hyxmp@num>"FF %
767 \lccode'\!=\intcalDiv{\hyxmp@num}{\number"100}\relax
768 \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
769 \edef\hyxmp@num{\intcalMod{\hyxmp@num}{\number"100}}%
770 \else
771 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
772 \fi
773 \ifnum\hyxmp@num>0 %
774 \lccode'\!=\hyxmp@num\relax
775 \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
776 \else
777 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
778 \fi
779 \expandafter\hyxmp@crap@convert
780 \fi
781 }

```

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

```

782 \begingroup
783 \catcode0=12 %
784 \gdef\hyxmp@zero{^^00}%
785 \endgroup

```

### 3.3.5 Outputting structured XML

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

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

```

786 \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.

```

787 \newcommand*{\hyxmp@add@simple}[2]{%
788 \ifnotmtargexp{#2}{%
789 \hyxmp@xmlify{#2}%
790 \hyxmp@add@to@xml{\hyxmp@extra@indent_<}}%
791 \xdef\hyxmp@xml{\hyxmp@xml#1}%
792 \hyxmp@add@to@xml{>\hyxmp@xmlified</}%
793 \xdef\hyxmp@xml{\hyxmp@xml#1>^^J}%
794 }%
795 }

```

`\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.

```

796 \newcommand*{\hyxmp@add@simple@var}[2]{%
797   \expandafter\ifx\csname#2\endcsname\relax
798   \else
799     \hyxmp@xmlify{\csname#2\endcsname}%
800     \hyxmp@add@to@xml{%
801       \hyxmp@extra@indent-----<#1>\hyxmp@xmlified</#1>^^J%
802     }%
803   \fi
804 }
```

`\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.

```

805 \newcommand*{\hyxmp@add@simple@lang}[2]{%
806   \ifnotmtarg{#2}{%
807     \hyxmp@xmlify{#2}%
808     \expandafter\hyxmp@add@simple@lang@i\hyxmp@xmlified\relax{#1}%
809   }%
810 }
```

`\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.

```

811 \newcommand*{\hyxmp@add@simple@lang@i}{%
812   \ifnextchar[\hyxmp@add@simple@lang@ii{\hyxmp@add@simple@lang@ii[]}%
813 }
```

`\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.

```

814 \def\hyxmp@add@simple@lang@ii[#1]#2\relax#3{%
815   \ifnotmtarg{#2}{%
816     \hyxmp@xmlify{#2}%
817     \ifmtarg{#1}{%
818       \hyxmp@add@to@xml{%
819         -----<#3>\hyxmp@xmlified</#3>^^J%
820       }%
821     }{%
822       \hyxmp@add@to@xml{%
823         -----<#3 xml:lang="#1">\hyxmp@xmlified</#3>^^J%
824       }%
825     }%
826   }
```

```

826 }%
827 }

```

`\hyxmp@add@simple@pfx` Given an XMP tag (#1), a—typically hard-wired—prefix string (#2), and a main string (#2), if the main string is nonempty, add a begin tag, both strings, and an end tag to the packet. The “simple” in the macro name indicates that the string is output without variations for different languages.

```

828 \newcommand*{\hyxmp@add@simple@pfx}[3]{%
829   \ifnotmtargexp{#3}{%
830     \hyxmp@add@to@xml{\hyxmp@extra@indent_<}}%
831     \xdef\hyxmp@xml{\hyxmp@xml#1}%
832     \hyxmp@pdfstringdef\hyxmp@iprefix{#2}%
833     \hyxmp@xmlify{\hyxmp@iprefix}%
834     \hyxmp@add@to@xml{>\hyxmp@xmlified}%
835     \hyxmp@xmlify{#3}%
836     \hyxmp@add@to@xml{\hyxmp@xmlified</}%
837     \xdef\hyxmp@xml{\hyxmp@xml#1>^^J}%
838   }%
839 }

```

### 3.3.6 Providing metadata in multiple languages

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

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

```

840 \def\hyxmp@alt@title{}
841 \def\hyxmp@alt@description{}
842 \def\hyxmp@alt@rights{}

```

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

```

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

```

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

```

845   \hyxmp@pdfstringdef\hyxmp@value{##1}%
846   \xdef#2{#2\noexpand\do {\hyxmp@cur@lang} {\hyxmp@value}}%
847 }
848 }

```

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

```
849 \hyxmp@LA@accept{pdftitle}{\hyxmp@alt@title}
850 \hyxmp@LA@accept{pdfsubject}{\hyxmp@alt@description}
851 \hyxmp@LA@accept{pdfcopyright}{\hyxmp@alt@rights}
```

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

```
852 \newcommand{\XMPLangAlt}[2]{%
853   \let\do=\relax
```

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

```
854   \edef\hyxmp@cur@lang{#1}%
855   \setkeys{hyxmp@LA}{#2}%
856 }
```

### 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  $\text{\TeX}$  and is good enough for producing the XMP `xmpMM:DocumentID` and `xmpMM:InstanceID` fields.

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

```
857 \def\hyxmp@modulo@a#1{%
858   \@tempcntb=\@tempcnta
859   \divide\@tempcntb by #1
860   \multiply\@tempcntb by #1
861   \advance\@tempcnta by -\@tempcntb
862 }
```

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

```
\hyxmp@big@prime@ii 863 \def\hyxmp@big@prime{536870923}
864 \def\hyxmp@big@prime@ii{536870027}
```

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

```
\hyxmp@one@token 865 \def\hyxmp@seed@rng#1{%
866   \@tempcnta=\hyxmp@big@prime
867   \futurelet\hyxmp@one@token\hyxmp@seed@rng@i#1\@empty
868 }
```

`\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`



```

869 \def\hyxmp@seed@rng@if{%
870   \ifx\hyxmp@one@token\empty
871     \let\next=\relax
872   \else
873     \def\next##1{%
874       \multiply\@tempcnta by 3
875       \advance\@tempcnta by '##1
876       \hyxmp@modulo@a{\hyxmp@big@prime}%
877       \futurelet\hyxmp@one@token\hyxmp@seed@rng@if
878     }%
879   \fi
880   \next
881 }

```

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

```

882 \def\hyxmp@set@rand@num{%
883   \@tempcnta=\hyxmp@rand@num
884   \multiply\@tempcnta by 3
885   \advance\@tempcnta by \hyxmp@big@prime@ii
886   \hyxmp@modulo@a{\hyxmp@big@prime}%
887   \xdef\hyxmp@rand@num{\the\@tempcnta}%
888 }

```

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

```

889 \def\hyxmp@append@hex#1{%
890   \hyxmp@set@rand@num
891   \@tempcnta=\hyxmp@rand@num
892   \hyxmp@modulo@a{16}%
893   \ifnum\@tempcnta<10
894     \xdef#1{#1\the\@tempcnta}%
895   \else

```

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

```

896     \advance\@tempcnta by -10
897     \ifcase\@tempcnta
898       \xdef#1{#1a}%
899       \or\xdef#1{#1b}%
900       \or\xdef#1{#1c}%
901       \or\xdef#1{#1d}%
902       \or\xdef#1{#1e}%
903       \or\xdef#1{#1f}%
904     \fi
905   \fi
906 }

```

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

```

907 \def\hyxmp@append@hex@iii#1{%
908   \hyxmp@append@hex#1%
909   \hyxmp@append@hex#1%
910   \hyxmp@append@hex#1%
911 }

```

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

```

912 \def\hyxmp@append@hex@iv#1{%
913   \hyxmp@append@hex@iii#1%
914   \hyxmp@append@hex#1%
915 }

```

\hyxmp@create@uuid As per the definition of a version 4 UUID [11], define macro #1 as a UUID of the form “`uuid:xxxxxxxx-xxx-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.

```

916 \def\hyxmp@create@uuid#1{%
917   \def#1{uuid:}%
918   \hyxmp@append@hex@iv#1%
919   \hyxmp@append@hex@iv#1%
920   \g@addto@macro#1{-}%
921   \hyxmp@append@hex@iv#1%
922   \g@addto@macro#1{-4}%
923   \hyxmp@append@hex@iii#1%
924   \g@addto@macro#1{-}%

```

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

```

925   \hyxmp@set@rand@num
926   \@tempcnta=\hyxmp@rand@num
927   \hyxmp@modulo@a{4}%
928   \ifcase\@tempcnta
929     \g@addto@macro#1{8}%
930     \or\g@addto@macro#1{9}%
931     \or\g@addto@macro#1{a}%
932     \or\g@addto@macro#1{b}%
933   \fi
934   \hyxmp@append@hex@iii#1%
935   \g@addto@macro#1{-}%
936   \hyxmp@append@hex@iv#1%
937   \hyxmp@append@hex@iv#1%
938   \hyxmp@append@hex@iv#1%
939 }

```

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

```

940 \newcommand*{\hyxmp@def@DocumentID}{%
941   \edef\hyxmp@seed@string{\jobname:\@pdftitle:\@pdfauthor:}%

```

```

942 \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
943 \edef\hyxmp@rand@num{\the\@tempcnta}%
944 \hyxmp@create@uuid\hyxmp@DocumentID
945 }

```

`\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).

```

946 \newcommand*{\hyxmp@def@InstanceID}{%
947   \hyxmp@today@xmp@define{\hyxmp@seed@string}%
948   \edef\hyxmp@seed@string{%
949     \jobname:\@pdftitle:\@pdfauthor:\hyxmp@today@xmp:\hyxmp@seed@string
950   }%
951   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
952   \edef\hyxmp@rand@num{\the\@tempcnta}%
953   \hyxmp@create@uuid\hyxmp@InstanceID
954 }

```

### 3.5 Constructing the XMP packet

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

#### 3.5.1 XMP utility functions

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

```

955 \newcommand*{\hyxmp@add@to@xml}[1]{%
956   \bgroup
957   \@tempcnta=0
958   \ifhyxmp@unicodetex
959     \@tempcntb=65536%

```

```

960     \else
961         \@tempcntb=256%
962     \fi
963 \loop
964     \lccode\@tempcnta=\@tempcnta
965     \advance\@tempcnta by 1
966     \ifnum\@tempcnta<\@tempcntb
967 \repeat
968 \lccode'\_='\ \relax
969 \lccode'\^C='\ \relax
970 \lccode'\^U='\_ \relax
971 \lowercase{\xdef\hyxmp@new@xml{#1}}%
972 \xdef\hyxmp@xml{\hyxmp@xml\hyxmp@new@xml}%
973 \egroup
974 }

```

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

```

975 \bgroup
976 \catcode'\#=11
977 \gdef\hyxmp@hash{#}
978 \egroup

```

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

```

979 \bgroup
980 \xdef\hyxmp@xml{%
981 \hyxmp@add@to@xml{%
982 -----~J%
983 }
984 \xdef\hyxmp@padding{\hyxmp@xml}%
985 \egroup
986 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
987 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
988 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
989 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
990 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}

```

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

### 3.5.2 The Adobe PDF schema

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

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

```

\pdfproducer Define \pdfproducer using the banner string if available or the TEX engine's
\hyxmp@define@pdfproducer version number if not.
992 \newcommand*{\hyxmp@define@pdfproducer}{%
993   \gdef\pdfproducer{TeX}
994   \ifLuaTeX
995     \expandafter\hyxmp@banner@to@producer\expandafter{\luatexbanner}%
996   \else
997     \ifPDFTeX
998       \expandafter\hyxmp@banner@to@producer\expandafter{\pdftexbanner}%
999     \else
1000       \ifXeTeX
1001         \edef\pdfproducer{XeTeX version \the\XeTeXversion\XeTeXrevision}%
1002       \fi
1003     \fi
1004   \fi
1005 }

\pdfproducer Define \pdfproducer as the TEX engine's banner string (e.g., "This is pdfTeX,
\hyxmp@banner@to@producer Version 3.14159265-2.6-1.40.20 (TeX Live 2019/Debian) kpathsea
version 6.3.1"), removing the initial "This is" if possible (specifically, when
ε-TEX's \scantokens primitive is available).
1006 \def\hyxmp@banner@to@producer#1{%
1007   \ifx\scantokens\relax
1008     \gdef\pdfproducer{#1}%
1009   \else
1010     {\scantokens{\makeatletter\hyxmp@remove@this#1\relax}}%
1011   \fi
1012 }

\pdfproducer Define \pdfproducer as a given banner string with the initial "This is" stripped
\hyxmp@remove@this off the beginning.
1013 \def\hyxmp@remove@this This is #1\relax{\gdef\pdfproducer{#1}}

If pdfproducer wasn't specified and hyperref didn't already define
\pdfproducer—old versions of hyperref did; newer ones don't—try to assign
a meaningful producer string and use that.
1014 \AtBeginDocument{%
1015   \ifx\pdfproducer\relax
1016     \hyxmp@define@pdfproducer
1017   \fi
1018 }

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

```

\hyxmp@major@minor

1019 \newcommand*{\hyxmp@assign@major@minor}{%
1020   \@ifundefined{pdfvariable}{%
1021     \@ifundefined{pdfminorversion}{%
      Case 1: Neither \pdfvariable nor \pdfminorversion is defined (XeLaTeX and
      regular LATEX).
1022   }{%
      Case 2: \pdfminorversion is defined (pdfLATEX and pre-0.85 LuaLATEX).
1023     \xdef\hyxmp@major@minor{\the\pdfminorversion}%
1024     \@ifundefined{pdfmajorversion}{%
      Case 2(a): \pdfmajorversion is not defined (older versions of pdfLATEX and
      LuaLATEX).
1025       \xdef\hyxmp@major@minor{1.\hyxmp@major@minor}%
1026     }{%
      Case 2(b): \pdfmajorversion is defined (pdfLATEX 1.40.21+).
1027       \xdef\hyxmp@major@minor{\the\pdfmajorversion.\hyxmp@major@minor}%
1028     }%
1029   }%
1030 }{%
      Case 3: \pdfvariable is defined (LuaLATEX 0.85+).
1031   \xdef\hyxmp@major@minor{\the\pdfvariable majorversion.\the\pdfvariable minorversion}%
1032 }%
1033 }

\hyxmp@pdf@schema  Add properties defined by the Adobe PDF schema to the \hyxmp+xml macro.
1034 \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.
1035   \hyxmp@add@simple@var{pdf:Producer}{@pdfproducer}%
1036   \hyxmp@add@simple@var{pdf:Keywords}{@pdfkeywords}%
1037   \hyxmp@add@simple{pdf:Trapped}{\@pdftrapped}%
1038   \hyxmp@assign@major@minor
1039   \hyxmp@add@simple@var{pdf:PDFVersion}{\hyxmp@major@minor}%
1040 }

```

### 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.

```
1041 \newcommand*{\hyxmp@rdf@dc}[3][\iffalse]{%
```

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

```
1042 \ifmtargexp{#3}{\@tempswafalse}{\@tempswatrue}%
```

```
1043 #1
```

```
1044 \@tempswatrue
```

```
1045 \fi
```

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

```
1046 \if@tempswa
```

```
1047 \hyxmp@xmlify{#3}%
```

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

```
1048 \let\hyxmp@value=\hyxmp@xmlified
```

```
1049 \hyxmp@add@to@xml{%
```

```
1050 -----<dc:#2>^^J%
```

```
1051 -----<rdf:Alt>^^J%
```

```
1052 }%
```

```
1053 \ifx\@pdfmetalang\hyxmp@x@default
```

```
1054 \else
```

```
1055 \hyxmp@xmlify{\@pdfmetalang}%
```

```
1056 \hyxmp@add@to@xml{%
```

```
1057 -----<rdf:li xml:lang="\hyxmp@xmlified">\hyxmp@value</rdf:li>^^J%
```

```
1058 }%
```

```
1059 \fi
```

```
1060 \hyxmp@add@to@xml{%
```

```
1061 -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@value</rdf:li>^^J%
```

```
1062 }%
```

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

```
1063 \def\do##1##2{
```

```
1064 \hyxmp@xmlify{##2}%
```

```
1065 \hyxmp@add@to@xml{%
```

```
1066 -----<rdf:li xml:lang="##1">\hyxmp@xmlified</rdf:li>^^J%
```

```
1067 }%
```

```
1068 }%
```

```
1069 \csname hyxmp@alt@#2\endcsname
```

Complete this XMP element.

```
1070 \hyxmp@add@to@xml{%
```

```
1071 -----</rdf:Alt>^^J%
```

```
1072 -----</dc:#2>^^J%
```

```
1073 }%
```

```

1074 \fi
1075 }%

\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.
1076 \newcommand*{\hyxmp@list@to@xml}[4][\iffalse]{%
    Set \@tempswatrue only if the given list is nonempty or the provided conditional
    evaluates to TRUE.
1077 \ifmtargexp{#4}{\@tempswafalse}{\@tempswatrue}%
1078 #1
1079 \@tempswatrue
1080 \fi
    Append the corresponding XML only if \@tempswatrue.
1081 \if@tempswa
1082 \hyxmp@add@to@xml{%
1083 -----<dc:#2>^^J%
1084 -----<rdf:#3>^^J%
1085 }%
1086 \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.
1087 \hyxmp@xmlify{#4}%
1088 \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
1089 \def\@elt##1{%
1090 \hyxmp@add@to@xml{%
1091 -----<rdf:li>##1</rdf:li>^^J%
1092 }%
1093 }%
1094 \hyxmp@list
1095 \egroup
1096 \hyxmp@add@to@xml{%
1097 -----</rdf:#3>^^J%
1098 -----</dc:#2>^^J%
1099 }%
1100 \fi
1101 }

\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.

1102 \newcommand{\hyxmp@singleton@dc}[3][Bag]{%
1103 \ifnotmtarg{#3}{%
1104 \hyxmp@xmlify{#3}%
1105 \hyxmp@add@to@xml{%
1106 -----<dc:#2>^^J%
1107 -----<rdf:#1>^^J%

```



```

1108 -----<rdf:li>\hyxmp@xmlified</rdf:li>^^J%
1109 -----</rdf:#1>^^J%
1110 -----</dc:#2>^^J%
1111     }%
1112 }
1113 }

```

`\hyxmp@cond@dc@identifier` Conditionally add a `dc:identifier` tag. Given a prefix string (#1) and a main string (#2), wrap these in a `dc:identifier` if the main string is nonempty and `\hyxmp@xmlified` is empty (implying the `dc:identifier` has not yet been written).

```

1114 \newcommand*{\hyxmp@cond@dc@identifier}[2]{%
1115   \ifx\hyxmp@xmlified\@empty
1116     \@ifnotmtargexp{#2}{%
1117       \hyxmp@add@simple@pfx{dc:identifier}{#1}{#2}%
1118     }%
1119   \fi
1120 }

```

`\hyxmp@dc@schema` Add properties defined by the Dublin Core schema to the `\hyxmp@xml` macro. Specifically, we add entries for the `dc:title` property if the author specified a `pdftitle`, the `dc:description` property if the author specified a `pdfsubject`, the `dc:rights` property if the author specified a `pdfcopyright`, the `dc:creator` property if the author specified a `pdfauthor`, the `dc:subject` property if the author specified `pdfkeywords`, the `dc:language` property if the author specified `pdflang`, the `dc:type` property if the author specified `pdftype`, and the `dc:identifier` if the author specified `pdfidentifier` or if we can derive it from other options. We also specify the `dc:date` property using the date the document was run through L<sup>A</sup>T<sub>E</sub>X and the `dc:source` property using the base name of the source file with `.tex` appended.

```

1121 \newcommand*{\hyxmp@dc@schema}{%
1122   \hyxmp@add@simple{dc:format}{application/pdf}%
1123   \hyxmp@rdf@dc[\ifHy@pdfa]{title}{\@pdftitle}%
1124   \hyxmp@rdf@dc[\ifHy@pdfa]{description}{\@pdfsubject}%
1125   \hyxmp@rdf@dc{rights}{\@pdfcopyright}%
1126   \hyxmp@singleton@dc{publisher}{\@pdfpublisher}%
1127   \hyxmp@singleton@dc[Seq]{date}{\hyxmp@today@xmp}%
1128   \hyxmp@singleton@dc{language}{\@pdflang}%
1129   \hyxmp@singleton@dc{type}{\@pdftype}%
1130   \hyxmp@list@to@xml[\ifHy@pdfa]{creator}{Seq}{\hyxmp@pdfauthor}%
1131   \hyxmp@list@to@xml{subject}{Bag}{\hyxmp@pdfkeywords}%
1132   \ifx\@pdfsource\@empty
1133     \else
1134       \hyxmp@add@simple{dc:source}{\@pdfsource}%
1135     \fi

```

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

```

1136   \@ifmtargexp{\@pdfidentifier}{%
1137     \let\hyxmp@xmlified=\@empty
1138     \hyxmp@cond@dc@identifier{info:doi/}{\@pdfdoi}%

```

```

1139 \hyxmp@cond@dc@identifier{urn:ISSN:}{\@pdfeissn}%
1140 \hyxmp@cond@dc@identifier{urn:ISSN:}{\@pdfissn}%
1141 \hyxmp@cond@dc@identifier{urn:ISBN:}{\@pdfisbn}%
1142 }{%
1143 \hyxmp@add@simple{dc:identifier}{\@pdfidentifier}%
1144 }%
1145 }

```

### 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.

```

1146 \newcommand*{\hyxmp@xmpRights@schema}{%

```

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

```

1147 \let\hyxmp@rights=\@empty
1148 \ifx\@pdflicenseurl\@empty
1149 \else
1150 \def\hyxmp@rights{YES}%
1151 \fi
1152 \ifx\@pdfcopyright\@empty
1153 \else
1154 \def\hyxmp@rights{YES}%
1155 \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.

```

1156 \ifx\hyxmp@rights\@empty
1157 \else
1158 \ifx\@pdfcopyright\@empty
1159 \else
1160 \hyxmp@add@simple{xmpRights:Marked}{True}%
1161 \fi
1162 \hyxmp@add@simple{xmpRights:WebStatement}{\@pdflicenseurl}%
1163 \fi
1164 }

```

### 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`.

```

1165 \gdef\hyxmp@mm@schema{%
1166   \ifmtargexp{\hyxmp@DocumentID}{\hyxmp@def@DocumentID}{}%
1167   \ifmtargexp{\hyxmp@InstanceID}{\hyxmp@def@InstanceID}{}%
1168   \hyxmp@add@simple{xmpMM:DocumentID}{\hyxmp@DocumentID}%
1169   \hyxmp@add@simple{xmpMM:InstanceID}{\hyxmp@InstanceID}%
1170   \hyxmp@add@simple{xmpMM:VersionID}{\@pdfversionid}%
1171   \hyxmp@add@simple{xmpMM:RenditionClass}{\@pdfrendition}%
1172 }

```

### 3.5.6 The XMP Basic schema

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

```

1173 \newcommand*{\hyxmp@xmp@basic@schema}{%
    For the document's creation date, use the user-specified \@pdfcreationdate if
    defined and non-empty. Otherwise use our fabricated \hyxmp@today@xmp.

```

```

1174   \ifmtargexp{\@pdfcreationdate}{%
1175     \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@today@xmp}%
1176   }{%
1177     \hyxmp@add@simple{xmp:CreateDate}{%
1178       \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfcreationdate}}%
1179   }%

```

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

```

1180   \ifmtargexp{\@pdfmoddate}{%
1181     \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@today@xmp}%
1182   }{%
1183     \hyxmp@add@simple{xmp:ModifyDate}{%
1184       \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfmoddate}}%
1185   }%

```

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

```

1186   \ifmtargexp{\@pdfmetadatetime}{%
1187     \hyxmp@add@simple{xmp:MetadataDate}{\hyxmp@today@xmp}%
1188   }{%
1189     \hyxmp@add@simple{xmp:MetadataDate}{\@pdfmetadatetime}%
1190   }%

```

Define the creation tool and the base URL.

```

1191   \hyxmp@add@simple{xmp:CreatorTool}{\@pdfcreator}%
1192   \hyxmp@add@simple{xmp:BaseURL}{\@baseurl}%
1193 }

```

### 3.5.7 The Photoshop schema

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

properties.

```

1194 \gdef\hyxmp@photoshop@schema{%
1195   \edef\hyxmp@photoshop@data{\@pdfauthortitle\@pdfcaptionwriter}%
1196   \hyxmp@add@simple{photoshop:AuthorsPosition}{\@pdfauthortitle}%
1197   \hyxmp@add@simple{photoshop:CaptionWriter}{\@pdfcaptionwriter}%
1198 }

```

### 3.5.8 PDF/\* Identification schemata

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

```

1199 \newcommand*{\hyxmp@pdfa@id@schema}{%
1200   \ifHy@pdfa
1201     \hyxmp@add@simple{pdfaid:part}{\@pdfapart}%
1202     \hyxmp@add@simple{pdfaid:conformance}{\@pdfaconformance}%
1203   \fi
1204 }

```

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

```

1205 \newcommand*{\hyxmp@pdfua@id@schema}{%
1206   \hyxmp@add@simple{pdfuaid:part}{\@pdfuapart}%
1207 }

```

`\hyxmp@pdfx@id@schema` If the document conforms to a PDF/X standard, the author can indicate the standard version with `pdfxstandard`. We separately handle PDF/X-1, PDF/X-2 and PDF/X-3, and PDF/X-4 onwards.

```

1208 \newcommand*{\hyxmp@pdfx@id@schema}{%
1209   \@tempcnta=0\hyxmp@pdfx@major\relax
1210   \ifnum\@tempcnta=0
1211     \else
1212       \ifnum\@tempcnta=1
1213         \hyxmp@add@simple{pdfx:GTS_PDFXVersion}{PDF/X-1:2001}%
1214         \hyxmp@add@simple{pdfx:GTS_PDFXConformance}{\@pdfxstandard}%
1215       \else
1216         \ifnum\@tempcnta<4
1217           \hyxmp@add@simple{pdfx:GTS_PDFXVersion}{\@pdfxstandard}%
1218         \else
1219           \hyxmp@add@simple{pdfxid:GTS_PDFXVersion}{\@pdfxstandard}%
1220         \fi
1221       \fi
1222     \fi
1223 }

```

### 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.

```
1224 \begingroup
1225   \catcode'\&=12
1226   \catcode'\#=12
1227   \gdef\xmplinesep{&#xA;}
1228 \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.

```
1229 \newcommand*{\hyxmp@list@to@lines}[2]{%
1230   \ifnotmtargexp{#2}{%
1231     \bgroup
1232     \hyxmp@add@to@xml{%
1233       \hyxmp@extra@indent_____<#1>%
1234     }%
```

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

```
1235   \def\@elt@first##1{%
1236     \hyxmp@add@to@xml{##1}%
1237     \let\@elt=\@elt@rest
1238   }%
```

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

```
1239   \def\@elt@rest##1{%
1240     \hyxmp@add@to@xml{\xmplinesep##1}%
1241   }%
```

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

```
1242   \let\@elt=\@elt@first
1243   \hyxmp@xmlify{#2}%
1244   \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
1245   \hyxmp@list
1246   \hyxmp@add@to@xml{</#1>^J}%
1247 \egroup
1248 }%
1249 }
```

`\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.

```
1250 \gdef\hyxmp@iptc@schema{%
```

Because we currently support only `Iptc4xmpCore:CreatorContactInfo` it suffices to check if we have any relevant data. If so, we instantiate a `Iptc4xmpCore:ContactInfo` structure with all available fields.

```

1251 \ifx\hyxmp@iptc@data\@empty
1252 \else
1253   \hyxmp@add@to@xml{%
1254   -----<Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">^^J%
1255   }%

```

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

```

1256   \bgroup
1257   \edef\hyxmp@extra@indent{\hyxmp@extra@indent\space\space}%
1258   \hyxmp@list@to@lines{Iptc4xmpCore:CiAdrExtadr}{\@pdfcontactaddress}%
1259   \hyxmp@add@simple{Iptc4xmpCore:CiAdrCity}{\@pdfcontactcity}%
1260   \hyxmp@add@simple{Iptc4xmpCore:CiAdrRegion}{\@pdfcontactregion}%
1261   \hyxmp@add@simple{Iptc4xmpCore:CiAdrPcode}{\@pdfcontactpostcode}%
1262   \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.

```

1263   \def\xmplinesep{,}%
1264   \hyxmp@list@to@lines{Iptc4xmpCore:CiTelWork}{\@pdfcontactphone}%
1265   \hyxmp@list@to@lines{Iptc4xmpCore:CiEmailWork}{\@pdfcontactemail}%
1266   \hyxmp@list@to@lines{Iptc4xmpCore:CiUrlWork}{\@pdfcontacturl}%
1267   \egroup
1268   \hyxmp@add@to@xml{%
1269   -----</Iptc4xmpCore:CreatorContactInfo>^^J%
1270   }%
1271 \fi
1272 }

```

### 3.5.10 The PRISM Basic Metadata schema

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

```

1273 \newcommand*{\hyxmp@prism@schema}{%
1274 \ifx\hyxmp@prism@data\@empty
1275 \else
1276   \hyxmp@add@simple{prism:complianceProfile}{three}%
1277 \fi
1278 \hyxmp@add@simple@lang{prism:subtitle}{\@pdfsubtitle}%
1279 \hyxmp@add@simple@lang{prism:publicationName}{\@pdfpublication}%
1280 \hyxmp@add@simple{prism:aggregationType}{\@pdfpubtype}%
1281 \hyxmp@add@simple@lang{prism:bookEdition}{\@pdfbookedition}%

```

```

1282 \hyxmp@add@simple{prism:volume}{\@pdfvolumenum}%
1283 \hyxmp@add@simple{prism:number}{\@pdfissuenum}%
1284 \hyxmp@add@simple{prism:pageRange}{\@pdfpagerange}%
1285 \hyxmp@add@simple{prism:isbn}{\@pdfisbn}%
1286 \hyxmp@add@simple{prism:issn}{\@pdfissn}%
1287 \hyxmp@add@simple{prism:eIssn}{\@pdfeissn}%
1288 \hyxmp@add@simple{prism:doi}{\@pdfdoi}%
1289 \hyxmp@add@simple{prism:url}{\@pdfurl}%
1290 \hyxmp@add@simple{prism:byteCount}{\@pdfbytes}%
1291 \hyxmp@add@simple{prism:pageCount}{\@pdfnumpages}%
1292 }

```

### 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.

```
1293 \newcommand*{\hyxmp@check@iptc@data}{%
```

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

```

1294 \edef\hyxmp@iptc@data{%
1295   \@pdfcontactaddress
1296   \@pdfcontactcity
1297   \@pdfcontactregion
1298   \@pdfcontactpostcode
1299   \@pdfcontactcountry
1300   \@pdfcontactphone
1301   \@pdfcontactemail
1302   \@pdfcontacturl
1303 }%
1304 }%

```

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

```
1305 \newcommand*{\hyxmp@check@prism@data}{%
```

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

```

1306 \edef\hyxmp@prism@data{%
1307   \@pdfbookedition
1308   \@pdfbytes
1309   \@pdfdoi
1310   \@pdfeissn
1311   \@pdfisbn
1312   \@pdfissn

```

```

1313 \pdfissuenum
1314 \pdfnumpages
1315 \pdfpagerange
1316 \pdfpublication
1317 \pdfpubtype
1318 \pdfsubtitle
1319 \pdfurl
1320 \pdfvolumenum
1321 }%
1322 }%

```

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

```

1323 \newcommand*{\hyxmp@begin@extension@decls}{%
1324 \hyxmp@add@to@xml{%
1325 -----<pdfaExtension:schemas>^^J%
1326 -----<rdf:Bag>^^J%
1327 }%
1328 }

```

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

```

1329 \newcommand*{\hyxmp@end@extension@decls}{%
1330 \hyxmp@add@to@xml{%
1331 -----</rdf:Bag>^^J%
1332 -----</pdfaExtension:schemas>^^J%
1333 }%
1334 }

```

`\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.

```

1335 \newcommand*{\hyxmp@begin@ext@decl}[3]{%
1336 \hyxmp@add@to@xml{%
1337 -----<rdf:li rdf:parseType="Resource">^^J%
1338 -----<pdfaSchema:schema>#1</pdfaSchema:schema>^^J%
1339 -----<pdfaSchema:prefix>#2</pdfaSchema:prefix>^^J%
1340 -----<pdfaSchema:namespaceURI>#3</pdfaSchema:namespaceURI>^^J%
1341 -----<pdfaSchema:property>^^J%
1342 -----<rdf:Seq>^^J%
1343 }%
1344 }%

```

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

```

1345 \newcommand*{\hyxmp@end@ext@decl}{%
1346 \hyxmp@add@to@xml{%
1347 -----</rdf:Seq>^^J%
1348 -----</pdfaSchema:property>^^J%
1349 -----</rdf:li>^^J%
1350 }%
1351 }%

```



`\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.

```

1352 \newcommand{\hyxmp@declare@property}[4][Text]{%
1353   \hyxmp@add@to@xml{%
1354     -----<rdf:li rdf:parseType="Resource">^^J%
1355     -----<pdfaProperty:name>}%
1356   \xdef\hyxmp@xml{\hyxmp@xml#2}%
1357   \hyxmp@add@to@xml{</pdfaProperty:name>^^J%
1358     -----<pdfaProperty:valueType>#1</pdfaProperty:valueType>^^J%
1359     -----<pdfaProperty:category>#3</pdfaProperty:category>^^J%
1360     -----<pdfaProperty:description>#4</pdfaProperty:description>^^J%
1361     -----</rdf:li>^^J%
1362   }%
1363 }%

```

`\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.

```

1364 \newcommand{\hyxmp@declare@field}[3][Text]{%
1365   \hyxmp@add@to@xml{%
1366     -----<rdf:li rdf:parseType="Resource">^^J%
1367     -----<pdfaField:name>#2</pdfaField:name>^^J%
1368     -----<pdfaField:valueType>#1</pdfaField:valueType>^^J%
1369     -----<pdfaField:description>#3</pdfaField:description>^^J%
1370     -----</rdf:li>^^J%
1371   }%
1372 }%

```

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

```

1373 \newcommand*{\hyxmp@pdf@extensions}{%
1374   \hyxmp@begin@ext@decl
1375     {Adobe PDF Schema}%
1376     {pdf}%
1377     {http://ns.adobe.com/pdf/1.3/}%
1378   \hyxmp@declare@property
1379     {Trapped}%
1380     {internal}%
1381     {Indication if the document has been modified to include trapping information}%
1382   \hyxmp@end@ext@decl
1383 }%

```

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

```

1384 \newcommand*{\hyxmp@mm@extensions}{%
1385   \hyxmp@begin@ext@decl
1386     {XMP Media Management Schema}%
1387     {xmpMM}%
1388     {http://ns.adobe.com/xap/1.0/mm/}%
1389   \hyxmp@declare@property

```

```

1390     [URI]
1391     {DocumentID}%
1392     {internal}%
1393     {UUID based identifier for all versions and renditions of a document}%
1394 \hyxmp@declare@property
1395     [URI]
1396     {InstanceID}%
1397     {internal}%
1398     {UUID based identifier for specific incarnation of a document}%
1399 \hyxmp@declare@property
1400     {VersionID}%
1401     {internal}%
1402     {Document version identifier}%
1403 \hyxmp@declare@property
1404     {RenditionClass}%
1405     {internal}%
1406     {The manner in which a document is rendered}%
1407 \hyxmp@end@ext@decl
1408 }%

```

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

```

1409 \newcommand*{\hyxmp@pdfa@id@extensions}{%
1410 \hyxmp@begin@ext@decl
1411     {PDF/A Identification Schema}%
1412     {pdfaid}%
1413     {http://www.aiim.org/pdfa/ns/id/}%
1414 \hyxmp@declare@property
1415     [Integer]%
1416     {part}%
1417     {internal}%
1418     {Part of PDF/A standard}%
1419 \hyxmp@declare@property
1420     {conformance}%
1421     {internal}%
1422     {Conformance level of PDF/A standard}%
1423 \hyxmp@end@ext@decl
1424 }%

```

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

```

1425 \newcommand*{\hyxmp@pdfua@id@extensions}{%
1426 \hyxmp@begin@ext@decl
1427     {PDF/UA Universal Accessibility Schema}%
1428     {pdfuaid}%
1429     {http://www.aiim.org/pdfua/ns/id/}%
1430 \hyxmp@declare@property
1431     [Integer]%
1432     {part}%
1433     {internal}%
1434     {Part of ISO 14289 standard}%
1435 \hyxmp@end@ext@decl

```

1436 }%

`\hyxmp@pdfx@id@extensions` Declare the schema used pre-PDF/X-4. Because Adobe Acrobat DC (at least) defines this even for PDF/X-4 and later, we follow suit.

```
1437 \newcommand*{\hyxmp@pdfx@id@extensions}{%
1438   \ifx\hyxmp@pdfx@major\empty
1439   \else
1440     \hyxmp@begin@ext@decl
1441       {Adobe Document Info PDF/X eXtension Schema}%
1442       {pdfx}%
1443       {http://ns.adobe.com/pdfx/1.3/}%
1444     \hyxmp@declare@property
1445       {GTS_PDFXVersion}%
1446       {internal}%
1447       {ID of PDF/X standard}%
1448     \hyxmp@declare@property
1449       {GTS_PDFXConformance}%
1450       {internal}%
1451       {Conformance level of PDF/X standard}%
1452     \hyxmp@end@ext@decl
1453   \fi
```

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

```
1454   \@tempcnta=0\hyxmp@pdfx@major\relax
1455   \ifnum\@tempcnta>3
1456     \hyxmp@begin@ext@decl
1457       {PDF/X ID Schema}%
1458       {pdfxid}%
1459       {http://www.npes.org/pdfx/ns/id/}%
1460     \hyxmp@declare@property
1461       {GTS_PDFXVersion}%
1462       {internal}%
1463       {ID of PDF/X standard}%
1464     \hyxmp@end@ext@decl
1465   \fi
1466 }%
```

`\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.

```
1467 \newcommand*{\hyxmp@iptc@extensions}{%
1468   \hyxmp@begin@ext@decl
1469     {IPTC Core Schema}%
1470     {Iptc4xmpCore}%
1471     {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}%
1472   \hyxmp@declare@property
1473     [ContactInfo]
1474     {CreatorContactInfo}
1475     {external}
```

1476       {Document creator's contact information}

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

```
1477 \hyxmp@add@to+xml{%
1478 -----</rdf:Seq>^^J%
1479 -----</pdfaSchema:property>^^J%
1480 -----<pdfaSchema:valueType>^^J%
1481 -----<rdf:Seq>^^J%
1482 -----<rdf:li rdf:parseType="Resource">^^J%
1483 -----<pdfaType:type>ContactInfo</pdfaType:type>^^J%
1484 -----<pdfaType:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaType:
1485 -----<pdfaType:prefix>Iptc4xmpCore</pdfaType:prefix>^^J%
1486 -----<pdfaType:description>%
1487       Basic set of information to get in contact with a person%
1488       </pdfaType:description>^^J%
1489 -----<pdfaType:field>^^J%
1490 -----<rdf:Seq>^^J%
1491 }%
1492 \hyxmp@declare@field
1493     {CiAdrCity}%
1494     {Contact information city}%
1495 \hyxmp@declare@field
1496     {CiAdrCtry}%
1497     {Contact information country}%
1498 \hyxmp@declare@field
1499     {CiAdrExtadr}%
1500     {Contact information address}%
1501 \hyxmp@declare@field
1502     {CiAdrPcode}%
1503     {Contact information local postal code}%
1504 \hyxmp@declare@field
1505     {CiAdrRegion}%
1506     {Contact information regional information such as state or province}%
1507 \hyxmp@declare@field
1508     {CiEmailWork}%
1509     {Contact information email address(es)}%
1510 \hyxmp@declare@field
1511     {CiTelWork}%
1512     {Contact information telephone number(s)}%
1513 \hyxmp@declare@field
1514     {CiUrlWork}%
1515     {Contact information Web URL(s)}%
1516 \hyxmp@add@to+xml{%
1517 -----</rdf:Seq>^^J%
1518 -----</pdfaType:field>^^J%
1519 -----</rdf:li>^^J%
1520 -----</rdf:Seq>^^J%
1521 -----</pdfaSchema:valueType>^^J%
1522 -----</rdf:li>^^J%
```

```

1523 }%
1524 }

```

`\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.

```

1525 \newcommand*{\hyxmp@prism@extensions}{%
1526   \hyxmp@begin@ext@decl
1527     {PRISM Basic Metadata}%
1528     {prism}%
1529     {http://prismstandard.org/namespaces/basic/2.1/}%
1530   \hyxmp@declare@property
1531     {complianceProfile}%
1532     {internal}%
1533     {PRISM specification compliance profile to which this document adheres}%
1534   \hyxmp@declare@property
1535     {publicationName}%
1536     {external}%
1537     {Publication name}%
1538   \hyxmp@declare@property
1539     {aggregationType}%
1540     {external}%
1541     {Publication type}%
1542   \hyxmp@declare@property
1543     {bookEdition}%
1544     {external}%
1545     {Edition of the book in which the document was published}%
1546   \hyxmp@declare@property
1547     {volume}%
1548     {external}%
1549     {Publication volume number}%
1550   \hyxmp@declare@property
1551     {number}%
1552     {external}%
1553     {Publication issue number within a volume}%
1554   \hyxmp@declare@property
1555     {pageRange}%
1556     {external}%
1557     {Page range for the document within the print version of its publication}%
1558   \hyxmp@declare@property
1559     {issn}%
1560     {external}%
1561     {ISSN for the printed publication in which the document was published}%
1562   \hyxmp@declare@property
1563     {eIssn}%
1564     {external}%
1565     {ISSN for the electronic publication in which the document was published}%
1566   \hyxmp@declare@property
1567     {isbn}%

```

```

1568         {external}%
1569         {ISBN for the publication in which the document was published}%
1570 \hyxmp@declare@property
1571         {doi}%
1572         {external}%
1573         {Digital Object Identifier for the document}%
1574 \hyxmp@declare@property
1575         [URL]
1576         {url}%
1577         {external}%
1578         {URL at which the document can be found}%
1579 \hyxmp@declare@property
1580         [Integer]
1581         {byteCount}%
1582         {internal}%
1583         {Approximate file size in octets}%
1584 \hyxmp@declare@property
1585         [Integer]
1586         {pageCount}%
1587         {internal}%
1588         {Number of pages in the print version of the document}%
1589 \hyxmp@declare@property
1590         {subtitle}%
1591         {external}%
1592         {Document's subtitle}%
1593 \hyxmp@end@ext@decl
1594 }%

```

`\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.

```

1595 \newcommand*{\hyxmp@declare@extensions}{%

```

```

1596 \hyxmp@begin@extension@decls

```

Declare the Adobe PDF schema (always present).

```

1597 \hyxmp@pdf@extensions

```

Declare the XMP Media Management extensions (always present).

```

1598 \hyxmp@mm@extensions

```

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

```

1599 \ifHy@pdfa

```

```

1600 \hyxmp@pdfa@id@extensions

```

```

1601 \fi

```

Conditionally declare the PDF/UA Universal Accessibility extensions.

```

1602 \ifx\@pdfuapart\@empty

```

```

1603 \else

```

```

1604 \hyxmp@pdfua@id@extensions

```

```

1605 \fi

```

\next Conditionally declare the PDF/X extensions.

```
1606 \ifx\@pdfxversion\@empty
1607 \else
1608 \hyxmp@pdfx@id@extensions
1609 \fi
```

Conditionally declare IPTC photo metadata extensions.

```
1610 \ifx\hyxmp@iptc@data\@empty
1611 \else
1612 \hyxmp@iptc@extensions
1613 \fi
```

Conditionally declare PRISM basic metadata extensions.

```
1614 \ifx\hyxmp@prism@data\@empty
1615 \else
1616 \hyxmp@prism@extensions
1617 \fi

1618 \hyxmp@end@extension@decls
1619 }
```

### 3.5.12 Combining schemata into an XMP packet

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

```
1620 \begingroup
1621 \ifhyxmp@unicodetex
1622 \lccode'\!="FEFF %
1623 \lowercase{%
1624 \gdef\hyxmp@bom{!}%
1625 }%
1626 \else
1627 \catcode'\^^ef=12
1628 \catcode'\^^bb=12
1629 \catcode'\^^bf=12
1630 \gdef\hyxmp@bom{^^ef^^bb^^bf}%
1631 \fi
1632 \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.

```
1633 \def\hyxmp@construct@packet{%
1634 \gdef\hyxmp+xml{%
1635 \hyxmp@add@to+xml{<?xpacket begin="\hyxmp@bom" %
1636 id="W5M0MPCehiHzreSzNTczkc9d"?>^^J%
1637 <x:xmpmeta xmlns:x="adobe:ns:meta/">^^J%
1638 __<rdf:RDF %
1639 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\hyxmp@hash">^^J%
1640 ____<rdf:Description rdf:about="">^^J%
```

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

```

1641 -----xmlns:pdf="http://ns.adobe.com/pdf/1.3/"^^J%
1642 -----xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/"^^J%
1643 -----xmlns:dc="http://purl.org/dc/elements/1.1/"^^J%
1644 -----xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/"^^J%
1645 -----xmlns:xmp="http://ns.adobe.com/xap/1.0/"^^J%
1646 -----xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"^^J%
1647 -----xmlns:stEvt="http://ns.adobe.com/xap/1.0/sType/ResourceEvent\hyxmp@hash"^^J%
1648 -----xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/"^^J%
1649 -----xmlns:pdfuaid="http://www.aiim.org/pdfua/ns/id/"^^J%
1650 -----xmlns:pdfx="http://ns.adobe.com/pdfx/1.3/"^^J%
1651 -----xmlns:pdfxid="http://www.npes.org/pdfx/ns/id/"^^J%
1652 -----xmlns:prism="http://prismstandard.org/namespaces/basic/2.1/"^^J%
1653 -----xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"^^J%
1654 -----xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"^^J%
1655 -----xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema\hyxmp@hash"^^J%
1656 -----xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property\hyxmp@hash"^^J%
1657 -----xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type\hyxmp@hash"^^J%
1658 -----xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field\hyxmp@hash">^^J%
1659 }%

```

Declare non-standard schemata.

```

1660 \hyxmp@check@iptc@data
1661 \hyxmp@check@prism@data
1662 \hyxmp@declare@extensions

```

Insert all the metadata we know how to insert.

```

1663 \hyxmp@pdf@schema
1664 \hyxmp@xmpRights@schema
1665 \hyxmp@dc@schema
1666 \hyxmp@photoshop@schema
1667 \hyxmp@xmp@basic@schema
1668 \hyxmp@pdfa@id@schema
1669 \hyxmp@pdfua@id@schema
1670 \hyxmp@pdfx@id@schema
1671 \hyxmp@mm@schema
1672 \hyxmp@iptc@schema
1673 \hyxmp@prism@schema
1674 \hyxmp@add@to+xml{%
1675 ____</rdf:Description>^^J%
1676 __</rdf:RDF>^^J%
1677 </x:xmpmeta>^^J%
1678 \hyxmp@padding
1679 <?xpacket end="w"?>^^J%
1680 }%
1681 }

```



### 3.6 Embedding the XMP packet

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

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

```

1682 \newcommand*{\hyxmp@embed@packet}{%
1683   \hyxmp@construct@packet
1684   \def\hyxmp@driver{hpdfTeX}%
1685   \ifx\hyxmp@driver\Hy@driver
1686     \hyxmp@embed@packet@pdfTeX
1687   \else
1688     \def\hyxmp@driver{hLuaTeX}%
1689     \ifx\hyxmp@driver\Hy@driver
1690       \hyxmp@embed@packet@LuaTeX
1691     \else
1692       \def\hyxmp@driver{hdvipdfm}%
1693       \ifx\hyxmp@driver\Hy@driver
1694         \hyxmp@embed@packet@dvipdfm
1695       \else
1696         \def\hyxmp@driver{hXeTeX}%
1697         \ifx\hyxmp@driver\Hy@driver
1698           \hyxmp@embed@packet@XeTeX
1699         \else
1700           \@ifundefined{pdfmark}{%
1701             \PackageWarningNoLine{hyperxmp}{%
1702               Unrecognized hyperref driver ‘\Hy@driver’. \MessageBreak
1703               \jobname.tex’s XMP metadata will *not* be \MessageBreak
1704               embedded in the resulting file}%
1705           }{%
1706             \hyxmp@embed@packet@pdfmark
1707           }%
1708         \fi
1709       \fi
1710     \fi
1711   \fi
1712 }
```

#### 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.

```
1713 \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.

```
1714 \newcommand*{\hyxmp@embed@packet@pdftex}{%
1715   \bgroup
1716   \ifluatex
1717   \else
1718     \pdfcompresslevel=0
1719   \fi
1720   \immediate\pdfobj \ifluatex uncompressed\fi stream attr {%
1721     /Type /Metadata
1722     /Subtype /XML
1723   }{\hyxmp@xml}%
1724   \pdfcatalog {/Metadata \the\pdflastobj\space 0 R}%
1725   \egroup
1726 }
```

### 3.6.2 Embedding using LuaTeX 0.85+

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

```
1727 \newcommand*{\hyxmp@embed@packet@luatex}{%
1728   \immediate\pdfextension obj uncompressed stream attr {%
1729     /Type /Metadata
1730     /Subtype /XML
1731   }{\hyxmp@xml}%
1732   \pdfextension catalog {/Metadata \the\numexpr\pdffeedback lastobj\relax\space 0 R}%
1733 }
```

### 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.

```
1734 \newcommand*{\hyxmp@embed@packet@pdfmark}{%
1735   \pdfmark{%
1736     pdfmark=/NamespacePush
1737   }%
1738   \pdfmark{%
1739     pdfmark=/OBJ,
1740     Raw={/_objdef \string\hyxmp@Metadata\string} /type /stream}%

```

```

1741 }%
1742 \pdfmark{%
1743   pdfmark=/PUT,
1744   Raw={\string{hyxmp@Metadata\string}
1745     2 dict begin
1746       /Type /Metadata def
1747       /Subtype /XML def
1748       currentdict
1749     end
1750 }%
1751 }%
1752 \pdfmark{%
1753   pdfmark=/PUT,
1754   Raw={\string{hyxmp@Metadata\string} (\hyxmp@xml)}}%
1755 }%
1756 \pdfmark{%
1757   pdfmark=/Metadata,
1758   Raw={\string{Catalog\string} \string{hyxmp@Metadata\string}}%
1759 }%
1760 \pdfmark{%
1761   pdfmark=/NamespacePop
1762 }%
1763 }

```

### 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.

```

1764 \newcommand*{\hyxmp@embed@packet@dvipdfm}{%
1765   \hyxmp@string@len{\hyxmp@xml}%
1766   \special{pdf: object @hyxmp@Metadata
1767     <<
1768       /Type /Metadata
1769       /Subtype /XML
1770       /Length \the\@tempcnta
1771     >>
1772     stream~J\hyxmp@xml endstream%
1773 }%
1774 \special{pdf: docview
1775   <<
1776     /Metadata @hyxmp@Metadata
1777   >>
1778 }%
1779 }

```

`\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.

```
1780 \newcommand*{\hyxmp@string@len}[1]{%
1781   \@tempcnta=0
1782   \expandafter\hyxmp@count@spaces#1 {} %
1783   \expandafter\hyxmp@count@non@spaces#1{}%
1784 }
```

`\hyxmp@count@spaces` Count the number of spaces in a given string. We rely on the built-in pattern matching of T<sub>E</sub>X’s `\def` primitive to pry one word at a time off the head of the input string.

```
1785 \def\hyxmp@count@spaces#1 {%
1786   \def\hyxmp@one@token{#1}%
1787   \ifx\hyxmp@one@token\@empty
1788     \advance\@tempcnta by -1
1789   \else
1790     \advance\@tempcnta by 1
1791     \expandafter\hyxmp@count@spaces
1792   \fi
1793 }
```

`\hyxmp@count@non@spaces` Count the number of non-spaces in a given string. Ideally, we’d count both spaces and non-spaces but T<sub>E</sub>X 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.

```
1794 \newcommand*{\hyxmp@count@non@spaces}[1]{%
1795   \def\hyxmp@one@token{#1}%
1796   \ifx\hyxmp@one@token\@empty
1797   \else
1798     \advance\@tempcnta by 1
1799     \expandafter\hyxmp@count@non@spaces
1800   \fi
1801 }
```

### 3.6.5 Embedding using X<sub>q</sub>T<sub>E</sub>X

`\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.

```
1802 \newcommand*{\hyxmp@embed@packet@xetex}{%
1803   \special{pdf:stream @hyxmp@Metadata (\hyxmp@xml)
1804     <<
1805       /Type /Metadata
1806       /Subtype /XML
1807     >>
1808   }%
1809   \special{pdf:put @catalog
1810     <<
1811     /Metadata @hyxmp@Metadata
```

```

1812     >>
1813   }%
1814 }

```

### 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.

```

1815 \catcode'\="=\hyxmp@dq@code

```

## 4 Help Wanted

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

## A Sample XMP Packet

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

```

<?xpacket begin="\357\273\277" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/">
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about=""
      xmlns:pdf="http://ns.adobe.com/pdf/1.3/"
      xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/"
      xmlns:xmp="http://ns.adobe.com/xap/1.0/"
      xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"
      xmlns:stEvt="http://ns.adobe.com/xap/1.0/sType/ResourceEvent#"
      xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/"
      xmlns:pdfuaid="http://www.aiim.org/pdfua/ns/id/"
      xmlns:pdfx="http://ns.adobe.com/pdfx/1.3/"
      xmlns:pdfxid="http://www.npes.org/pdfx/ns/id/"
      xmlns:prism="http://prismstandard.org/namespaces/basic/2.1/"
      xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"
      xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"
      xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema#"

```

```

xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property#"
xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type#"
xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field#">
<pdfaExtension:schemas>
  <rdf:Bag>

      ⋮
      [over 200 lines of boilerplate definitions not shown]
      ⋮

  </rdf:Bag>
</pdfaExtension:schemas>
<pdf:Keywords>
  energy quanta, Hertz effect, quantum physics
</pdf:Keywords>
<pdf:Producer>
  pdfTeX, Version 3.14159265-2.6-1.40.20 (TeX Live 2019/Debian)
</pdf:Producer>
<pdf:PDFVersion>1.5</pdf:PDFVersion>
<xmpRights:Marked>True</xmpRights:Marked>
<xmpRights:WebStatement>
  http://creativecommons.org/licenses/by-nc-nd/3.0/
</xmpRights:WebStatement>
<dc:format>application/pdf</dc:format>
<dc:title>
  <rdf:Alt>
    <rdf:li xml:lang="en">
      On a heuristic viewpoint concerning the production
      and transformation of light
    </rdf:li>
    <rdf:li xml:lang="x-default">
      On a heuristic viewpoint concerning the production
      and transformation of light
    </rdf:li>
    <rdf:li xml:lang="de">
      Über einen die Erzeugung und Verwandlung des Lichtes
      betreffenden heuristischen Gesichtspunkt
    </rdf:li>
  </rdf:Alt>
</dc:title>
<dc:description>
  <rdf:Alt>
    <rdf:li xml:lang="en">photoelectric effect</rdf:li>
    <rdf:li xml:lang="x-default">photoelectric effect</rdf:li>
  </rdf:Alt>

```

```

</dc:description>
<dc:rights>
  <rdf:Alt>
    <rdf:li xml:lang="en">
      Copyright (C) 1905, Albert Einstein
    </rdf:li>
    <rdf:li xml:lang="x-default">
      Copyright (C) 1905, Albert Einstein
    </rdf:li>
  </rdf:Alt>
</dc:rights>
<dc:publisher>
  <rdf:Bag>
    <rdf:li>Wiley-VCH</rdf:li>
  </rdf:Bag>
</dc:publisher>
<dc:creator>
  <rdf:Seq>
    <rdf:li>Albert Einstein</rdf:li>
  </rdf:Seq>
</dc:creator>
<dc:subject>
  <rdf:Bag>
    <rdf:li>energy quanta</rdf:li>
    <rdf:li>Hertz effect</rdf:li>
    <rdf:li>quantum physics</rdf:li>
  </rdf:Bag>
</dc:subject>
<dc:date>
  <rdf:Seq>
    <rdf:li>1905-03-17</rdf:li>
  </rdf:Seq>
</dc:date>
<dc:language>
  <rdf:Bag>
    <rdf:li>en</rdf:li>
  </rdf:Bag>
</dc:language>
<dc:type>
  <rdf:Bag>
    <rdf:li>Text</rdf:li>
  </rdf:Bag>
</dc:type>
<dc:source>einstein.tex</dc:source>
<dc:identifier>info:lcnn/50013519</dc:identifier>
<photoshop:AuthorsPosition>

```

```

    Technical Assistant, Level III
  </photoshop:AuthorsPosition>
  <photoshop:CaptionWriter>Scott Pakin</photoshop:CaptionWriter>
  <xmp:CreateDate>2019-03-16T23:07:38-06:00</xmp:CreateDate>
  <xmp:ModifyDate>2019-03-16T23:07:38-06:00</xmp:ModifyDate>
  <xmp:MetadataDate>2019-03-16T23:07:38-06:00</xmp:MetadataDate>
  <xmp:CreatorTool>LaTeX with hyperref package</xmp:CreatorTool>
  <xmpMM:DocumentID>
    uuid:6d1ac9ec-4ff2-515a-954b-648eeb4853b0
  </xmpMM:DocumentID>
  <xmpMM:InstanceID>
    uuid:3e4c4182-b182-46c9-995f-754c41d13390
  </xmpMM:InstanceID>
  <xmpMM:VersionID>2.998e8</xmpMM:VersionID>
  <xmpMM:RenditionClass>default</xmpMM:RenditionClass>
  <Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">
    <Iptc4xmpCore:CiAdrExtadr>Kramgasse 49</Iptc4xmpCore:CiAdrExtadr>
    <Iptc4xmpCore:CiAdrCity>Bern</Iptc4xmpCore:CiAdrCity>
    <Iptc4xmpCore:CiAdrPcode>3011</Iptc4xmpCore:CiAdrPcode>
    <Iptc4xmpCore:CiAdrCtry>Switzerland</Iptc4xmpCore:CiAdrCtry>
    <Iptc4xmpCore:CiTelWork>031 312 00 91</Iptc4xmpCore:CiTelWork>
    <Iptc4xmpCore:CiEmailWork>aeinstein@ipi.ch</Iptc4xmpCore:CiEmailWork>
    <Iptc4xmpCore:CiUrlWork>
      http://einstein.biz/,
      https://www.facebook.com/AlbertEinstein
    </Iptc4xmpCore:CiUrlWork>
  </Iptc4xmpCore:CreatorContactInfo>
  <prism:complianceProfile>three</prism:complianceProfile>
  <prism:subtitle xml:lang="en-US">
    Putting that bum Maxwell in his place
  </prism:subtitle>
  <prism:publicationName xml:lang="de">
    Annalen der Physik
  </prism:publicationName>
  <prism:aggregationType>journal</prism:aggregationType>
  <prism:volume>322</prism:volume>
  <prism:number>6</prism:number>
  <prism:pageRange>132-148</prism:pageRange>
  <prism:issn>0003-3804</prism:issn>
  <prism:eIssn>1521-3889</prism:eIssn>
  <prism:doi>10.1002/andp.19053220607</prism:doi>
  <prism:url>
    http://www.physik.uni-augsburg.de/annalen/history/einstein-papers/190517132-
  </prism:url>
  <prism:byteCount>59846</prism:byteCount>
  <prism:pageCount>17</prism:pageCount>

```



```

        </rdf:Description>
    </rdf:RDF>
</x:xmpmeta>
<?xpacket end="w"?>

```

## References

- [1] Adobe Systems, Inc., San Jose, California. *Adobe Acrobat X SDK Help, pdfmark Reference*. Available from <http://www.adobe.com/devnet/acrobat/documentation.html>.
- [2] Adobe Systems, Inc. *PostScript Language Reference Manual*. Addison-Wesley, 2nd edition, January 1996, ISBN: 0-201-18127-4.
- [3] Adobe Systems, Inc., San Jose, California. *Document Management—Portable Document Format—Part 1: PDF 1.7*, July 2008. ISO 32000-1 standard document. Available from [http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000\\_2008.pdf](http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000_2008.pdf).
- [4] Adobe Systems, Inc., San Jose, California. *XMP Specification Part 1: Data model, Serialization, and Core Properties*, April 2012. Available from <http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/xmp/pdfs/cc-201306/XMPSpecificationPart1.pdf>.
- [5] DCMI Usage Board *DCMI Metadata Terms*, June 14, 2012. Available from <http://dublincore.org/documents/dcmi-terms/>.
- [6] Michael Downes. Around the bend #15, answers, 4th (last) installment. `comp.text.tex` newsgroup posting, January 3, 1994. Archived by Google at <http://groups.google.com/group/comp.text.tex/msg/7da7643b9e8f3b48>.
- [7] International Digital Enterprise Alliance, Inc. *Publishing Requirements for Industry Standard Metadata, Version 3.0: PRISM Basic Metadata Specification*, October 12, 2012. Available from [http://www.prismstandard.org/specifications/3.0/PRISM\\_Basic\\_Metadata\\_3.0.htm](http://www.prismstandard.org/specifications/3.0/PRISM_Basic_Metadata_3.0.htm).
- [8] International Digital Enterprise Alliance, Inc. *Publishing Requirements for Industry Standard Metadata, Version 3.0: PRISM Controlled Vocabularies Specification*, October 4, 2012. Available from [http://www.prismstandard.org/specifications/3.0/PRISM\\_CV\\_Spec\\_3.0.pdf](http://www.prismstandard.org/specifications/3.0/PRISM_CV_Spec_3.0.pdf).
- [9] International Press Telecommunications Council. *IPTC Photo Metadata: Core 1.1/Extension 1.1*, July 2010. Revision 1. Available from [http://www.iptc.org/std/photometadata/specification/IPTC-PhotoMetadata-201007\\_1.pdf](http://www.iptc.org/std/photometadata/specification/IPTC-PhotoMetadata-201007_1.pdf).

- [10] Internet Assigned Numbers Authority. Language subtag registry, January 11, 2011. Available from <http://www.iana.org/assignments/language-subtag-registry>.
- [11] Paul J. Leach, Michael Mealling, and Rich Salz. A Universally Unique IDentifier (UUID) URN namespace. Request for Comments 4122, Internet Engineering Task Force, Network Working Group, July 2005. Category: Standards Track. Available from <http://www.ietf.org/rfc/rfc4122.txt>.
- [12] PDF/A Competence Center, Berlin, Germany. *TechNote 0008: Pre-defined XMP Properties in PDF/A-1*, March 20, 2008. Available from [http://www.pdfa.org/wp-content/uploads/2011/08/tn0008\\_predefined\\_xmp\\_properties\\_in\\_pdfa-1\\_2008-03-20.pdf](http://www.pdfa.org/wp-content/uploads/2011/08/tn0008_predefined_xmp_properties_in_pdfa-1_2008-03-20.pdf).
- [13] PDF/A Competence Center, Berlin, Germany. *TechNote 0009: XMP Extension Schemas in PDF/A-1*, March 20, 2008. Available from [http://www.pdfa.org/wp-content/uploads/2011/08/tn0009\\_xmp\\_extension\\_schemas\\_in\\_pdfa-1\\_2008-03-20.pdf](http://www.pdfa.org/wp-content/uploads/2011/08/tn0009_xmp_extension_schemas_in_pdfa-1_2008-03-20.pdf).
- [14] Misha Wolf and Charles Wicksteed. Date and time formats. Note NOTE-datetime, World Wide Web Consortium (W3C), September 15, 1997. Available from <http://www.w3.org/TR/NOTE-datetime>.

## Change History

v1.0	General: Initial version . . . . .	1	which enables an author to specify the language in which he wrote the document's metadata	30
v1.1	\hyxmp@construct@packet: Explicitly set the category codes of characters $\langle EF \rangle$ , $\langle BB \rangle$ , and $\langle BF \rangle$ to "letter". Thanks to Daniel Schömer for the bug report . . . . .	71	v1.4 \hyxmp@mm@schema: Renamed the xapMM namespace prefix to xmpMM . . . . .	58
v1.2	General: Added support for the X <sub>Y</sub> L <sub>A</sub> T <sub>E</sub> X backend (xdvipdfmx) . .	1	\hyxmp@rdf@dc: Included metadata in the x-default language regardless of the specified metadata language . . . . .	55
	Added support for the Photoshop schema . . . . .	1	\hyxmp@xmpRights@schema: Renamed the xapRights namespace prefix to xmpRights	58
	Made the package compatible with ngerman. Thanks to Tobias Mueller for the bug report. . . . .	17	v1.5 General: Made the XMP inclusion more robust. Thanks to Heiko Oberdiek for the bug report and suggested modifications. .	17
v1.3	General: Introduced the pdfmetalang package option,		v2.0 \ProcessKeyvalOptions: Added	

this macro .....	27	\hyxmp@xmp@basic@schema: Added	
\XMPTruncateList: Added this		this macro .....	59
macro .....	33	\hyxmp@xmpRights@schema:	
\hyxmp@ProcessKeyvalOptions:		Modified to include	
Added this macro .....	27	xmpRights:Marked only when	
\hyxmp@SpaceOther: Added by		pdfcopyright is specified and	
Heiko Oberdiek .....	43	xmpRights:WebStatement only	
\hyxmp@add@simple: Added this		when pdflicenseurl is specified .	58
macro .....	45	\hyxmp@zero: Added by Heiko	
\hyxmp@add@to@xml: Updated also		Oberdiek .....	45
to replace commas .....	51	\ifhyxmp@unicodetex: Added by	
\hyxmp@bom: Added by Heiko		Heiko Oberdiek .....	39
Oberdiek .....	71	\xmpcomma: Added this macro ...	32
\hyxmp@comma: Added this macro	32	\xmpquote: Added this macro ...	33
\hyxmp@construct@packet:		General: Added support for the	
Modified by Heiko Oberdiek to		XMP Basic schema and	
use an appropriate BOM		miscellaneous other bits of	
representation via \hyxmp@bom	71	metadata .....	1
\hyxmp@crap@convert: Added by		Heiko Oberdiek's major rewrite	
Heiko Oberdiek .....	44	of the code to better support	
\hyxmp@crap@test: Added by		native-Unicode T <sub>E</sub> X	
Heiko Oberdiek .....	44	implementations (X <sub>Y</sub> T <sub>E</sub> X and	
\hyxmp@dc@schema: Added support		LuaT <sub>E</sub> X) .....	1
for dc:language and dc:source .	57	New \AtBeginDocument code	
\hyxmp@is@unicode: Added by		from Heiko Oberdiek to	
Heiko Oberdiek .....	41	properly encode	
\hyxmp@list@to@xml: Modified by		\@pdfmetalang .....	30
Heiko Oberdiek to use the new			
Unicode-processing macros ..	56	v2.1	
\hyxmp@photoshop@schema:		\hypersetup: Added this macro .	28
Simplified using		\hyxmp@hypersetup: Added this	
\hyxmp@add@simple .....	59	macro .....	28
\hyxmp@skiptorelax: Added by		\hyxmp@redefine@Hyp: Added this	
Heiko Oberdiek .....	44	macro .....	26
\hyxmp@skipzeros: Added by		General: Enabled hyperxmp and	
Heiko Oberdiek .....	43	hyperref to be loaded in either	
\hyxmp@toxml: Added by Heiko		order. This addresses a bug	
Oberdiek .....	41	report by Yury Donskoy ....	26
Escaped parentheses written		v2.2	
with pdfmarks to prevent dvips		\hyxmp@iptc@extensions: Added	
from line-wrapping the XMP		this macro to support PDF/A	
packet .....	42	generation .....	67
\hyxmp@toxml@unicodetex: Added		\hyxmp@iptc@schema: Added this	
by Heiko Oberdiek .....	42	macro .....	61
\hyxmp@xetex@crap: Added by		\hyxmp@list@to@lines: Added	
Heiko Oberdiek .....	43	this macro .....	61
\hyxmp@xmlify: Completely		\xmpcomma: Changed the default	
rewritten by Heiko Oberdiek to		from \relax to an ordinary	
better support Unicode-enabled		comma .....	32
T <sub>E</sub> X programs .....	40	\xmplinesep: Added this macro ..	61

General: Added support for the IPTC Photo Metadata schema . . .	1	\hyxmp@today@xmp: Modified the code to parse the time and timezone from \pdfcreationdate, as proposed by Florian Breitwieser . . . . .	38
v2.3		\hyxmp@iptc@extensions: Gave the lptc4xmpCore:CreatorContactInfo fields a unique pdfaType:prefix to better support conversion of the document to PDF/A . . . . .	67
v2.3a		General: Bug fix: Redefine \@pdflang as \@empty when hyperref has set it to \relax . . .	30
v2.3b		\XMPTruncateList: Made all definitions local to avoid spurious Too many unprocessed floats errors when running with memoir . . .	33
v2.4		\hyxmp@add@simple@var: Added this macro . . . . .	46
		\hyxmp@create@uuid: Modified this macro to produce a proper version 4 (random or pseudorandom) UUID . . . . .	50
		\hyxmp@dc@schema: Made dc:language a Bag instead of an individual item so as to conform to the latest XMP specifications, a detail identified by Florian Breitwieser . . . . .	57
		\hyxmp@parse@time: Added this macro . . . . .	34
		\hyxmp@parse@tz: Added this macro . . . . .	35
		\hyxmp@parse@tz@char: Added this macro . . . . .	34
		\hyxmp@pdf@schema: Made \hyxmp@pdf@schema <i>always</i> include the Adobe PDF schema, even when empty. Florian Breitwieser noted that this is necessary for PDF/A-1b compliance . . . . .	54
		\hyxmp@pdf@to@xmp@date: Added this macro . . . . .	34
		\hyxmp@pdfa@id@schema: Added this macro . . . . .	60
	v2.5	\hyxmp@add@to+xml: Updated also to replace underscores and to modify only the text being added, not the already-modified text . . . . .	51
		\hyxmp@textunderscore: Added this macro . . . . .	18
		\hyxmp@uscore: Added this macro . . .	33
		General: Enabled “\_” to work within email addresses, as requested by Leonid Sinev . . . . .	1
	v2.6	General: Added support for a new pdfdate key to explicitly specify the document date (and optionally time) . . . . .	1
	v2.7	General: Automatically use \title and \author if pdftitle and pdfauthor are left unspecified. Thanks to Maciej Radziejewski for the suggestion . . . . .	30
	v2.8	\hyxmp@add@to+xml: Corrected inadvertent lowercasing of non-Latin characters when run under Xe <sup>L</sup> A <sup>T</sup> E <sub>X</sub> or Lua <sup>L</sup> A <sup>T</sup> E <sub>X</sub> (bug reported by Leonid Sinev) . . .	51
	v2.9	\hyxmp@iptc@schema: Use lptc4xmpCore instead of lptc4ContInfo as the contact-information metadata prefix. Leonid Sinev reports	

that Acrobat's PDF/A validator seems to prefer <code>lptc4xmpCore</code> .	61	<code>\hyxmp@today@xmp@define:</code> Modified to include hours and minutes . . . . .	37
<code>\hyxmp@pdfa@id@schema:</code> Let the author specify the PDF/A part and conformance IDs, as requested by Leonid Sinev . . .	60	<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 . . . .	59
General: Force inclusion of <code>dc:creator</code> , <code>dc:title</code> , and <code>dc:description</code> —even if empty—when <code>hyperref</code> is loaded with the <code>pdfa</code> option (suggested by Leonid Sinev) . . . . .	1	v3.3 <code>\@pdfsource:</code> Added this macro and the corresponding <code>pdfsource</code> option, at Niklas Beisert's request . . . . .	22
Introduced the <code>pdftype</code> package option, which enables an author to specify the type of document being produced . . . . .	1	<code>\XMPLangAlt:</code> Added this macro based on a request—and some code—by Niklas Beisert to support metadata expressed in multiple languages . . . . .	48
v3.0 <code>\hyxmp@embed@packet@luatex:</code> Added this macro . . . . .	74	<code>\hyxmp@rdf@dc:</code> Bug fix: Output the metadata language as correct XML even when <code>hyperref</code> is loaded with the <code>unicode</code> option . . . . .	55
<code>\hyxmp@today@xmp@define:</code> Modified to accept the name of a macro to define . . . . .	37	General: Don't overwrite an existing <code>pdfmetalang</code> with <code>pdflang</code> or <code>x-default</code> . This addresses a bug report by Niklas Beisert . . . . .	30
<code>\hyxmp@xmp@basic@schema:</code> Made the XMP <code>xmp:CreateDate</code> , <code>xmp:ModifyDate</code> , and <code>xmp:MetadataDate</code> match the PDF <code>CreationDate</code> . . . . .	59	v3.4 <code>\hyxmp@seed@string:</code> Correctly handle an author field of all spaces. Bug reported by Gaëtan Leurent . . . . .	50
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	General: Use <code>ifmtarg</code> to test for empty arguments, including non-empty but all spaces . . . .	1
v3.1 <code>\hyxmp@embed@packet@luatex:</code> Updated to use <code>\pdfextension</code> <code>obj uncompressed</code> as suggested by Hans Hagen . . . . .	74	v3.5 <code>\hyxmp@DocumentID:</code> Added the <code>pdfdocumentid</code> option, at Michael Osipov's request . . . .	22
<code>\hyxmp@embed@packet@pdftex:</code> Leave the XMP packet—and only the XMP packet—uncompressed in both <code>pdfTeX</code> and pre-0.85 LuaTeX . . . .	74	<code>\hyxmp@InstanceID:</code> Added the <code>pdfinstanceid</code> option, at Michael Osipov's request . . . . .	22
v3.2 <code>\hyxmp@as@pdf@date:</code> Added this macro . . . . .	35	<code>\hyxmp@mm@schema:</code> Generate <code>\hyxmp@DocumentID</code> and <code>\hyxmp@InstanceID</code> only if the document does not already define these using the	
<code>\hyxmp@as@xmp@date:</code> Added this macro . . . . .	34		

pdfdocumentid and pdfinstanceid options . . . . .	58	\hyxmp@prism@schema: Added this macro . . . . .	62
\hyxmp@seed@string: Seed with the T <sub>E</sub> X timestamp in addition to the document-specified timestamp . . . . .	51	General: Include all metadata within a single rdf:Description block . . . . .	1
v4.0		v4.1	
\XMPTruncateList: Deprecated this macro . . . . .	33	\hyxmp@singleton@dc: Added this macro . . . . .	56
\hyxmp@add@simple@lang: Added this macro . . . . .	46	General: Invoke \hyxmp@no@info@lists at the beginning of the document, for compatibility with both newer and older versions of hyperref .	31
\hyxmp@begin@ext@decl: Added this macro . . . . .	64	Updated the documentation to refer to \pdfnumpages by its correct name. Thanks to Volker RW Schaa for catching the discrepancy . . . . .	1
\hyxmp@declare@field: Replaced \hyxmp@declare@resource with this macro . . . . .	65		
\hyxmp@declare@property: Added this macro . . . . .	65	v5.0	
\hyxmp@end@ext@decl: Added this macro . . . . .	64	\@pdfrendition: Added the pdfrendition option . . . . .	22
\hyxmp@iptc@extensions: Moved the header code from here into \hyxmp@begin@extension@decls and the trailer code from here into \hyxmp@end@extension@decls	67	\@pdfxstandard: Added this macro	21
Rewrote to more closely honor the XMP specification . . . . .	67	\hyxmp@add@simple: Insert the tag name (#1) verbatim . . . . .	45
\hyxmp@iptc@schema: Moved the definition of \hyxmp@iptc@data from here into \hyxmp@check@iptc@data . . .	61	\hyxmp@check@standards: Added this macro . . . . .	29
Renamed this macro to \hyxmp@iptc@schema from \hyxmp@photometa@schema . .	61	\hyxmp@check@std: Added this macro . . . . .	21
Rewrote this macro entirely to correct the use of fields within a structure . . . . .	61	\hyxmp@declare@property: Insert the property name (#2) verbatim . . . . .	65
\hyxmp@mm@extensions: Added this macro . . . . .	65	\hyxmp@define@pdfproducer: Added this macro . . . . .	53
\hyxmp@mm@schema: Include xmpMM:VersionID in the XMP packet . . . . .	58	\hyxmp@no@info@lists: Renamed this macros from \hyxmp@suppress@pdf@metadata and rewrote it to replace, if possible, only Author and Keywords . . . . .	25
\hyxmp@no@info@lists: Added this macro . . . . .	25	\hyxmp@pdf@extensions: Added this macro . . . . .	65
\hyxmp@pdfa@id@extensions: Added this macro . . . . .	66	\hyxmp@pdf@schema: Honor pdftrapped . . . . .	54
\hyxmp@prism@extensions: Added this macro . . . . .	69	\hyxmp@pdfua@id@extensions: Added this macro . . . . .	66
		\hyxmp@pdfua@id@schema: Added this macro . . . . .	60
		\hyxmp@pdfx@id@extensions: Added this macro . . . . .	67

<code>\hyxmp@pdfx@id@schema</code> : Added this macro . . . . .	60	Robin Schwab for the bug report . . . . .	53
<code>\hyxmp@today@pdf</code> : Added this macro . . . . .	38	<code>\hyxmp@timestamp</code> : Don't rely on <code>\jobname.aux</code> existing to query the current time under $\text{\TeX}$ . Instead, use <code>\jobname.log</code> . Thanks to Ulrike Fischer for the bug report and for her suggestion to use the log file. . .	38
<code>\hyxmp@today@xmp</code> : Support $\text{\TeX}$ 's <code>\filemoddate</code> . . . .	38		
<code>\hyxmp@today@xmp@define</code> : Modified to specify UTC . . . .	37		
General: Added support for PDF/UA standards, as requested by Robin Schwab . . . . .	1	v5.2	
Added support for PDF/X standards, as requested by Robin Schwab . . . . .	1	<code>\hyxmp@add@simple@pfx</code> : Added this macro . . . . .	47
Define a default producer . . . .	53	<code>\hyxmp@assign@major@minor</code> : Added this macro. <code>hyperxmp</code> now correctly specifies <code>pdf:PDFVersion</code> when generating PDF 2.0+. Thanks to Ulrike Fischer for alerting me to PDF 2.0's availability in the $\text{\TeX}$ ecosystem and informing me how to activate it .	53
Don't set any document dates (creation, modification, or metadata) from <code>pdfdate</code> . . . .	30	<code>\hyxmp@cond@dc@identifier</code> : Added this macro . . . . .	57
v5.1		<code>\ifdraft</code> : Define <code>\ifdraft</code> only locally, at Niklas Beisert's request . . . . .	22
<code>\hyxmp@banner@to@producer</code> : Prevent the category code of "@" from propagating past the <code>\begin{document}</code> . Thanks to Robert Schlicht for noticing this catcode "leak" and providing a correction . . . . .	53	General: Introduced the <code>pdfidentifier</code> package option, which enables an author to specify a unique identifier for the document . . . . .	1
<code>\hyxmp@define@pdfproducer</code> : Check for $\text{\LaTeX}$ before checking for $\text{\pdfTeX}$ to work around <code>luatex85</code> 's confusing <code>iftex</code> by defining <code>\pdftexversion</code> . Thanks to			

## Index

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

<b>Symbols</b>	<code>\@ifmtarg</code> . . . .	18, 817	<code>\@ifnotmtarg</code> . . .	19,
<code>\#</code> . . . . .	<code>\@ifmtargexp</code> . . . .		61, 806, 815, 1103	
<code>\&amp;</code> . . . . .	. . .	<u>18</u> , 313, 330,	<code>\@ifnotmtargexp</code> <u>18</u> ,	
<code>\@author</code> . . . .	352, 358, 365,		329, 353, 359,	
<code>\@baseurl</code> . . .	553, 1042, 1077,		788, 829, 1116, 1230	
<code>\@elt</code> <u>414</u> , <u>1087</u> , <u>1237</u> , <u>1242</u>	1136, 1166, 1167,	<code>\@pdfaconformance</code> . .		
<code>\@elt@first</code> . . . . .	<u>1235</u>	1174, 1180, 1186	. . . .	<u>70</u> , 320, 1202
<code>\@elt@rest</code> . .	1237, <u>1239</u>	<code>\@ifnextchar</code> . . . . .	812	<code>\@pdfapart</code> . . . .
				<u>65</u> ,



313, 319, 325, 1201	\@pdflang . 287, 341,	\~ . . . . . 411, 722, 723
\@pdfauthor 194, <u>218</u> ,	342, 345, 348, 1128	
264, 358, 941, 949	\@pdflicenseurl . . .	
\@pdfauthortitle . .	<u>52</u> , 288, 1148, 1162	\_ . . . . . 688, 723, 968
<u>54</u> , 265, 1195, 1196	\@pdfmetadatetitle . .	
\@pdfbookedition . .	37, 289, 1186, 1189	<b>A</b>
<u>143</u> , 266, 1281, 1307	\@pdfmetalang . . . . .	\and . . . . . <u>218</u>
\@pdfbytes . . . . .	. . <u>58</u> , 344, 346,	ASCII . . . . . 18, 41
<u>133</u> , 267, 1290, 1308	348, 351, 1053, 1055	\AtBeginDocument . .
\@pdfcaptionwriter .	\@pdfmoddate . . . . .	. . . . . 339, 1014
<u>56</u> , 268, 1195, 1197	. . . 290, 1180, 1184	\AtEndDocument . . . . . 5
\@pdfcontactaddress	\@pdfnumpages . . . . .	\AtEndDvi . . . . . 8
<u>161</u> , 269, 1258, 1295	<u>135</u> , 291, 1291, 1314	atenddvi . . . . . 17
\@pdfcontactcity . .	\@pdfpagerange . . . . .	Author . . . . . 11, 25, 86
<u>169</u> , 270, 1259, 1296	<u>151</u> , 292, 1284, 1315	<b>B</b>
\@pdfcontactcountry	\@pdfproducer . . <u>992</u> ,	Bag . . . . . 84
<u>175</u> , 271, 1262, 1299	<u>1006</u> , <u>1013</u> , 1015	baseurl (option) . . . . .
\@pdfcontactemail . .	\@pdfpublication . . .	4, 6, 14, 18, 24, 59
<u>179</u> , 272, 1265, 1301	<u>129</u> , 293, 1279, 1316	BOM . . . . . 71, 83
\@pdfcontactphone . .	\@pdfpublisher <u>145</u> , 1126	
<u>177</u> , 273, 1264, 1300	\@pdfpubtype . . . . .	<b>C</b>
\@pdfcontactpostcode	<u>131</u> , 294, 1280, 1317	CiAdrCity . . . . . 2
<u>173</u> , 274, 1261, 1298	\@pdfrendition <u>122</u> , 1171	CiAdrCtry . . . . . 2
\@pdfcontactregion .	\@pdfsource . . . . .	CiAdrExtadr . . . . . 2
<u>171</u> , 275, 1260, 1297	. . . <u>111</u> , 1132, 1134	CiAdrPcode . . . . . 2
\@pdfcontacturl . . .	\@pdfsubject . 295, 1124	CiAdrRegion . . . . . 2
<u>181</u> , 276, 1266, 1302	\@pdfsubtitle . . . . .	CiEmailWork . . . . . 2
\@pdfcopyright . . . .	<u>159</u> , 296, 1278, 1318	CiTelWork . . . . . 3
. . . . . 48, 277,	\@pdftitle . 297, 330,	CiUrlWork . . . . . 3
1125, 1152, 1158	352, 941, 949, 1123	CreationDate . . . . . 31, 85
\@pdfcreationdate . .	\@pdftrapped . . . . . 1037	
. . . . . 278,	\@pdftype . . . . . <u>50</u> , 1129	<b>D</b>
365, 366, 1174, 1178	\@pdfuapart . . . . . <u>74</u> ,	Date . . . . . 38
\@pdfcreator . . . . . 1191	298, 327, 1206, 1602	\day . . . . . 531, 532, 534
\@pdfdatetime . . <u>26</u> , 279	\@pdfurl . . . . .	dc:creator . . 2, 12, 57, 85
\@pdfdoi . . <u>153</u> , 280,	<u>155</u> , 299, 1289, 1319	dc:date . . . . . 2, 57
1138, 1288, 1309	\@pdfversionid <u>117</u> , 1170	dc:description . . . . .
\@pdfeissn . <u>139</u> , 281,	\@pdfvolumenum . . . .	. . 3, 12, 47, 57, 85
1139, 1287, 1310	<u>147</u> , 300, 1282, 1320	dc:format . . . . . 2
\@pdfidentifier . . .	\@pdfxstandard . . . .	dc:identifier . . . . . 3, 57
<u>157</u> , 282, 1136, 1143	. . <u>92</u> , 301, 326,	dc:language . 3, 57, 83, 84
\@pdfisbn . <u>141</u> , 283,	1214, 1217, 1219	dc:publisher . . . . . 3
1141, 1285, 1311	\@pdfxversion . . . . 1606	dc:rights . . . 2, 16, 47, 57
\@pdfissn . <u>137</u> , 284,	\@tempwafalse 1042, 1077	dc:source . . . . . 2, 57, 83
1140, 1286, 1312	\@tempwattrue . 1042,	dc:subject . . . . . 2, 57
\@pdfissuenum . . . . .	1044, 1077, 1079	dc:title . . 3, 12, 47, 57, 85
<u>149</u> , 285, 1283, 1313	\@title . . . . . 353, 354	dc:type . . . . . 3, 57
\@pdfkeywords . . . . .	\~ . . . . 402, 406, 589,	DCMI . . . . . 8
. . . . . 200, <u>239</u> , 286	969, 970, 1627–1629	\define@key . . . . 27,
	\_ . . . . . 968, 970	38, 49, 51, 53, 55,



57, 59, 66, 71, 75, 94, 112, 114, 116, 118, 128, 130, 132, 134, 136, 138, 140, 142, 144, 146, 148, 150, 152, 154, 156, 158, 160, 162, 170, 172, 174, 176, 178, 180, 182, 206, 218, 239, 844		hyperxmp . . . . . 1, 2, 4–9, 12–18, 20, 21, 24, 26, 30–32, 34, 40, 48, 52, 54, 63, 73, 77, 83, 85, 87		\hyxmp@append@hex . . . 889, 908–910, 914
\do . . . . . 846, 853, 1063		\hyxmp@@is@unicode . 632		\hyxmp@append@hex@iii . 907, 913, 923, 934
DOI . . . . . 2, 6, 23		\hyxmp@add@simple . . . . . . . 787, 1037, 1122, 1134, 1143, 1160, 1162, 1168– 1171, 1175, 1177, 1181, 1183, 1187, 1189, 1191, 1192, 1196, 1197, 1201, 1202, 1206, 1213, 1214, 1217, 1219, 1259–1262, 1276, 1280, 1282–1291		\hyxmp@append@hex@iv . . . . . 912, 918, 919, 921, 936–938
draft (option) . . . . . 8		\hyxmp@add@simple@lang . . . . . 805, 1278, 1279, 1281		\hyxmp@as@pdf@date . 458
dvipdf (option) . . . . . 74		\hyxmp@add@simple@lang@i . . . . . 808, 811		\hyxmp@as@xmp@date . . . . . . 32, 43, 430, 568, 1178, 1184
dvipdfm . . . . . 75		\hyxmp@add@simple@lang@ii . . . . . 812, 814		\hyxmp@assign@major@minor . . . . . 1019, 1038
dvips (option) . . . . . 74		\hyxmp@add@simple@pfx . . . . . 828, 1117		\hyxmp@at@end . . . 3, 372
dvips . . . . . 10, 42, 83		\hyxmp@add@simple@var . . . . . 796, 1035, 1036, 1039		\hyxmp@banner@to@producer . . . . . 995, 998, 1006
dvipsone (option) . . . . 74		\hyxmp@add@to+xml . . . . . . . 790, 792, 800, 818, 822, 830, 834, 836, 955, 981, 1049, 1056, 1060, 1065, 1070, 1082, 1090, 1096, 1105, 1232, 1236, 1240, 1246, 1253, 1268, 1324, 1330, 1336, 1346, 1353, 1357, 1365, 1477, 1516, 1635, 1674		\hyxmp@begin@ext@decl . . . . . 1335, 1374, 1385, 1410, 1426, 1440, 1456, 1468, 1526
dviwindo (option) . . . 74		\hyxmp@add@to+xml@i . . . . . 790, 792, 800, 818, 822, 830, 834, 836, 955, 981, 1049, 1056, 1060, 1065, 1070, 1082, 1090, 1096, 1105, 1232, 1236, 1240, 1246, 1253, 1268, 1324, 1330, 1336, 1346, 1353, 1357, 1365, 1477, 1516, 1635, 1674		\hyxmp@begin@extension@decls . . . . . 1323, 1596
<b>E</b>		\hyxmp@alt@description . . . . . 840, 850		\hyxmp@big@prime . . . 863, 866, 876, 886
$\varepsilon$ -TeX . . . . . 53		\hyxmp@alt@rights . . . . . . . 840, 851		\hyxmp@big@prime@ii . . . . . 863, 885
\EdefEscapeHex 611, 624		\hyxmp@alt@title 840, 849		\hyxmp@bom . . 1620, 1635
\EdefUnescapeHex . . 628		\hyxmp@and . . . . . 218		\hyxmp@check@iptc@data . . . . . 1293, 1660
\EdefUnescapeString 598				\hyxmp@check@prism@data . . . . . 1305, 1661
\empty . . . . . 1438				\hyxmp@check@standards . . . . . 311, 370
\equal . . . . . 88				\hyxmp@check@std . . . . . . . 87, 99–107
etoolbox . . . . . 18				\hyxmp@comma . . . . . . 163, 219, 240, 401
ETX . . . . . 32, 40				\hyxmp@commas@to@list 385, 421, 1088, 1244
<b>F</b>				\hyxmp@commas@to@list@i . . . . . 387, 389
\filemoddate . . . . . 566				\hyxmp@concat@metadata . . . . . 261
<b>G</b>				\hyxmp@cond@dc@identifier . . 1114, 1138–1141
Ghostscript . . . . . 10, 11				\hyxmp@construct@packet . . . . . 1633, 1683
gitver . . . . . 6				\hyxmp@count@non@spaces . . . . . 1783, 1794
<b>H</b>				
\Hy@driver . . . . . . . 4, 1685, 1689, 1693, 1697, 1702				
\Hy@unicodefalse 29, 40				
hyperref . . . . . 1, 4–9, 12, 15, 18, 20, 25, 26, 28–32, 52–54, 73, 74, 83–86				
\hypersetup 256, 354, 360				

\hyxmp@count@spaces	\hyxmp@embed@packet@xetex	\hyxmp@modulo@a	857,
..... 1782, <u>1785</u>	..... 1698, <u>1802</u>		876, 886, 892, 927
\hyxmp@crap@convert	\hyxmp@end@ext@decl	\hyxmp@new@xml	971, 972
..... 714, <u>748</u>	... 1345, 1382,	\hyxmp@no@bad@parts	
\hyxmp@crap@result	1407, 1423, 1435,	..... 60, 67, 76	
..... <u>704</u> , 740	1452, 1464, 1593	\hyxmp@no@info@lists	
\hyxmp@crap@test	711, <u>736</u>	.... <u>183</u> , 207, 371	
\hyxmp@create@uuid	..... <u>1329</u> , 1618	\hyxmp@num	..... <u>748</u>
.... <u>916</u> , 944, 953	\hyxmp@extra@indent	\hyxmp@one@token	..
\hyxmp@cur@lang	846, <u>854</u>		865, <u>869</u> , 1786,
\hyxmp@dc@schema	..		1787, 1795, 1796
..... <u>1121</u> , 1665	\hyxmp@find@metadata	\hyxmp@padding	<u>979</u> , 1678
\hyxmp@declare@extensions	..... <u>261</u> , 373	\hyxmp@parse@time	..
..... <u>1595</u> , 1662	\hyxmp@first@char	..... 439, <u>441</u>	
\hyxmp@declare@field	\hyxmp@first@char@i	\hyxmp@parse@tz	...
..... <u>1364</u> ,	.... <u>428</u> , 431, 459	.... 448, 451, <u>455</u>	
1492, 1495, 1498,	\hyxmp@gobbletwo	\hyxmp@parse@tz@char	
1501, 1504,	<u>498</u> , 511	..... 443, <u>445</u>	
1507, 1510, 1513	\hyxmp@hash	\hyxmp@pdf@extensions	
\hyxmp@declare@property	<u>975</u> , 1639,	..... <u>1373</u> , 1597	
..... <u>1352</u> ,	1647, 1655–1658	\hyxmp@pdf@schema	..
1378, 1389, 1394,	\hyxmp@Hyp@pdfauthor	..... <u>1034</u> , 1663	
1399, 1403, 1414,	..... <u>212</u>	\hyxmp@pdf@to@xmp@date	
1419, 1430, 1444,	\hyxmp@Hyp@pdfkeywords	.. 432, <u>437</u> , 560, 563	
1448, 1460, 1472,	..... <u>233</u>	\hyxmp@pdfa@id@extensions	
1530, 1534, 1538,	\hyxmp@hypersetup	..... <u>1409</u> , 1600	
1542, 1546, 1550,	.. <u>256</u>	\hyxmp@pdfa@id@schema	
1554, 1558, 1562,	\hyxmp@InstanceID	..... <u>1199</u> , 1668	
1566, 1570, 1574,	..	\hyxmp@pdfauthor	..
1579, 1584, 1589	<u>115</u> , <u>946</u> , 1167, 1169	... <u>209</u> , <u>218</u> , 1130	
\hyxmp@def@DocumentID	\hyxmp@iprefix	\hyxmp@pdfkeywords	..
..... <u>940</u> , 1166	832, 833	... <u>209</u> , <u>239</u> , 1131	
\hyxmp@def@InstanceID	\hyxmp@iptc@data	\hyxmp@pdfstringdef	
..... <u>946</u> , 1167	.. 1251, <u>1294</u> , 1610	..... <u>20</u> , 31,	
\hyxmp@define@pdfproducer	\hyxmp@iptc@extensions	42, 49, 51, 53, 55,	
..... <u>992</u> , 1016	..... <u>1467</u> , 1612	57, 59, 68, 72, 77,	
\hyxmp@DocumentID	\hyxmp@iptc@schema	95, 112, 114, 116,	
.. <u>113</u> , <u>940</u> , 1166, 1168	..	118, 128, 130,	
\hyxmp@dq@code	.... 600, 617, <u>632</u>	132, 134, 136,	
.. <u>1</u> , 1815	\hyxmp@LA@accept	138, 140, 142,	
\hyxmp@driver	.... <u>843</u> , 849–851	144, 146, 148,	
.. <u>3</u> , <u>1682</u>	\hyxmp@legal	150, 152, 154,	
\hyxmp@embed@packet	..... <u>1147</u>	156, 158, 160,	
..... 374, <u>1682</u>	\hyxmp@list	165, 170, 172,	
\hyxmp@embed@packet@dvi	1094, 1244, 1245	174, 176, 178,	
\hyxmp@embed@packet@dvi	\hyxmp@list@to@lines	180, 182, 832, 845	
\hyxmp@embed@packet@dvi	..... <u>1229</u> ,	\hyxmp@pdfua@id@extensions	
\hyxmp@embed@packet@dvi	1258, 1264–1266	..... <u>1425</u> , 1604	
\hyxmp@embed@packet@dvi	..... <u>1694</u> , <u>1764</u>	\hyxmp@pdfua@id@schema	
\hyxmp@embed@packet@dvi	\hyxmp@list@to@xml	..... <u>1205</u> , 1669	
\hyxmp@embed@packet@dvi	.. <u>1076</u> , 1130, 1131		
\hyxmp@embed@packet@dvi	\hyxmp@major@minor		
\hyxmp@embed@packet@dvi	<u>1019</u>		
\hyxmp@embed@packet@dvi	\hyxmp@mm@extensions		
\hyxmp@embed@packet@dvi	..... <u>1706</u> , <u>1734</u>		
\hyxmp@embed@packet@dvi	..... <u>1384</u> , 1598		
\hyxmp@embed@packet@dvi	\hyxmp@mm@schema		
\hyxmp@embed@packet@dvi	..... <u>1686</u> , <u>1714</u>		
\hyxmp@embed@packet@dvi	..... <u>1165</u> , 1671		

\hyxmp@pdfx@id@extensions	\hyxmp@standards .. 324	1066, 1088, 1108,
..... 1437, 1608	\hyxmp@string@len ..	1115, 1137, 1244
\hyxmp@pdfx@id@schema	..... 1765, 1780	\hyxmp@xmllify .....
..... 1208, 1670	\hyxmp@sublist .....	..... 351, 596,
\hyxmp@pdfx@major ..	. 390, 391, 394, 395	789, 799, 807,
... 84, 93, 109,	\hyxmp@suppress@pdf@info	816, 833, 835,
1209, 1438, 1454	..... 184	1047, 1055, 1064,
\hyxmp@photoshop@data	\hyxmp@temp@list .. 414	1087, 1104, 1243
..... 1194	\hyxmp@temp@str ... 414	\hyxmp@xmp@basic@schema
\hyxmp@photoshop@schema	\hyxmp@text .....	..... 1173, 1667
..... 1194, 1666	. 596, 674, 704, 748	\hyxmp@xmp@to@pdf@date
\hyxmp@prism@data ..	\hyxmp@textunderscore	.... 462, 465, 576
.. 1274, 1306, 1614	..... 20	\hyxmp@xmp@to@pdf@date@i
\hyxmp@prism@extensions	\hyxmp@timestamp .. 565	..... 466, 468
..... 1525, 1616	\hyxmp@today@pdf 366, 575	\hyxmp@xmp@to@pdf@date@ii
\hyxmp@prism@schema	\hyxmp@today@xmp ..	..... 471, 474
..... 1273, 1673	..... 553, 558,	\hyxmp@xmp@to@pdf@date@iii
\hyxmp@ProcessKeyvalOptions	576, 949, 1127,	..... 477, 480
..... 251	1175, 1181, 1187	\hyxmp@xmp@to@pdf@date@iv
\hyxmp@rand@num 882,	\hyxmp@today@xmp@define	..... 483, 486
891, 926, 943, 952	.... 524, 573, 947	\hyxmp@xmp@to@pdf@date@v
\hyxmp@rdf@dc .....	\hyxmp@toxml .. 626, 649	..... 489, 492
.. 1041, 1123–1125	\hyxmp@toxml@unicodetex	\hyxmp@xmp@to@pdf@date@vi
\hyxmp@redefine@Hyp	..... 614, 674	..... 495, 499
.... 211, 253, 258	\hyxmp@trimb .. 582, 585	\hyxmp@xmp@to@pdf@date@vii
\hyxmp@remove@this ..	\hyxmp@trimc .. 585, 586	.... 502, 505, 515
..... 1010, 1013	\hyxmp@trimspaces ..	\hyxmp@xmp@to@pdf@date@viii
\hyxmp@rights . 1147,	..... 394, 578	..... 518, 521
1150, 1154, 1156	\hyxmp@try .....	\hyxmp@xmpRights@schema
\hyxmp@seed@rng ...	\hyxmp@try@today 552,	..... 1146, 1664
.... 865, 942, 951	559, 562, 565, 572	\hyxmp@zero .... 757,
\hyxmp@seed@rng@i ..	\hyxmp@unicodetextfalse	764, 771, 777, 782
..... 867, 869	..... 588	
\hyxmp@seed@string ..	\hyxmp@unicodetexttrue	
..... 940, 946	..... 588	
\hyxmp@set@pdfx@major	\hyxmp@uscore .. 22, 405	
..... 79, 109	\hyxmp@value . 845, 1048	
\hyxmp@set@pdfx@major@i	\hyxmp@x@default ..	
..... 79, 80	346, 991, 1053, 1061	
\hyxmp@set@pdfx@major@ii	\hyxmp@xetex@crap ..	
..... 81, 84	..... 605, 704	
\hyxmp@set@rand@num	\hyxmp@xml . 791, 793,	
.... 882, 890, 925	831, 837, 972,	
\hyxmp@singleton@dc	979, 1356, 1633,	
.. 1102, 1126–1129	1723, 1731, 1754,	
\hyxmp@skiptorelax ..	1765, 1772, 1803	
..... 741, 747	\hyxmp@xmllified 596,	
\hyxmp@skipzeros .. 699	792, 801, 808,	
\hyxmp@SpaceOther ..	819, 823, 834,	
..... 708, 721	836, 1048, 1057,	

## I

IETF .....	5
\if@tempswa ..	1046, 1081
ifdraft .....	22
\ifdraft .....	119, 122
\iffalse ...	1041, 1076
\ifHy@pdfa .....	
312, 1123, 1124,	
1130, 1200, 1599	
\ifhyxmp@unicodetex	588, 599, 958, 1621
\ifLuaTeX .....	994
ifluatex .....	74
\ifluatex ..	1716, 1720
ifmtarg .....	18, 85
\ifPDFTeX .....	997
iftex .....	18, 87

ifthen . . . . . 18	ngerman . . . . . 17, 82	pdfisbn . . . . . 4, 6
\ifthenelse . . . . . 88	\number 751, 753, 755,	pdfissn . . . . . 4, 6
\ifXeTeX . . . . . 604, 1000	760, 762, 767, 769	pdfissuenum . . . . . 4, 7
Info . . . . . 8, 11–13, 25, 31, 53	\numexpr . . . . . 1732	pdfkeywords . . . . . 4,
intcalc . . . . . 18		12, 14, 18, 26, 57
\intcalcDiv 753, 760, 767	<b>O</b>	pdflang . . . . . 4–7,
\intcalcMod 755, 762, 769	options	15, 18, 30, 47, 57, 85
IPTC . . . . . 13, 24,	baseurl . . . . .	pdflicenseurl . . . . .
51, 61–63, 67, 71, 84	4, 6, 14, 18, 24, 59	. . . . . 5, 14, 58, 83
lptc4xmpCore:ContactInfo	draft . . . . . 8	pdfmark . . . . . 74
. . . . . 62, 68	dvipdf . . . . . 74	pdfmetadate . . . . .
lptc4xmpCore:CreatorContactInfo	dvips . . . . . 74	. . . . . 5, 7, 8, 19, 85
. . . . . 2, 3, 61, 62, 84	dvipsone . . . . . 74	pdfmetalang . . . . .
ISBN . . . . . 2, 6, 23	dviwindo . . . . . 74	5, 6, 15, 47, 82, 85
ISO . . . . . 5–7, 15, 20, 47	keeppdfinfo . . . . . 8, 13, 25	pdfmoddate 4, 7, 8, 85
ISSN . . . . . 2, 6, 23	nativepdf . . . . . 74	pdfnumpages . . . . .
	pdfa . . . . . 8, 20, 29, 85	. . . . . 5, 7, 16, 17
<b>J</b>	pdfaconformance . . . . .	pdfpagerange . . . . .
\jobname 111, 305, 378,	. . . . . 4, 8, 60	. . . . . 5, 7, 16, 17
566, 941, 949, 1703	pdfapart . . . . . 4, 8, 20, 60	pdfproducer . . . . . 4, 18, 53
<b>K</b>	pdfauthor . . . . .	pdfpublication . . . . . 5, 7
keeppdfinfo (option) . . . . .	. . . . . 4, 5, 12, 14,	pdfpublisher . . . . . 5, 7
. . . . . 8, 13, 25	15, 18, 26, 30, 57, 84	pdfpubtype . . . . . 5, 7
Keywords . . . . . 11, 25, 54, 86	pdfauthortitle . . . . . 4, 5, 14	pdfrendition . . . . . 5, 8, 86
\KV@Hyp@pdfauthor . . . . . 218	pdfbookedition . . . . . 4, 7	pdfsource . . . . . 5, 9, 85
\KV@Hyp@pdfkeywords 239	pdfbytes . . . . . 4, 8	pdfsubject 4, 12, 18, 57
kvoptions . . . . . 18, 27	pdfcaptionwriter . . . . . 4, 5	pdfsubtitle . . . . . 5
<b>L</b>	pdfcontactaddress . . . . .	pdftitle . . . . . 4, 5,
LF . . . . . 61	. . . . . 4, 5, 13	12, 15, 18, 30, 57, 84
LuaL <sup>A</sup> T <sub>E</sub> X . . . . . 10,	pdfcontactcity . . . . . 4, 5	pdftrapped 4, 8, 18, 86
13, 14, 38, 54, 84	pdfcontactcountry 4, 5	pdftype . . . . . 5, 8, 57, 85
LuaT <sub>E</sub> X . . . . . 39, 42,	pdfcontactemail . . . . . 4, 5	pdfuapart . . . . . 5, 8, 20, 60
73, 74, 77, 83, 85, 87	pdfcontactphone . . . . . 4, 5	pdfurl . . . . . 5, 6
luatex85 . . . . . 87	pdfcontactpostcode . . . . .	pdfversionid . . . . . 5, 6, 58
\luatexbanner . . . . . 995	. . . . . 4, 5	pdfvolumenum . . . . . 5, 7
<b>M</b>	pdfcontactregion . . . . . 4, 5	pdfxstandard . . . . .
\makeatletter . . . . . 1010	pdfcontacturl . . . . . 4, 5, 14	. . . . . 5, 8, 9, 20, 22, 60
memoir . . . . . 84	pdfcopyright . . . . .	ps2pdf . . . . . 74
Metadata . . . . . 11, 73, 76	. . . . . 4, 5, 57, 58, 83	textures . . . . . 74
\month . . . . . 526, 527, 529	pdfcreationdate . . . . .	unicode . . . . . 15, 85
<b>N</b>	. . . . . 4, 7, 8, 31, 85	vtexpdfmark . . . . . 74
NAK . . . . . 18, 33, 40	pdfdate . . . . . 4,	
nativepdf (option) . . . . . 74	7, 15, 19, 51, 84, 87	<b>P</b>
\newif . . . . . 588	pdfdocumentid . . . . .	\PackageWarning . . . . .
\next . . . . . 30, 41,	. . . . . 4, 6, 85, 86	. . . . . 62, 97, 415
89, 96, 184, 389,	pdfdoi . . . . . 4, 6	\PackageWarningNoLine
567, 570, 869, 1606	pdf ISSN . . . . . 4, 6	. . . . . 186, 304,
	pdfidentifier 4, 6, 57, 87	314, 331, 377, 1701
	pdfinstanceid . . . . .	\patchcmd . . . . . 192, 198
	. . . . . 4, 6, 85, 86	

PDF ..... 1–4, 7, 8,  
 10–14, 16, 19, 25,  
 29, 31, 32, 34, 35,  
 38, 40, 42, 50, 51,  
 53, 54, 65, 70, 71,  
 73, 74, 76, 84, 85, 87  
 PDF/A ..... 3,  
 8, 12, 13, 20, 25,  
 29, 52, 54, 60, 63,  
 66, 67, 69, 70, 83–85  
 PDF/UA ..... 3, 8,  
 20, 29, 60, 66, 70, 87  
 PDF/X ... 3, 8, 9, 21,  
 22, 29, 60, 67, 71, 87  
 pdf:Keywords ... 2, 12, 54  
 pdf:PDFVersion . 3, 54, 87  
 pdf:Producer ... 3, 52, 54  
 pdf:trapped ..... 3  
 \PDF@FinishDoc ....  
 .... 185, 193, 199  
 pdfa (option) 8, 20, 29, 85  
 pdfa:conformance (op-  
 tion) ... 4, 8, 60  
 pdfaid:conformance ... 3  
 pdfaid:part ..... 3  
 pdfapart (option) ....  
 .... 4, 8, 20, 60  
 pdfaType:prefix ..... 84  
 pdfauthor (option) ...  
 ... 4, 5, 12, 14,  
 15, 18, 26, 30, 57, 84  
 pdfauthor:tile (option)  
 ..... 4, 5, 14  
 pdfbookedition (option)  
 ..... 4, 7  
 pdfbytes (option) ... 4, 8  
 pdfcaptionwriter (op-  
 tion) ..... 4, 5  
 \pdfcatalog ..... 1724  
 \pdfcompresslevel . 1718  
 pdfcontactaddress (op-  
 tion) ... 4, 5, 13  
 pdfcontactcity (option)  
 ..... 4, 5  
 pdfcontactcountry (op-  
 tion) ..... 4, 5  
 pdfcontactemail (op-  
 tion) ..... 4, 5  
 pdfcontactphone (op-  
 tion) ..... 4, 5  
 pdfcontactpostcode (op-  
 tion) ..... 4, 5  
 pdfcontactregion (op-  
 tion) ..... 4, 5  
 pdfcontacturl (option)  
 ..... 4, 5, 14  
 pdfcopyright (option) .  
 ... 4, 5, 57, 58, 83  
 pdfcreationdate (option)  
 .... 4, 7, 8, 31, 85  
 \pdfcreationdate .. 560  
 pdfdate (option) ... 4,  
 7, 15, 19, 51, 84, 87  
 PDFDocEncoding ...  
 ..... 26, 40, 41  
 pdfdocumentid (option)  
 ..... 4, 6, 85, 86  
 pdfdoi (option) .... 4, 6  
 pdfeissn (option) ... 4, 6  
 pdfescape ..... 18  
 \pdfextension 1728, 1732  
 \pdffeedback . 563, 1732  
 pdfidentifier (option) .  
 ..... 4, 6, 57, 87  
 pdfinstanceid (option)  
 ..... 4, 6, 85, 86  
 pdfisbn (option) ... 4, 6  
 pdfissn (option) .... 4, 6  
 pdfissuenum (option) 4, 7  
 pdfkeywords (option) 4,  
 12, 14, 18, 26, 57  
 pdflang (option) . 4–7,  
 15, 18, 30, 47, 57, 85  
 \pdflastobj ..... 1724  
 pdfL<sup>A</sup>T<sub>E</sub>X 4, 10, 13, 38, 54  
 pdflicenseurl (option) .  
 ..... 5, 14, 58, 83  
 \pdfmajorversion . 1027  
 pdfmark (option) .... 74  
 \pdfmark ..... 1735,  
 1738, 1742,  
 1752, 1756, 1760  
 pdfmetadate (option) .  
 .... 5, 7, 8, 19, 85  
 pdfmetalang (option) .  
 5, 6, 15, 47, 82, 85  
 \pdfminorversion . 1023  
 pdfmoddate (option) .  
 ..... 4, 7, 8, 85  
 pdfnumpages (option)  
 ..... 5, 7, 16, 17  
 \pdfobj ..... 1720  
 pdfpagerange (option)  
 ..... 5, 7, 16, 17  
 pdfproducer (option) .  
 ..... 4, 18, 53  
 pdfpublication (option)  
 ..... 5, 7  
 pdfpublisher (option) 5, 7  
 pdfpubtype (option) . 5, 7  
 pdfrendition (option) .  
 ..... 5, 8, 86  
 pdfsource (option) 5, 9, 85  
 \pdfstringdef ..... 23  
 pdfsubject (option) ...  
 ..... 4, 12, 18, 57  
 pdfsubtitle (option) ... 5  
 pdfT<sub>E</sub>X ..... 8, 17,  
 42, 73, 74, 77, 85, 87  
 \pdf<sub>tex</sub>banner ..... 998  
 pdftitle (option) 4, 5,  
 12, 15, 18, 30, 57, 84  
 pdftrapped (option) ...  
 ..... 4, 8, 18, 86  
 pdftype (option) 5, 8, 57, 85  
 pdfuaid:part ..... 3  
 pdfuapart (option) ...  
 ..... 5, 8, 20, 60  
 pdfurl (option) .... 5, 6  
 \pdfvariable ..... 1031  
 pdfversionid (option) .  
 ..... 5, 6, 58  
 pdfvolumenum (option)  
 ..... 5, 7  
 pdfxid:GTS\_PDFXVersion  
 ..... 3  
 pdfxstandard (option)  
 . 5, 8, 9, 20, 22, 60  
 photoshop:AuthorsPosition  
 ..... 3, 59  
 photoshop:CaptionWriter  
 ..... 3, 59  
 PI ..... 51  
 PRISM ... 7, 62, 63, 69, 71  
 prism:aggregationType . 3  
 prism:bookEdition .... 2  
 prism:byteCount ..... 2  
 prism:doi ..... 2  
 prism:elssn ..... 2

prism:isbn . . . . .	2	prism:elssn . . . . .	2	\special . . . . .	1766,
prism:issn . . . . .	2	prism:isbn . . . . .	2		1774, 1803, 1809
prism:number . . . . .	2	prism:issn . . . . .	2	stringenc . . . . .	18
prism:pageCount . . . . .	3	prism:number . . . . .	2	\StringEncodingConvert	
prism:pageRange . . . . .	3	prism:pageCount . . . . .	3		601,
prism:publicationName . . . . .	3	prism:pageRange . . . . .	3		607, 618, 621, 716
prism:subtitle . . . . .	3	prism:publicationName		Subject . . . . .	11
prism:url . . . . .	3		3		
prism:volume . . . . .	3	prism:subtitle . . . . .	3		
\ProcessKeyvalOptions		prism:url . . . . .	3	<b>T</b>	
	<u>251</u>	prism:volume . . . . .	3	TeX . . . . .	15, 16,
Producer . . . . .	54	xmp:BaseURL . . . . .	2		18, 37, 38, 40, 42,
properties, XMP		xmp:CreateDate . . . . .	2, 31, 85	texdate . . . . .	48, 51–53, 58, 76, 77
dc:creator . . . . .	2, 12, 57, 85	xmp:CreatorTool . . . . .	3	Text . . . . .	15
dc:date . . . . .	2, 57	xmp:MetadataDate		\textunderscore . . . . .	65
dc:description . . . . .			2, 85		21, 22, 24
	3, 12, 47, 57, 85	xmp:ModifyDate . . . . .	2, 85	textures (option) . . . . .	74
dc:format . . . . .	2	xmpMM:DocumentID		\time . . . . .	536, 544
dc:identifier . . . . .	3, 57		3, 48, 58, 70	Title . . . . .	11
dc:language . . . . .		xmpMM:InstanceId		totpages . . . . .	16, 17
	3, 57, 83, 84		3, 48, 58, 70		
dc:publisher . . . . .	3	xmpMM:RenditionClass		<b>U</b>	
dc:rights . . . . .	2, 16, 47, 57		3	Unicode . . . . .	15, 18, 39–
dc:source . . . . .	2, 57, 83	xmpMM:VersionID			44, 56, 61, 71, 77, 83
dc:subject . . . . .	2, 57		3, 58, 86	unicode (option) . . . . .	15, 85
dc:title . . . . .	3, 12, 47, 57, 85	xmpRights:Marked		URI . . . . .	6
dc:type . . . . .	3, 57		2, 58, 83	URL . . . . .	2, 3, 5, 6, 14,
lptc4xmpCore:ContactInfo		xmpRights:WebStatement			19, 24, 25, 58, 59, 62
	62, 68		3, 58, 83	UTF-16BE . . . . .	41
lptc4xmpCore:CreatorContactInfo		ps2pdf (option) . . . . .	74	UTF-32BE . . . . .	40, 41
	2, 3, 61, 62, 84			UTF-8 . . . . .	41
pdf:Keywords . . . . .	2, 12, 54	<b>Q</b>		UUID . . . . .	3, 6, 22, 48, 50, 51, 84
pdf:PDFVersion . . . . .	3, 54	\Q . . . . .	578, 587	<b>V</b>	
pdf:Producer . . . . .	3, 52, 54			\vfuzz . . . . .	586
pdf:trapped . . . . .	3	<b>R</b>		vtexpdfmark (option) . . . . .	74
pdfaid:conformance . . . . .	3	RDF . . . . .	56	<b>X</b>	
pdfaid:part . . . . .	3	rdf:Description . . . . .	86	\x . . . . .	<u>704</u>
pdfaType:prefix . . . . .	84	rdf:li . . . . .	2	xdvipdfmx . . . . .	13, 14, 31, 76
pdfuaid:part . . . . .	3	rdf:Seq . . . . .	2	X <sub>q</sub> L <sup>A</sup> T <sub>E</sub> X . . . . .	10, 13,
pdfxid:GTS_PDFXVersion		\renewcommand . . . . .	252		14, 31, 38, 54, 84, 87
	3	\RequirePackage . . . . .	7, 10–17, 121, 1713	X <sub>q</sub> T <sub>E</sub> X . . . . .	39, 42,
photoshop:AuthorsPosition					43, 76, 77, 82, 83, 87
	3, 59	<b>S</b>		\XeTeXrevision . . . . .	1001
photoshop:CaptionWriter		\scantokens . . . . .	1007, 1010	\XeTeXversion . . . . .	1001
	3, 59	\SE->pdfdoc@03 . . . . .	<u>594</u>	XML . . . . .	1, 2, 13, 32, 39, 41,
prism:aggregationType		\SE->pdfdoc@15 . . . . .	<u>595</u>		42, 45, 46, 51, 54–
	3	\setkeys . . . . .	855		56, 61, 62, 64, 71, 85
prism:bookEdition . . . . .	2				
prism:byteCount . . . . .	2				
prism:doi . . . . .	2				

XMP . 1, 2, 4, 6–8, 11–	xmp:ModifyDate . . . 2, 85	xmpMM:RenditionClass 3
19, 21, 25, 30–35,	\xmpcomma . . . . 163,	xmpMM:VersionID 3, 58, 86
38, 39, 42, 45–48,	166, <u>218</u> , <u>239</u> , <u>400</u>	\xmpquote . . . . 164,
51–55, 58, 59, 63,	xmpincl . . . . . 4	167, <u>218</u> , <u>239</u> , <u>409</u>
65, 70, 73–77, 82–86	\XMPLangAlt . . . . . <u>852</u>	xmpRights:Marked 2, 58, 83
properties . . . . .	\xmplinesep . . . . .	xmpRights:WebStatement
<i>see</i> properties, XMP	. . <u>1224</u> , 1240, <u>1263</u>	. . . . . 3, 58, 83
xmp:BaseURL . . . . . 2	xmpMM:DocumentID .	\xmptilde . . . . . <u>410</u>
xmp:CreateDate . 2, 31, 85	. . . . . 3, 48, 58, 70	\XMPTruncateList . . <u>414</u>
xmp:CreatorTool . . . . . 3	xmpMM:InstanceID . .	
xmp:MetadataDate . . 2, 85	. . . . . 3, 48, 58, 70	<b>Y</b>
		\year . . . . . 525