The hyperxmp package*

Scott Pakin
scott+hyxmp@pakin.org

March 25, 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 v5.0, dated 2020/03/20.

```
      <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 (Iptc4xmpCore:CreatorContactInfo.CiAdrExtadr, Iptc4xmpCore:CreatorContactInfo.CiAdrCity, Iptc4xmpCore:CreatorContactInfo.CiAdrRegion, Iptc4xmpCore:CreatorContactInfo.CiAdrPcode, and Iptc4xmpCore: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 (Iptc4xmpCore:CreatorContactInfo.CiEmailWork)
- file format (dc:format)
- file name of main LaTeX source file (dc:source)
- file size in bytes (prism:byteCount)
- ISBN (prism:isbn)
- ISSN—both print (prism:issn) and electronic (prism:eIssn)
- issue number of parent publication (prism:number)
- keywords (pdf:Keywords and dc:subject)

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

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

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

xmpincl can embed XMP only when running under pdfLaTeX 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
- pdfinstanceid
- pdfisbn
- pdfissn
- pdfissuenum
- pdflicenseurl

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

2.1 Option descriptions

pdftitle
pdfsubtitle

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
pdfauthortitle
pdfcaptionwriter

hyperref's pdfauthor option specifies the document's author(s). See Note 4 on page 13 for a discussion of the correct syntax. If pdfauthor is not specified it will inherit its value from the document's `\author`. pdfauthortitle indicates the primary author's position or title. pdfcaptionwriter specifies the name of the person who added the metadata to the document.

pdfcontactaddress
pdfcontactcity
pdfcontactcountry
pdfcontactemail
pdfcontactphone
pdfcontactpostcode
pdfcontactregion
pdfcontacturl

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
pdflicenseurl

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

pdfmetalang

pdfmetalang indicates the natural language in which certain metadata—specifically, the document's title, subject, and copyright statement—are written. The language should be specified using an IETF language tag [10], for example, "`en`" for English, "`en-US`" for specifically United States English, "`de`" for German, and so forth. If pdfmetalang is not specified, hyperxmp assumes the metadata language is the same as the document language (hyperref's pdflang option). If neither pdfmetalang nor pdflang is specified, hyperxmp uses only "`x-default`" as the metadata language. Note that "`x-default`" metadata are always included in addition to the specified metadata language, as the user reading the document may not have specified a language preference.

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 `uuid:`*xxxxxxxx*-*xxxx*-*xxxx*-*xxxx*-*xxxxxxxxxxxx*, where "*x*" is a lowercase hexadecimal number. For example, `uuid:53ab7f19-a48c-5177-8bb2-403ad907f632` is a valid argument to pdfdocumentid (or pdfinstanceid). See Leach, Mealling, and Salz's UUID specification document for details on how to produce the various forms of UUIDs [11]. A more freeform mechanism than pdfinstanceid for versioning documents is available via pdfversionid. The version specified by pdfversionid can be incremented as `1`, `2`, `3`, …; identified with a hierarchical numbering scheme (e.g., this document is versioned `5.0` to match the package version); or labeled using any other approach. One possibility is to use a revision number or commit hash from the version-control software maintaining the document. For example, the `\gitVer` macro from the gitver package is an expandable (see Note 8 on page 15) version of the current Git hash that can suitably be passed to pdfversionid. If not specified, pdfversionid defaults to `1`.

pdfversionid

pdfisbn
pdfissn
pdfeissn
pdfdoi
pdfurl
baseurl

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

pdfpublication

Already-published documents can further be identified by the publication in which they appear. pdfpublication specifies the title of the journal, magazine, or other parent document. The title language is assumed to be the same as the document language (pdflang) but can be overridden by preceding the text with a bracketed ISO 639-1 two-letter language code and an optional ISO 3166-1 two-letter region code. For example, `pdfpublication={[fr]Charlie Hedbo}` indicates a French-language title. Were the language or pronunciation differences significant, `fr-FR` would indicate specifically the French spoken in France, as opposed to that spoken in, say, Canada (`fr-CA`) or Belgium (`fr-BE`). The publisher itself can be named using pdfpublisher.

pdfpublisher
pdfpubtype

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

pdfvolumenum
pdfissuenum
pdfpagerange
pdfbookedition

with pdfvolumenum and the issue number within the volume with pdfissuenum. pdfpagerange indicates the page numbers at which the document appears within the publication. The intention is that this be a comma-separated list of dash-separated ranges, as in `pdfpagerange={1,4-5}`. See Note 9 on page 16 for advice on how to assign pdfpagerange semi-automatically. For books, pdfbookedition names the edition of the book. This is specified as text, not a number. As with pdfpublication (above), pdfbookedition accepts a bracketed language code, as in `pdfbookedition={[en]Second edition}`.

pdfnumpages

The number of pages in the published, print version of the document can be expressed with pdfnumpages. Note 9 on page 16 explains how to automatically assign a value to pdfnumpages.

pdfdate

XMP metadata can include a number of dates (in fact, timestamps, as they include both date and time components). pdfdate specifies the document date. It is analogous to the LaTeX `\date` command, and, like `\date`, defaults to the date the document was built. It must be specified in either XMP format [4] or PDF format [3]. XMP dates are written in the form `YYYY-MM-DDThh:mm:ss+TT:tt`.[1] A W3C recommendation [14] discusses this format in more detail, but as an example, 14 hours, 15 minutes, 9 seconds past midnight U.S. Mountain Daylight Time (UTC-6) on the 23rd day of September in the year 2014 should be written as `2014-09-23T14:15:09-06:00`. This can be truncated (with loss of information) to `2014-09-23T14:15:09`, `2014-09-23T14:15`, `2014-09-23`, `2014-09`, or `2014` but no other subsets. PDF dates are written in the form `D:YYYYMMDDhhmmss+TT'tt'`. The same date in the preceding example would be written as `D:20140923141509-06'00'` in PDF format.

pdfcreationdate
pdfmoddate
pdfmetadate

The document's creation date, modification date, and metadata date are normally set automatically, but pdfcreationdate, pdfmoddate, and pdfmetadate can be used to override the defaults. Like pdfdate, pdfmetadate can be specified in either XMP or PDF format. However, because hyperref defines pdfcreationdate and pdfmoddate and expects these to be written as PDF dates, hyperxmp concomitantly accepts these two dates only in PDF format as well. Note that it's rare that a document would need to specify any of pdfcreationdate, pdfmoddate, or pdfmetadate.

pdftype

pdftype describes the type of document being produced. This refers to "the nature or genre of the resource" [4] such as `poem`, `novel` or `working paper`, as opposed to the file format (always `application/pdf` when generated by hyperxmp). Although pdftype can be assigned an arbitrary piece of text, the XMP specification recommends selecting types from a "controlled vocabulary" such as the DCMI Type Vocabulary [5]. The DCMI Type Vocabulary currently consists of only `Collection`, `Dataset`, `Event`, `Image`, `InteractiveResource`, `MovingImage`, `PhysicalObject`, `Service`, `Software`, `Sound`, `StillImage`, and `Text`. pdftype defaults to `Text`, which refers to "books, letters, dissertations, poems, newspapers, articles, archives of mailing lists," [5] and other forms of text—all things LaTeX is commonly used to typeset.

pdfrendition

Sometimes a base document is rendered in different forms. pdfrendition indi-

---

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

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

pdfbytes

The pdfbytes option expresses the document's file size in bytes. The intention is for this to be used to display an estimate of download time to a user or to serve as a quick check on whether a file was transmitted correctly between systems. This feature is easiest to use in conjunction with pdfTEX's `\pdffilesize` primitive: "`pdfbytes={\pdffilesize{\jobname.pdf}}`". Note that this requires a second run of `pdftex` because it queries the size of the PDF file from the *previous* run.

pdftrapped

hyperxmp honors hyperref's pdftrapped option. A document can indicate whether it employs color trapping by specifying pdftrapped=`True` or pdftrapped=`False`. (pdftrapped=`Unknown` is also allowed.) A current limitation of hyperxmp is that if a value other than `False` is provided, a document will additionally need to specify keeppdfinfo (page 12) to ensure that the PDF Info dictionary specifies the correct trapping value.

pdfapart
pdfaconformance

pdfapart and pdfaconformance, are used in conjunction with hyperref's pdfa option to claim a particular PDF/A standard by which the document abides. They default to pdfapart=`1` and pdfaconformance=`B`, indicating the PDF/A-1b standard. These can be changed (with caution) to assert that the document abides by a different standard (e.g., PDF/A-2u). A document that conforms to the PDF/UA standard can use pdfuapart to indicate the PDF/UA conformance level. For example, pdfuapart=`1` asserts that the document respects PDF/UA-1. pdfxstandard indicates the particular PDF/X standard by which the document abides. Unlike pdfpart and pdfaconformance, which accept a number and a letter, respectively, pdfxstandard 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.

pdfuapart
pdfxstandard

- 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 LATEX source file. It defaults to `\jobname.tex` but can be replaced by any other string. If pdfsource is given an empty argument, no document source will be specified at all.

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

2.2 A complete example

The following is a sample LaTeX document that provides values for most of the metadata options that hyperxmp recognizes:

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

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

\hypersetup{%
  pdftitle={%
    On a heuristic viewpoint concerning the production and
    transformation of light},
  pdfsubtitle={[en-US]Putting that bum Maxwell in his place},
  pdfauthor={Albert Einstein},
  pdfauthortitle={\xmpquote{Technical Assistant\xmpcomma\ Level III}},
  pdfdate={1905-03-17},
  pdfcopyright={Copyright (C) 1905, Albert Einstein},
  pdfsubject={photoelectric effect},
  pdfkeywords={energy quanta, Hertz effect, quantum physics},
  pdflicenseurl={http://creativecommons.org/licenses/by-nc-nd/3.0/},
  pdfcaptionwriter={Scott Pakin},
  pdfcontactaddress={Kramgasse 49},
  pdfcontactcity={Bern},
  pdfcontactpostcode={3011},
  pdfcontactcountry={Switzerland},
  pdfcontactphone={031 312 00 91},
  pdfcontactemail={aeinstein@ipi.ch},
  pdfcontacturl={%
    http://einstein.biz/,
    https://www.facebook.com/AlbertEinstein
  },
  pdfdocumentid={uuid:6d1ac9ec-4ff2-515a-954b-648eeb4853b0},
  pdfversionid={2.998e8},
  pdfpublication={[de]Annalen der Physik},
  pdfpublisher={Wiley-VCH},
  pdfpubtype={journal},
  pdfvolumenum={322},
  pdfissuenum={6},
  pdfpagerange={132-148},
  pdfnumpages={17},
  pdfissn={0003-3804},
  pdfeissn={1521-3889},
  pdflang={en},
  pdfmetalang={en},
  pdfurl={http://www.physik.uni-augsburg.de/annalen/history/einstein-papers/1905_17_132-148.pd
  pdfdoi={10.1002/andp.19053220607},
  pdfbytes={\pdffilesize{\jobname.pdf}}   % Requires pdflatex
}
\XMPLangAlt{de}{pdftitle={Über einen die Erzeugung und Verwandlung des
  Lichtes betreffenden heuristischen Gesichtspunkt}}

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

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

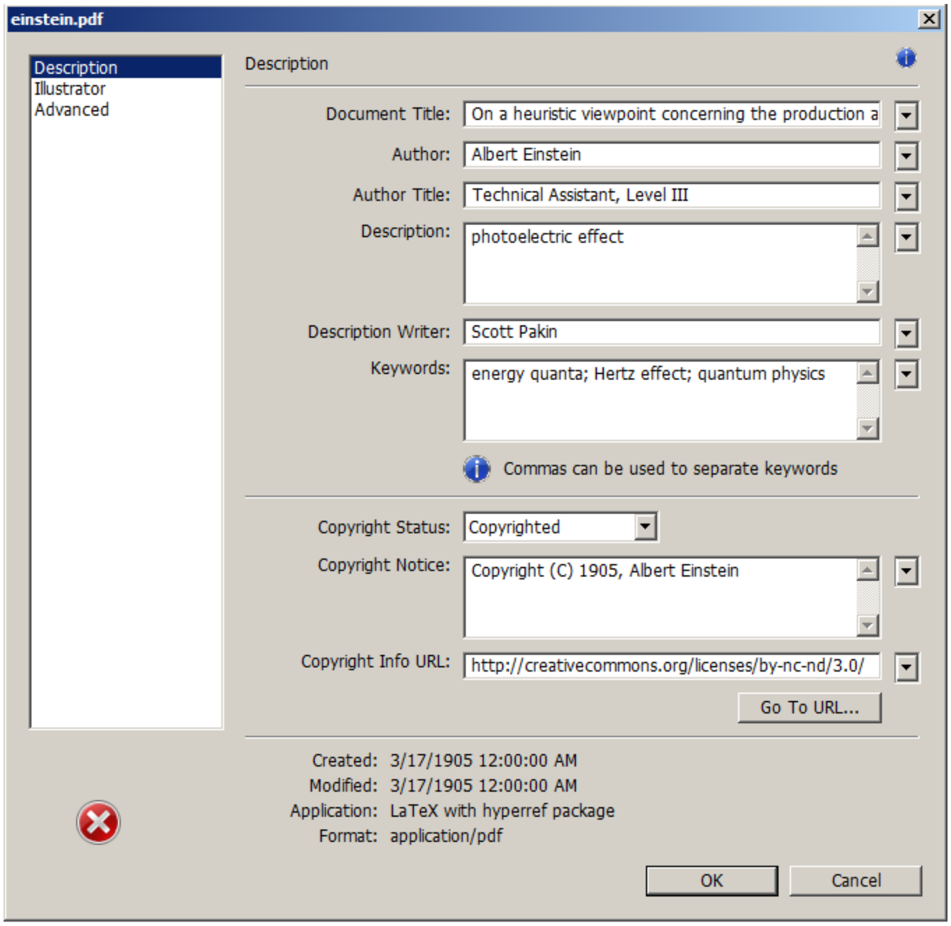
- pdfLaTeX
- LuaLaTeX
- LaTeX + Dvipdfm
- LaTeX + Dvips + Adobe Acrobat Distiller
- XeLaTeX

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

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

2.3 Usage notes

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



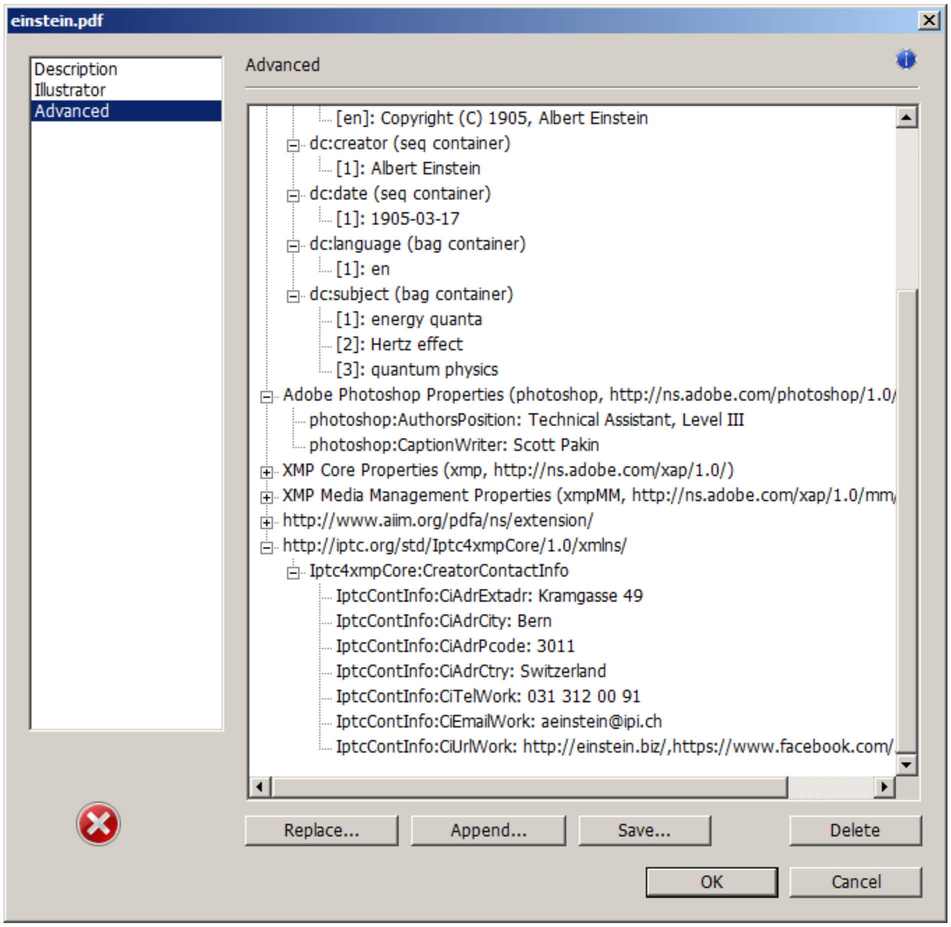
Document Title: On a heuristic viewpoint concerning the production a
Author: Albert Einstein
Author Title: Technical Assistant, Level III
Description: photoelectric effect
Description Writer: Scott Pakin
Keywords: energy quanta; Hertz effect; quantum physics
Commas can be used to separate keywords
Copyright Status: Copyrighted
Copyright Notice: Copyright (C) 1905, Albert Einstein
Copyright Info URL: http://creativecommons.org/licenses/by-nc-nd/3.0/
Created: 3/17/1905 12:00:00 AM
Modified: 3/17/1905 12:00:00 AM
Application: LaTeX with hyperref package
Format: application/pdf

Figure 1: XMP metadata as it appears in Adobe Acrobat

pdf:Keywords. The PDF/A specification indicates that values that appear in both the PDF Info dictionary and XMP packet must match. The problem is that in XMP, the author and keywords can be proper lists, as in

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

while in PDF, the author and keywords are specified as flat strings. Alas, there is no definition of how a list should be collapsed to a flat string: “`Curly Howard,`



[en]: Copyright (C) 1905, Albert Einstein
dc:creator (seq container)
[1]: Albert Einstein
dc:date (seq container)
[1]: 1905-03-17
dc:language (bag container)
[1]: en
dc:subject (bag container)
[1]: energy quanta
[2]: Hertz effect
[3]: quantum physics
Adobe Photoshop Properties (photoshop, http://ns.adobe.com/photoshop/1.0/
photoshop:AuthorsPosition: Technical Assistant, Level III
photoshop:CaptionWriter: Scott Pakin
XMP Core Properties (xmp, http://ns.adobe.com/xap/1.0/)
XMP Media Management Properties (xmpMM, http://ns.adobe.com/xap/1.0/mm/
http://www.aiim.org/pdfa/ns/extension/
http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/
Iptc4xmpCore:CreatorContactInfo
IptcContInfo:CiAdrExtadr: Kramgasse 49
IptcContInfo:CiAdrCity: Bern
IptcContInfo:CiAdrPcode: 3011
IptcContInfo:CiAdrCtry: Switzerland
IptcContInfo:CiTelWork: 031 312 00 91
IptcContInfo:CiEmailWork: aeinstein@ipi.ch
IptcContInfo:CiUrlWork: http://einstein.biz/,https://www.facebook.com/

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

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

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

keeppdfinfo

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

`\xmplinesep`

```
\renewcommand*{\xmlinesep}{;}
```

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 LuaLaTeX earlier than 0.85, hyperxmp leaves all PDF objects uncompressed. This is due to LuaLaTeX treating object compression as a global parameter, unlike pdfLaTeX, which treats it as a local parameter. Hence, when hyperxmp requests that the XMP packet be left uncompressed, LuaLaTeX 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 LuaLaTeX v0.85 onwards.
2. XƎLaTeX (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., LuaLaTeX), (2) pass the `--output-driver="xdvipdfmx -z0"` option to XƎLaTeX to instruct `xdvipdfmx` to turn off all compression (which will of course make the PDF file substantially larger), or (3) postprocess the generated PDF file by loading it into the commercial version of Adobe Acrobat and re-saving it with the Save As… menu option.

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

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

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

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

`\xmpcomma`
`\xmpquote`

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.

`\xmptilde`

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

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

Note 6: Automatically specified metadata pdftitle defaults to the document's title as specified by `\title{...}`. pdfauthor defaults to the document's author(s) as specified by `\author{...}`. pdfdate defaults to the current date and time. pdfmetalang defaults to the same value as pdflang if non-empty, "`x-default`" otherwise. An implication of automatic metadata specification is that an author can simply include `\usepackage{hyperxmp}` in a document's preamble and benefit from a modicum of XMP metadata with no additional effort.

`\XMPLangAlt`

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

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

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

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

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

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

```
\usepackage{texdate}
\initcurrdate
\hypersetup{%
  pdfcopyright={Copyright \textcopyright\ \printfdate{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 `\printfdate` 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 LaTeX to do that on your behalf:

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

3 Implementation

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

3.1 Initial preparation

`\hyxmp@dq@code`

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

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

`\hyxmp@at@end`
`\hyxmp@driver`

The `\hyxmp@at@end` macro includes code at the end of the document. For pdfTeX, 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 LaTeX run.

```
\def\hyxmp@driver{hpdftex}
\ifx\hyxmp@driver\Hy@driver
  \let\hyxmp@at@end=\AtEndDocument
\else
  \RequirePackage{atenddvi}
  \let\hyxmp@at@end=\AtEndDvi
\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 LaTeX's document-metadata naming conventions), we do not prefix metadata-related macro names with our package-specific `\hyxmp@` prefix. That is, we use names like `\@pdfcopyright` instead of `\hyxmp@pdfcopyright`.

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

```
\RequirePackage{kvoptions}
\RequirePackage{pdfescape}
\RequirePackage{stringenc}
\RequirePackage{intcalc}
\RequirePackage{iftex}
\RequirePackage{ifmtarg}
\RequirePackage{etoolbox}
\RequirePackage{ifthen}
```

`\@ifmtargexp`
`\@ifnotmtargexp`

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

```
\def\@ifmtargexp#1{\expandafter\@ifmtarg\expandafter{#1}}
\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.

```
\newcommand{\hyxmp@pdfstringdef}[2]{%
  \let\hyxmp@textunderscore=\textunderscore
  \let\textunderscore=\hyxmp@uscore
  \pdfstringdef{#1}{#2}%
  \let\textunderscore=\hyxmp@textunderscore
}
```

`\@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.

```
\def\@pdfdatetime{}
\define@key{Hyp}{pdfdate}{%
  \begingroup
    \Hy@unicodefalse
```

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

```
    \edef\next{%
      \noexpand\hyxmp@pdfstringdef\noexpand\@pdfdatetime{%
        \noexpand\hyxmp@as@xmp@date{#1}}%
    }%
    \next
  \endgroup
}
```

`\@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.

```
\def\@pdfmetadatetime{}
\define@key{Hyp}{pdfmetadate}{%
  \begingroup
    \Hy@unicodefalse
```

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

```
    \edef\next{%
      \noexpand\hyxmp@pdfstringdef\noexpand\@pdfmetadatetime{%
        \noexpand\hyxmp@as@xmp@date{#1}}%
    }%
    \next
  \endgroup
}
```

`\@pdfcopyright` Prepare to store the document's copyright statement.

```
\def\@pdfcopyright{}
\define@key{Hyp}{pdfcopyright}{\hyxmp@pdfstringdef\@pdfcopyright{#1}}
```

`\@pdftype` Prepare to store the document's logical type, which defaults to "`Text`".

```
\def\@pdftype{Text}
\define@key{Hyp}{pdftype}{\hyxmp@pdfstringdef\@pdftype{#1}}
```

`\@pdflicenseurl` Prepare to store the URL containing the document's license agreement.

```
\def\@pdflicenseurl{}
\define@key{Hyp}{pdflicenseurl}{\hyxmp@pdfstringdef\@pdflicenseurl{#1}}
```

`\@pdfauthortitle` Prepare to store the author's position/title (e.g., Staff Writer).

```
\def\@pdfauthortitle{}
\define@key{Hyp}{pdfauthortitle}{\hyxmp@pdfstringdef\@pdfauthortitle{#1}}
```

`\@pdfcaptionwriter` Prepare to store the name of the person who inserted the hyperxmp metadata.

```
\def\@pdfcaptionwriter{}
\define@key{Hyp}{pdfcaptionwriter}{\hyxmp@pdfstringdef\@pdfcaptionwriter{#1}}
```

`\@pdfmetalang` Prepare to store the natural language of the document's metadata, typically as an ISO 639-1 two-letter abbreviation.

```
\def\@pdfmetalang{}
\define@key{Hyp}{pdfmetalang}{\hyxmp@pdfstringdef\@pdfmetalang{#1}}
```

`\hyxmp@no@bad@parts` Complain about a bad pdfapart or pdfuapart if given trailing non-digits after a part number.

```
\def\hyxmp@no@bad@parts#1\relax{%
  \@ifnotmtarg{#1}{%
    \PackageWarning{hyperxmp}{pdfapart and pdfuapart must be numeric}%
  }%
}
```

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

```
\def\@pdfapart{1}
\define@key{Hyp}{pdfapart}{%
  \afterassignment\hyxmp@no@bad@parts\@tempcnta=0#1\relax
  \hyxmp@pdfstringdef\@pdfapart{\the\@tempcnta}%
}
```

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

```
\def\@pdfaconformance{B}
\define@key{Hyp}{pdfaconformance}{%
  \uppercase{\hyxmp@pdfstringdef\@pdfaconformance{#1}}%
}
```

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

```
\def\@pdfuapart{}
\define@key{Hyp}{pdfuapart}{%
  \afterassignment\hyxmp@no@bad@parts\@tempcnta=0#1\relax
  \hyxmp@pdfstringdef\@pdfuapart{\the\@tempcnta}%
}
```

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

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

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

```
\def\hyxmp@set@pdfx@major@i PDF/X-{%
  \afterassignment\hyxmp@set@pdfx@major@ii
  \@tempcnta=%
}
```

`\hyxmp@set@pdfx@major@ii`
`\hyxmp@pdfx@major` 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.

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

`\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.

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

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

```
\def\@pdfxstandard{}
\def\hyxmp@pdfx@major{}
\define@key{Hyp}{pdfxstandard}{%
  \hyxmp@pdfstringdef\@pdfxstandard{#1}%
```

`\next` Issue a warning message if the PDF/X standard named by the user does not appear in a list of known PDF/X standards. This is to caution the user that hyperxmp generates standard-specific XMP metadata and it can only guess at the correct format for new standard versions. (See the comments on page 58 above the definition of `\hyxmp@pdfx@id@schema`, for example.)

```
  \gdef\next{%
    \PackageWarning{hyperxmp}{Unrecognized PDF/X standard ‘#1’}%
  }%
  \hyxmp@check@std{#1}{PDF/X-1a:2001}%
  \hyxmp@check@std{#1}{PDF/X-1a:2003}%
  \hyxmp@check@std{#1}{PDF/X-3:2002}%
  \hyxmp@check@std{#1}{PDF/X-3:2003}%
  \hyxmp@check@std{#1}{PDF/X-4}%
  \hyxmp@check@std{#1}{PDF/X-4p}%
  \hyxmp@check@std{#1}{PDF/X-5g}%
  \hyxmp@check@std{#1}{PDF/X-5n}%
  \hyxmp@check@std{#1}{PDF/X-5pg}%
  \next
```

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

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

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

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

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

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

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

```
\def\hyxmp@InstanceID{}
\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`”.

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

```
\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`.

```
\ifdraft{%
  \def\@pdfrendition{draft}%
}{%
  \def\@pdfrendition{default}%
}
\define@key{Hyp}{pdfrendition}{\hyxmp@pdfstringdef\@pdfrendition{#1}}
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
\def\@pdfvolumenum{}
\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.

```
\def\@pdfissuenum{}
\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.

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

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

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

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

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

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

```
\def\@pdfcontactaddress{}
\define@key{Hyp}{pdfcontactaddress}{%
  \let\xmpcomma=\hyxmp@comma
  \def\xmpquote##1{##1}%
  \hyxmp@pdfstringdef\@pdfcontactaddress{#1}%
  \def\xmpcomma{,}%
  \let\xmpquote=\relax
}
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

`\hyxmp@suppress@pdf@info`
`\next`

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.

```
  \def\hyxmp@suppress@pdf@info{%
    \global\let\PDF@FinishDoc=\@empty
    \PackageWarningNoLine{hyperxmp}{%
      Suppressing the _entire_ PDF Info dictionary.\MessageBreak
      Please notify the hyperxmp maintainer%
    }%
  }%
  \let\next=\relax
  \patchcmd
    {\PDF@FinishDoc}%
    {/Author(\@pdfauthor)}%
    {}%
    {}%
    {\let\next=\hyxmp@suppress@pdf@info}%
  \patchcmd
    {\PDF@FinishDoc}%
    {/Keywords(\@pdfkeywords)}%
    {}%
    {}%
    {\let\next=\hyxmp@suppress@pdf@info}%
  \next
}
```

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

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`
`\hyxmp@pdfkeywords`

Prepare to store the name of the author and a list of keywords.

```
\def\hyxmp@pdfauthor{}
\def\hyxmp@pdfkeywords{}
```

`\hyxmp@redefine@Hyp`

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

```
\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.

```
  \@ifundefined{KV@Hyp@pdfauthor}{}{%
    \@ifundefined{hyxmp@Hyp@pdfauthor}{%
      \expandafter\let\expandafter\hyxmp@Hyp@pdfauthor
        \csname KV@Hyp@pdfauthor\endcsname
    }{}%
  }%
```

`\KV@Hyp@pdfauthor`
`\xmpcomma`
`\xmpquote`
`\hyxmp@and`
`\and`
`\hyxmp@pdfauthor`
`\@pdfauthor`

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

```
  \define@key{Hyp}{pdfauthor}{%
    \let\xmpcomma=\hyxmp@comma
    \def\xmpquote####1{####1}%
    \let\hyxmp@and=\and
    \def\and{,}%
    \hyxmp@Hyp@pdfauthor{##1}%
    \global\let\hyxmp@pdfauthor=\@pdfauthor
    \def\and{and\space}%
    \def\xmpcomma{,}%
    \def\xmpquote####1{"####1"}%
    \hyxmp@Hyp@pdfauthor{##1}%
    \def\xmpcomma{,}%
    \let\xmpquote=\relax
    \let\and=\hyxmp@and
  }%
```

`\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`.

```
  \@ifundefined{KV@Hyp@pdfkeywords}{}{%
    \@ifundefined{hyxmp@Hyp@pdfkeywords}{%
      \expandafter\let\expandafter\hyxmp@Hyp@pdfkeywords
        \csname KV@Hyp@pdfkeywords\endcsname
    }{}%
  }%
```

\KV@Hyp@pdfkeywords
\xmpcomma
\xmpquote
\hyxmp@pdfkeywords
\@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).

```
  \define@key{Hyp}{pdfkeywords}{%
    \let\xmpcomma=\hyxmp@comma
    \def\xmpquote####1{####1}%
    \hyxmp@Hyp@pdfkeywords{##1}%
    \global\let\hyxmp@pdfkeywords=\@pdfkeywords
    \def\xmpcomma{,}%
    \def\xmpquote####1{"####1"}%
    \hyxmp@Hyp@pdfkeywords{##1}%
    \def\xmpcomma{,}%
    \let\xmpquote=\relax
  }%
}
```

\hyxmp@ProcessKeyvalOptions
\ProcessKeyvalOptions

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

```
\let\hyxmp@ProcessKeyvalOptions=\ProcessKeyvalOptions
\renewcommand*{\ProcessKeyvalOptions}{%
  \hyxmp@redefine@Hyp
  \hyxmp@ProcessKeyvalOptions
}
```

\hyxmp@hypersetup
\hypersetup

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

```
\let\hyxmp@hypersetup=\hypersetup
\def\hypersetup{%
  \hyxmp@redefine@Hyp
  \hyxmp@hypersetup
}
```

\hyxmp@find@metadata
\hyxmp@concated@metadata

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

```
\newcommand*{\hyxmp@find@metadata}{%
  \edef\hyxmp@concated@metadata{%
    \@baseurl
    \@pdfauthor
    \@pdfauthortitle
    \@pdfbookedition
    \@pdfbytes
    \@pdfcaptionwriter
    \@pdfcontactaddress
    \@pdfcontactcity
    \@pdfcontactcountry
    \@pdfcontactemail
    \@pdfcontactphone
    \@pdfcontactpostcode
    \@pdfcontactregion
    \@pdfcontacturl
    \@pdfcopyright
    \@pdfcreationdate
    \@pdfdatetime
    \@pdfdoi
    \@pdfeissn
    \@pdfisbn
    \@pdfissn
    \@pdfissuenum
    \@pdfkeywords
    \@pdflang
    \@pdflicenseurl
    \@pdfmetadatetime
    \@pdfmoddate
    \@pdfnumpages
    \@pdfpagerange
    \@pdfpublication
    \@pdfpubtype
    \@pdfsubject
    \@pdfsubtitle
    \@pdftitle
    \@pdfuapart
    \@pdfurl
    \@pdfvolumenum
    \@pdfxstandard
  }%
  \ifx\hyxmp@concated@metadata\@empty
    \PackageWarningNoLine{hyperxmp}{%
      \jobname.tex did not specify any metadata to\MessageBreak
      include in the XMP packet.\space\space Please see the\MessageBreak
      hyperxmp documentation for instructions on how to\MessageBreak
      provide metadata values to hyperxmp}%
  \fi
}
```

`\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.

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

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

```
 \def\hyxmp@standards{%
  \@pdfapart
  \@pdfxstandard
  \@pdfuapart
 }%
```

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

```
 \@ifnotmtargexp{\hyxmp@standards}{%
  \@ifmtargexp{\@pdftitle}{%
   \PackageWarning{hyperxmp}{%
    Missing pdftitle (required for PDF standards\MessageBreak
    compliance)%
   }%
  }%
 }%
}
```

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.

```
\AtBeginDocument{%
 \@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`.

```
  \ifx\@pdflang\relax
   \let\@pdflang=\@empty
  \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.

```
  \ifx\@pdfmetalang\@empty
   \ifx\@pdflang\@empty
    \let\@pdfmetalang=\hyxmp@x@default
   \else
    \edef\@pdfmetalang{\@pdflang}%
   \fi
  \fi
  \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.

```
  \@ifmtargexp{\@pdftitle}{%
    \@ifnotmtargexp{\@title}{%
      \hypersetup{pdftitle={\@title}}%
    }%
  }%
  {}%
  \@ifmtargexp{\@pdfauthor}{%
    \@ifnotmtargexp{\@author}{%
      \hypersetup{pdfauthor={\@author}}%
    }%
  }%
  {}%
```

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 XƎLaTeX. 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.

```
  \@ifundefined{XeTeXversion}{}{%
    \@ifmtargexp{\@pdfcreationdate}{%
      \let\@pdfcreationdate=\hyxmp@today@pdf
    }%
    {}%
  }%
```

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

```
  \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.

```
  \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.

```
  \hyxmp@at@end{%
    \hyxmp@find@metadata
    \hyxmp@embed@packet
  }%
}{%
```

```
    \PackageWarningNoLine{hyperxmp}{%
      \jobname.tex failed to include a\MessageBreak
      \string\usepackage\string{hyperref\string}
      in the preamble.\MessageBreak
      Consequently, all hyperxmp functionality will be\MessageBreak
      disabled}%
  }%
}
```

3.3 Manipulating author-supplied data

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

`\hyxmp@commas@to@list`

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

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

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

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

```
\def\hyxmp@commas@to@list@i#1#2,{%
  \gdef\hyxmp@sublist{#2}%
  \ifx\hyxmp@sublist\@empty
    \let\next=\relax
  \else
    \hyxmp@trimspaces\hyxmp@sublist
    \@cons{#1}{{\hyxmp@sublist}}%
    \def\next{\hyxmp@commas@to@list@i{#1}}%
  \fi
  \next
}
```

`\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.

```
\def\xmpcomma{,}%
```

`\hyxmp@comma`

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

```
\bgroup
  \catcode`\^^C=11
  \gdef\hyxmp@comma{^^C}
\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.

```
\bgroup
  \catcode`\^^U=11
  \gdef\hyxmp@uscore{^^U}
\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.

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

`\xmptilde`

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

```
\bgroup
  \catcode`\~=12%
  \gdef\xmptilde{~}%
\egroup
```

`\XMPTruncateList`
`\hyxmp@temp@str`
`\hyxmp@temp@list`
`\@elt`

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.

```
\newcommand{\XMPTruncateList}[1]{{%
    \PackageWarning{hyperxmp}{%
      \noexpand\XMPTruncateList has been deprecated since\MessageBreak
      hyperxmp 4.0 and may be removed in future\MessageBreak
      versions of the package.  \noexpand\XMPTruncateList\MessageBreak
      was found}%
  \edef\hyxmp@temp@str{\csname hyxmp@#1\endcsname}%
  \hyxmp@commas@to@list{\hyxmp@temp@list}{\hyxmp@temp@str}%
  \def\@elt##1{%
    \expandafter\gdef\csname @#1\endcsname{##1}%
    \let\@elt=\@gobble
  }
  \hyxmp@temp@list
}}
```

3.3.2 Date manipulation

hyperxmp needs to manipulate two types of date (really, timestamp) formats: PDF format and XMP format. PDF timestamps are of the form “D:YYYYMMDDhhmmss+TT’tt’” (e.g., `D:20200325151959-06’00’`) [3], while XMP timestamps are of the form “YYYY-MM-DDThh:mm:ss+TT:tt” (e.g., `2020-03-25T15:19:59-06:00`) [4]. The `\hyxmp@as@pdf@date` and `\hyxmp@as@xmp@date` macros defined in this section facilitate timestamp conversions to PDF and XMP formats, respectively.

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

```
\def\hyxmp@first@char#1{\hyxmp@first@char@i#1\relax}
\def\hyxmp@first@char@i#1#2\relax{#1}
```

`\hyxmp@as@xmp@date` If necessary, convert a timestamp to XMP format. That is, if the timestamp is in PDF format, convert it; otherwise, leave it unmodified. This macro is fully expandable.

```
\def\hyxmp@as@xmp@date#1{%
  \expandafter\ifnum\expandafter`\hyxmp@first@char@i#1\relax=`D
    \hyxmp@pdf@to@xmp@date{#1}%
  \else
    #1%
  \fi
}
```

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

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

`\hyxmp@parse@time` This is a helper function for `\hyxmp@pdf@to@xmp@date`. `\hyxmp@pdf@to@xmp@date` proper parses only the year, month, and day then calls `\hyxmp@parse@time`. `\hyxmp@parse@time` parses the hours, minutes, and seconds then calls `\hyxmp@parse@tz@char`.

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

`\hyxmp@parse@tz@char`

This is another helper function for `\hyxmp@pdf@to@xmp@date`. So far, the date and time have been parsed. `\hyxmp@parse@tz@char` parses the first character of the timezone descriptor. This can be one of "`+`" for eastern timezones (UTC$+x$, including Asia, Oceania, and most of Europe), "`-`" for western timezones (UTC$-x$, primarily the Americas), or "`Z`" for Zulu time (UTC+0). Timezones beginning with "`+`" or "`-`" are followed by an offset in hours and minutes (parsed by `\hyxmp@parse@tz`; timezones beginning with "`Z`" are not.

```
\def\hyxmp@parse@tz@char#1{%
  #1%
  \ifx#1-%
    \expandafter\hyxmp@parse@tz
  \else
    \ifx#1+%
      \expandafter\hyxmp@parse@tz
    \fi
  \fi
}
```

`\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.

```
\def\hyxmp@parse@tz#1'#2'{%
  #1:#2%
}
```

`\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.

```
\def\hyxmp@as@pdf@date#1{%
  \expandafter\ifx\hyxmp@first@char@i#1\relax D%
    #1%
  \else
    \hyxmp@xmp@to@pdf@date{#1}%
  \fi
}
```

`\hyxmp@xmp@to@pdf@date`

Convert a timestamp from XMP format to PDF format. This macro is fully expandable.

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

`\hyxmp@xmp@to@pdf@date@i`

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

```
\def\hyxmp@xmp@to@pdf@date@i#1#2#3#4#5#6{%
  #1#2#3#4%
  \ifx#5-%
    \expandafter\hyxmp@xmp@to@pdf@date@ii\expandafter#6%
  \fi
}
```

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

```
\def\hyxmp@xmp@to@pdf@date@ii#1#2#3#4{%
  #1#2%
  \ifx#3-%
    \expandafter\hyxmp@xmp@to@pdf@date@iii\expandafter#4%
  \fi
}
```

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

```
\def\hyxmp@xmp@to@pdf@date@iii#1#2#3#4{%
  #1#2%
  \ifx#3T%
    \expandafter\hyxmp@xmp@to@pdf@date@iv\expandafter#4%
  \fi
}
```

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

```
\def\hyxmp@xmp@to@pdf@date@iv#1#2#3#4{%
  #1#2%
  \ifx#3:%
    \expandafter\hyxmp@xmp@to@pdf@date@v\expandafter#4%
  \fi
}
```

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

```
\def\hyxmp@xmp@to@pdf@date@v#1#2#3#4{%
  #1#2%
  \ifx#3:%
    \expandafter\hyxmp@xmp@to@pdf@date@vi\expandafter#4%
  \fi
}
```

`\hyxmp@gobbletwo` This is exactly the same as LaTeX 2$_\varepsilon$'s `\@gobbletwo` but needs to be a different literal for `\hyxmp@xmp@to@pdf@date@vii`'s pattern-matching to work.

```
\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).

```
\def\hyxmp@xmp@to@pdf@date@vi#1#2#3#4{%
  #1#2%
  \ifx#3+%
    +\expandafter\hyxmp@xmp@to@pdf@date@vii
  \fi
  \ifx#3-%
    -\expandafter\hyxmp@xmp@to@pdf@date@vii
  \fi
  \ifx#3Z%
    Z%
  \fi
  \ifx#3\relax
    \expandafter\hyxmp@gobbletwo
  \fi
  \@gobbletwo #4%
}
```

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

```
\def\hyxmp@xmp@to@pdf@date@vii#1\@gobbletwo#2#3#4#5{%
  #2#3%
  \ifx#4:%
    \expandafter\hyxmp@xmp@to@pdf@date@viii\expandafter#5%
  \fi
}
```

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

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

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

```
\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.

```
  \xdef#1{\the\year}%
  \ifnum\month<10
    \xdef#1{#1-0\the\month}%
  \else
    \xdef#1{#1-\the\month}%
  \fi
  \ifnum\day<10
    \xdef#1{#1-0\the\day}%
  \else
    \xdef#1{#1-\the\day}%
  \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`.

```
  \@tempcnta=\time
  \divide\@tempcnta by 60
  \ifnum\@tempcnta<10
    \xdef#1{#1T0\the\@tempcnta}%
  \else
    \xdef#1{#1T\the\@tempcnta}%
  \fi
  \multiply\@tempcnta by -60
  \advance\@tempcnta by \time
  \ifnum\@tempcnta<10
    \xdef#1{#1:0\the\@tempcnta}%
  \else
    \xdef#1{#1:\the\@tempcnta}%
  \fi
  \xdef#1{#1Z}%
}
```

`\hyxmp@try@today`

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

```
\def\hyxmp@try@today#1#2{%
  \@ifmtargexp{\hyxmp@today@xmp}{%
    \@ifundefined{#1}{}{#2}%
  }{}%
}
```

`\hyxmp@today@xmp`

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

```
\def\hyxmp@today@xmp{}
```

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

```
\hyxmp@try@today{pdfcreationdate}{%
  \edef\hyxmp@today@xmp{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
}
```

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

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

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

```
\hyxmp@try@today{filemoddate}{%
  \edef\hyxmp@today@xmp{\filemoddate{\jobname.aux}}%
  \edef\next{%
    \edef\noexpand\hyxmp@today@xmp{\noexpand\hyxmp@as@xmp@date{\hyxmp@today@xmp}}%
  }%
  \next
}%
```

Case 4: None of the above. Do the best we can using the available TeX primitives (`\year`, `\month`, `\day`, and `\time`.

```
\hyxmp@try@today{year}{%
  \hyxmp@today@xmp@define\hyxmp@today@xmp
}
```

`\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`.

```
\expandafter\edef\expandafter\hyxmp@today@pdf\expandafter{%
  \expandafter\hyxmp@xmp@to@pdf@date\expandafter{\hyxmp@today@xmp}%
}
```

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.

```
\catcode`\Q=3
```

`\hyxmp@trimspaces\x` redefines `\x` to have the same replacement text sans leading and trailing space tokens.

```
\newcommand{\hyxmp@trimspaces}[1]{%
```

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

```
  \begingroup
```

Put “`\toks 0 {`” into the `afterassignment` queue.

```
  \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.

```
  \expandafter\hyxmp@trimb\expandafter\noexpand#1Q Q}%
```

Transfer the trimmed text back into `#1`.

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

`\hyxmp@trimb` `\hyxmp@trimb` removes a trailing space if present, then calls `\hyxmp@trimc` to clean up any leftover bizarre `Q`s, and trim a leading space. In order for `\hyxmp@trimc` to work properly we need to put back a `Q` first.

```
\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.

```
\def\hyxmp@trimc#1Q#2{\afterassignment\endgroup \vfuzz\the\vfuzz#1}
\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`
`\hyxmp@unicodetextrue`
`\hyxmp@unicodetexfalse`

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

```
\newif\ifhyxmp@unicodetex
\ifnum64=`\^^^^0040\relax
  \hyxmp@unicodetextrue
\else
  \hyxmp@unicodetexfalse
\fi
```

`\SE->pdfdoc@03`

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

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

`\SE->pdfdoc@15`

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

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

`\hyxmp@xmlify`
`\hyxmp@xmlified`
`\hyxmp@text`

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

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

Escaped PDF string → PDFDocEncoding/Unicode

```
  \EdefUnescapeString\hyxmp@text{#1}%
  \ifhyxmp@unicodetex
```

PDFDocEncoding/Unicode → UTF-32BE

```
    \hyxmp@is@unicode\hyxmp@text{%
      \StringEncodingConvert
      \hyxmp@text\hyxmp@text{utf16be}{utf32be}%
    }{%
      \ifXeTeX
        \hyxmp@xetex@crap
      \else
        \StringEncodingConvert
    \hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
   \fi
  }%
```

UTF-32BE → UTF-32BE as hex string

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

UTF-32BE → XML in ASCII

```
  \edef\hyxmp@text{%
   \expandafter
  }\expandafter\hyxmp@toxml@unicodetex\hyxmp@text
  \relax\relax\relax\relax\relax\relax\relax\relax
 \else
```

PDFDocEncoding/Unicode → UTF-8

```
  \hyxmp@is@unicode\hyxmp@text{%
   \StringEncodingConvert
   \hyxmp@text\hyxmp@text{utf16be}{utf8}%
  }{%
   \StringEncodingConvert
   \hyxmp@text\hyxmp@text{pdfdoc}{utf8}%
  }%
```

UTF-8 → UTF-8 as hex string

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

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

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

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

```
  \EdefUnescapeHex\hyxmp@text{\hyxmp@text}%
 \fi
 \global\let\hyxmp@xmlified\hyxmp@text
}
```

`\hyxmp@is@unicode`
`\hyxmp@@is@unicode`

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

```
\begingroup
 \lccode‘\<=254 %
 \lccode‘\>=255 %
 \catcode254=12 %
 \catcode255=12 %
\lowercase{\endgroup
 \def\hyxmp@is@unicode#1{%
  \expandafter\hyxmp@@is@unicode#1<>\@nil
 }%
 \def\hyxmp@@is@unicode#1<>#2\@nil{%
  \ifx\\#1\\%
   \expandafter\@firstoftwo
  \else
```

```
      \expandafter\@secondoftwo
    \fi
  }%
}
```

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

```
\def\hyxmp@toxml#1#2{%
  \ifx#1\@empty
  \else
    \ifnum"#1#2=‘\& %
      26616D703B% &amp;
    \else\ifnum"#1#2=‘\< %
      266C743B% &lt;
    \else\ifnum"#1#2=‘\> %
      2667743B% &gt;
    \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.

```
      \@ifundefined{pdfmark}{%
        #1#2%
      }{%
      \ifnum"#1#2=‘\( %
        5C28% \(
      \else\ifnum"#1#2=‘\) %
        5C29% \)
      \else
        #1#2%
      \fi\fi
      }%
    \fi\fi\fi
    \expandafter\hyxmp@toxml
  \fi
}
```

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

```
\def\hyxmp@toxml@unicodetex#1#2#3#4#5#6#7#8{%
  \ifx#1\relax
  \else
    \ifnum"#1#2#3#4#5#6#7#8>127 %
      \uccode`\*="#1#2#3#4#5#6#7#8\relax
      \uppercase{%
        \edef\hyxmp@text{\hyxmp@text *}%
      }%
    \else\ifnum"#7#8=`\< %
      \edef\hyxmp@text{\hyxmp@text &lt;}%
    \else\ifnum"#7#8=`\& %
      \edef\hyxmp@text{\hyxmp@text &amp;}%
    \else\ifnum"#7#8=`\> %
      \edef\hyxmp@text{\hyxmp@text &gt;}%
    \else\ifnum"#7#8=`\ %
      \edef\hyxmp@text{\hyxmp@text\space}%
    \else
      \uccode`\*="#7#8\relax
      \uppercase{%
        \edef\hyxmp@text{\hyxmp@text *}%
      }%
    \fi\fi\fi\fi\fi
    \expandafter\hyxmp@toxml@unicodetex
  \fi
}
```

`\hyxmp@skipzeros`

Skip over leading zeroes in the input argument.

```
\def\hyxmp@skipzeros#1{%
  \ifx#10%
    \expandafter\hyxmp@skipzeros
  \fi
}
```

`\x`
`\hyxmp@xetex@crap`
`\hyxmp@try`
`\hyxmp@crap@result`
`\hyxmp@text`

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

```
\begingroup
\def\x#1{\endgroup
  \def\hyxmp@xetex@crap{%
    \edef\hyxmp@try{%
      \expandafter\hyxmp@SpaceOther\hyxmp@text#1\@nil
    }%
    \let\hyxmp@crap@result=N%
    \expandafter\hyxmp@crap@test\hyxmp@try\relax
    \ifx\hyxmp@crap@result Y%
      \let\hyxmp@text\@empty
      \expandafter\hyxmp@crap@convert\hyxmp@try\relax
    \else
      \StringEncodingConvert\hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
    \fi
  }%
}
\x{ }
```

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

```
\begingroup
  \catcode‘\~=12 %
  \lccode‘\~=‘\ %
\lowercase{\endgroup
  \def\hyxmp@SpaceOther#1 #2\@nil{%
    #1%
    \ifx\relax#2\relax
      \expandafter\@gobble
    \else
      ~%
      \expandafter\@firstofone
    \fi
    {\hyxmp@SpaceOther#2\@nil}%
  }%
}
```

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

```
\def\hyxmp@crap@test#1{%
  \ifx#1\relax
  \else
    \ifnum‘#1>127 %
      \let\hyxmp@crap@result=Y%
      \expandafter\expandafter\expandafter\hyxmp@skiptorelax
    \else
      \expandafter\expandafter\expandafter\hyxmp@crap@test
    \fi
  \fi
}
```

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

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

`\hyxmp@crap@convert`
`\hyxmp@num`
`\hyxmp@text`

Convert a hexadecimal string to a number.

```
\def\hyxmp@crap@convert#1{%
  \ifx#1\relax
  \else
    \edef\hyxmp@num{\number‘#1}%
    \ifnum\hyxmp@num>"FFFFFF %
      \lccode‘\!=\intcalcDiv{\hyxmp@num}{\number"1000000}\relax
      \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
      \edef\hyxmp@num{\intcalcMod{\hyxmp@num}{\number"1000000}}%
    \else
      \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
    \fi
    \ifnum\hyxmp@num>"FFFF %
      \lccode‘\!=\intcalcDiv{\hyxmp@num}{\number"10000}\relax
      \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
      \edef\hyxmp@num{\intcalcMod{\hyxmp@num}{\number"10000}}%
    \else
      \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
    \fi
    \ifnum\hyxmp@num>"FF %
      \lccode`\!=\intcalcDiv{\hyxmp@num}{\number"100}\relax
      \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
      \edef\hyxmp@num{\intcalcMod{\hyxmp@num}{\number"100}}%
    \else
      \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
    \fi
    \ifnum\hyxmp@num>0 %
      \lccode`\!=\hyxmp@num\relax
      \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
    \else
      \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
    \fi
    \expandafter\hyxmp@crap@convert
  \fi
}
```

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

```
\begingroup
  \catcode0=12 %
  \gdef\hyxmp@zero{^^00}%
\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.

```
\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.

```
\newcommand*{\hyxmp@add@simple}[2]{%
  \@ifnotmtargexp{#2}{%
    \hyxmp@xmlify{#2}%
    \hyxmp@add@to@xml{\hyxmp@extra@indent______<}%
    \xdef\hyxmp@xml{\hyxmp@xml#1}%
    \hyxmp@add@to@xml{>\hyxmp@xmlified</}%
    \xdef\hyxmp@xml{\hyxmp@xml#1>^^J}%
  }%
}
```

`\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.

```
\newcommand*{\hyxmp@add@simple@var}[2]{%
  \expandafter\ifx\csname#2\endcsname\relax
  \else
    \hyxmp@xmlify{\csname#2\endcsname}%
    \hyxmp@add@to@xml{%
      \hyxmp@extra@indent______<#1>\hyxmp@xmlified</#1>^^J%
    }%
  \fi
}
```

`\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.

```
\newcommand*{\hyxmp@add@simple@lang}[2]{%
  \@ifnotmtarg{#2}{%
    \hyxmp@xmlify{#2}%
    \expandafter\hyxmp@add@simple@lang@i\hyxmp@xmlified\relax{#1}%
  }%
}
```

`\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.

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

`\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.

```
\def\hyxmp@add@simple@lang@ii[#1]#2\relax#3{%
  \@ifnotmtarg{#2}{%
    \hyxmp@xmlify{#2}%
    \@ifmtarg{#1}{%
      \hyxmp@add@to@xml{%
______<#3>\hyxmp@xmlified</#3>^^J%
      }%
    }{%
      \hyxmp@add@to@xml{%
______<#3 xml:lang="#1">\hyxmp@xmlified</#3>^^J%
```

```
      }%
    }%
  }%
}
```

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`
`\hyxmp@alt@description`
`\hyxmp@alt@rights`

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

```
\def\hyxmp@alt@title{}
\def\hyxmp@alt@description{}
\def\hyxmp@alt@rights{}
```

`\hyxmp@LA@accept` This macro wraps `\define@key` to make the option "`#1`=⟨*value*⟩" append ⟨*value*⟩ to list `#2`.

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

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

Define ⟨*key*⟩=⟨*value*⟩ options for appending to each of the `\hyxmp@alt`⟨*tag*⟩ lists.

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

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

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

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

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

3.4 UUID generation

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

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

```
\def\hyxmp@modulo@a#1{%
  \@tempcntb=\@tempcnta
  \divide\@tempcntb by #1
  \multiply\@tempcntb by #1
  \advance\@tempcnta by -\@tempcntb
}
```

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

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

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

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

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

```
\def\hyxmp@seed@rng@i{%
  \ifx\hyxmp@one@token\@empty
    \let\next=\relax
  \else
    \def\next##1{%
      \multiply\@tempcnta by 3
      \advance\@tempcnta by ‘##1
      \hyxmp@modulo@a{\hyxmp@big@prime}%
      \futurelet\hyxmp@one@token\hyxmp@seed@rng@i
    }%
  \fi
  \next
}
```

`\hyxmp@set@rand@num`
`\hyxmp@rand@num`

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

```
\def\hyxmp@set@rand@num{%
  \@tempcnta=\hyxmp@rand@num
  \multiply\@tempcnta by 3
  \advance\@tempcnta by \hyxmp@big@prime@ii
  \hyxmp@modulo@a{\hyxmp@big@prime}%
  \xdef\hyxmp@rand@num{\the\@tempcnta}%
}
```

`\hyxmp@append@hex`

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

```
\def\hyxmp@append@hex#1{%
  \hyxmp@set@rand@num
  \@tempcnta=\hyxmp@rand@num
  \hyxmp@modulo@a{16}%
  \ifnum\@tempcnta<10
    \xdef#1{#1\the\@tempcnta}%
  \else
```

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

```
    \advance\@tempcnta by -10
    \ifcase\@tempcnta
      \xdef#1{#1a}%
      \or\xdef#1{#1b}%
      \or\xdef#1{#1c}%
      \or\xdef#1{#1d}%
      \or\xdef#1{#1e}%
      \or\xdef#1{#1f}%
    \fi
  \fi
}
```

`\hyxmp@append@hex@iii`

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

```
\def\hyxmp@append@hex@iii#1{%
  \hyxmp@append@hex#1%
  \hyxmp@append@hex#1%
  \hyxmp@append@hex#1%
}
```

`\hyxmp@append@hex@iv`

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

```
\def\hyxmp@append@hex@iv#1{%
  \hyxmp@append@hex@iii#1%
  \hyxmp@append@hex#1%
}
```

`\hyxmp@create@uuid`

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

```
\def\hyxmp@create@uuid#1{%
  \def#1{uuid:}%
  \hyxmp@append@hex@iv#1%
  \hyxmp@append@hex@iv#1%
  \g@addto@macro#1{-}%
  \hyxmp@append@hex@iv#1%
  \g@addto@macro#1{-4}%
  \hyxmp@append@hex@iii#1%
  \g@addto@macro#1{-}%
```

Randomly select one of "`8`", "`9`", "`a`", or "`b`".

```
  \hyxmp@set@rand@num
  \@tempcnta=\hyxmp@rand@num
  \hyxmp@modulo@a{4}%
  \ifcase\@tempcnta
    \g@addto@macro#1{8}%
    \or\g@addto@macro#1{9}%
    \or\g@addto@macro#1{a}%
    \or\g@addto@macro#1{b}%
  \fi
  \hyxmp@append@hex@iii#1%
  \g@addto@macro#1{-}%
  \hyxmp@append@hex@iv#1%
  \hyxmp@append@hex@iv#1%
  \hyxmp@append@hex@iv#1%
}
```

`\hyxmp@def@DocumentID`
`\hyxmp@DocumentID`
`\hyxmp@seed@string`

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.

```
\newcommand*{\hyxmp@def@DocumentID}{%
  \edef\hyxmp@seed@string{\jobname:\@pdftitle:\@pdfauthor:}%
  \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
  \edef\hyxmp@rand@num{\the\@tempcnta}%
  \hyxmp@create@uuid\hyxmp@DocumentID
}
```

`\hyxmp@def@InstanceID`
`\hyxmp@InstanceID`
`\hyxmp@seed@string`

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

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

```
  \hyxmp@today@xmp@define{\hyxmp@seed@string}%
  \edef\hyxmp@seed@string{%
    \jobname:\@pdftitle:\@pdfauthor:\hyxmp@today@xmp:\hyxmp@seed@string
  }%
  \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
  \edef\hyxmp@rand@num{\the\@tempcnta}%
  \hyxmp@create@uuid\hyxmp@InstanceID
}
```

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.

```
\newcommand*{\hyxmp@add@to@xml}[1]{%
  \bgroup
    \@tempcnta=0
    \ifhyxmp@unicodetex
      \@tempcntb=65536%
    \else
      \@tempcntb=256%
    \fi
    \loop
      \lccode\@tempcnta=\@tempcnta
      \advance\@tempcnta by 1
      \ifnum\@tempcnta<\@tempcntb
    \repeat
    \lccode`\_=`\ \relax
    \lccode`\^^C=`\,\relax
    \lccode`\^^U=`\_\relax
    \lowercase{\xdef\hyxmp@new@xml{#1}}%
    \xdef\hyxmp@xml{\hyxmp@xml\hyxmp@new@xml}%
  \egroup
}
```

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

```
\bgroup
\catcode`\#=11
\gdef\hyxmp@hash{#}
\egroup
```

`\hyxmp@padding`
`\hyxmp@xml`

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

```
\bgroup
  \xdef\hyxmp@xml{}%
  \hyxmp@add@to@xml{%
_______________________________________________________________^^J%
  }
  \xdef\hyxmp@padding{\hyxmp@xml}%
\egroup
\xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
\xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
\xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
\xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
\xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
```

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

```
\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`
`\hyxmp@define@pdfproducer`

Define `\@pdfproducer` using the banner string if available or the TeX engine's version number if not.

```
\newcommand*{\hyxmp@define@pdfproducer}{%
  \gdef\@pdfproducer{TeX}
  \ifPDFTeX
    \expandafter\hyxmp@banner@to@producer\expandafter{\pdftexbanner}
  \else
    \ifLuaTeX
      \expandafter\hyxmp@banner@to@producer\expandafter{\luatexbanner}
    \else
      \ifXeTeX
        \edef\@pdfproducer{XeTeX version \the\XeTeXversion\XeTeXrevision}
      \fi
    \fi
  \fi
}
```

`\@pdfproducer`
`\hyxmp@banner@to@producer`

Define `\@pdfproducer` as the TeX engine's banner string (e.g., "`This is pdfTeX, Version 3.14159265-2.6-1.40.20 (TeX Live 2019) kpathsea version 6.3.1`"), removing the initial "`This is`" if possible (specifically, when ε-TeX's `\scantokens` primitive is available).

```
\def\hyxmp@banner@to@producer#1{%
  \ifx\scantokens\relax
    \gdef\@pdfproducer{#1}%
  \else
    \scantokens{\makeatletter\hyxmp@remove@this#1\relax\makeatother}%
  \fi
}
```

`\@pdfproducer`
`\hyxmp@remove@this`

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

```
\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.

```
\AtBeginDocument{%
  \ifx\@pdfproducer\relax
    \hyxmp@define@pdfproducer
  \fi
}
```

`\hyxmp@pdf@schema`

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

```
\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.

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

Specify the PDF version.

```
  \@ifundefined{pdfvariable}{%
    \@ifundefined{pdfminorversion}{%
```

Case 1: Neither `\pdfvariable` nor `\pdfminorversion` is defined (XƎLaTeX and regular LaTeX).

```
    }{%
```

Case 2: `\pdfminorversion` is defined (pdfLaTeX and pre-0.85 LuaLaTeX).

```
      \hyxmp@add@simple{pdf:PDFVersion}{1.\the\pdfminorversion}%
    }%
  }{%
```

Case 3: `\pdfvariable` is defined (LuaLaTeX 0.85+).

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

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.

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

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

```
  \@ifmtargexp{#3}{\@tempswafalse}{\@tempswatrue}%
  #1
    \@tempswatrue
  \fi
```

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

```
  \if@tempswa
    \hyxmp@xmlify{#3}%
```

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

```
    \let\hyxmp@value=\hyxmp@xmlified
    \hyxmp@add@to@xml{%
␣␣␣␣␣␣<dc:#2>^^J%
␣␣␣␣␣␣␣␣<rdf:Alt>^^J%
    }%
    \ifx\@pdfmetalang\hyxmp@x@default
    \else
      \hyxmp@xmlify{\@pdfmetalang}%
      \hyxmp@add@to@xml{%
␣␣␣␣␣␣␣␣␣␣<rdf:li xml:lang="\hyxmp@xmlified">\hyxmp@value</rdf:li>^^J%
      }%
    \fi
    \hyxmp@add@to@xml{%
␣␣␣␣␣␣␣␣␣␣<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@value</rdf:li>^^J%
    }%
```

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

```
    \def\do##1##2{
      \hyxmp@xmlify{##2}%
      \hyxmp@add@to@xml{%
␣␣␣␣␣␣␣␣␣␣<rdf:li xml:lang="##1">\hyxmp@xmlified</rdf:li>^^J%
      }%
    }%
    \csname hyxmp@alt@#2\endcsname
```

Complete this XMP element.

```
    \hyxmp@add@to@xml{%
␣␣␣␣␣␣␣␣</rdf:Alt>^^J%
␣␣␣␣␣␣</dc:#2>^^J%
    }%
  \fi
}%
```

`\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.

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

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

```
  \@ifmtargexp{#4}{\@tempswafalse}{\@tempswatrue}%
  #1
    \@tempswatrue
  \fi
```

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

```
  \if@tempswa
    \hyxmp@add@to@xml{%
␣␣␣␣␣␣<dc:#2>^^J%
␣␣␣␣␣␣␣␣<rdf:#3>^^J%
    }%
    \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`.

```
      \hyxmp@xmlify{#4}%
      \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
      \def\@elt##1{%
        \hyxmp@add@to@xml{%
␣␣␣␣␣␣␣␣␣␣<rdf:li>##1</rdf:li>^^J%
        }%
      }%
      \hyxmp@list
    \egroup
    \hyxmp@add@to@xml{%
________</rdf:#3>^^J%
______</dc:#2>^^J%
    }%
  \fi
}
```

`\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.

```
\newcommand{\hyxmp@singleton@dc}[3][Bag]{%
  \@ifnotmtarg{#3}{%
    \hyxmp@xmlify{#3}%
    \hyxmp@add@to@xml{%
______<dc:#2>^^J%
________<rdf:#1>^^J%
__________<rdf:li>\hyxmp@xmlified</rdf:li>^^J%
________</rdf:#1>^^J%
______</dc:#2>^^J%
    }%
  }
}
```

`\hyxmp@dc@schema` Add properties defined by the Dublin Core schema to the `\hyxmp@xml` macro. Specifically, we add entries for the dc:title property if the author specified a pdftitle, the dc:description property if the author specified a pdfsubject, the dc:rights property if the author specified a pdfcopyright, the dc:creator property if the author specified a pdfauthor, the dc:subject property if the author specified pdfkeywords, and the dc:language property if the author specified pdflang. We also specify the dc:date property using the date the document was run through LaTeX and the dc:source property using the base name of the source file with `.tex` appended.

```
\newcommand*{\hyxmp@dc@schema}{%
  \hyxmp@add@simple{dc:format}{application/pdf}%
  \hyxmp@rdf@dc[\ifHy@pdfa]{title}{\@pdftitle}%
  \hyxmp@rdf@dc[\ifHy@pdfa]{description}{\@pdfsubject}%
  \hyxmp@rdf@dc{rights}{\@pdfcopyright}%
  \hyxmp@singleton@dc{publisher}{\@pdfpublisher}%
  \hyxmp@singleton@dc[Seq]{date}{\hyxmp@today@xmp}%
  \hyxmp@singleton@dc{language}{\@pdflang}%
  \hyxmp@singleton@dc{type}{\@pdftype}%
  \hyxmp@list@to@xml[\ifHy@pdfa]{creator}{Seq}{\hyxmp@pdfauthor}%
  \hyxmp@list@to@xml{subject}{Bag}{\hyxmp@pdfkeywords}%
  \ifx\@pdfsource\@empty
  \else
    \hyxmp@add@simple{dc:source}{\@pdfsource}%
  \fi
}
```

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.

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

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

```
  \let\hyxmp@rights=\@empty
  \ifx\@pdflicenseurl\@empty
  \else
    \def\hyxmp@rights{YES}%
  \fi
  \ifx\@pdfcopyright\@empty
  \else
    \def\hyxmp@rights{YES}%
  \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.

```
  \ifx\hyxmp@rights\@empty
  \else
    \ifx\@pdfcopyright\@empty
    \else
      \hyxmp@add@simple{xmpRights:Marked}{True}%
    \fi
    \hyxmp@add@simple{xmpRights:WebStatement}{\@pdflicenseurl}%
  \fi
}
```

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.

```
\gdef\hyxmp@mm@schema{%
  \@ifmtargexp{\hyxmp@DocumentID}{\hyxmp@def@DocumentID}{}%
  \@ifmtargexp{\hyxmp@InstanceID}{\hyxmp@def@InstanceID}{}%
  \hyxmp@add@simple{xmpMM:DocumentID}{\hyxmp@DocumentID}%
  \hyxmp@add@simple{xmpMM:InstanceID}{\hyxmp@InstanceID}%
  \hyxmp@add@simple{xmpMM:VersionID}{\@pdfversionid}%
  \hyxmp@add@simple{xmpMM:RenditionClass}{\@pdfrendition}%
}
```

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.

```
\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`.

```
  \@ifmtargexp{\@pdfcreationdate}{%
    \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@today@xmp}%
  }{%
    \hyxmp@add@simple{xmp:CreateDate}{%
      \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfcreationdate}}%
  }%
```

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

```
  \@ifmtargexp{\@pdfmoddate}{%
    \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@today@xmp}%
  }{%
    \hyxmp@add@simple{xmp:ModifyDate}{%
      \expandafter\hyxmp@as@xmp@date\expandafter{\@pdfmoddate}}%
  }%
```

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

```
  \@ifmtargexp{\@pdfmetadatetime}{%
    \hyxmp@add@simple{xmp:MetadataDate}{\hyxmp@today@xmp}%
  }{%
    \hyxmp@add@simple{xmp:MetadataDate}{\@pdfmetadatetime}%
  }%
```

Define the creation tool and the base URL.

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

3.5.7 The Photoshop schema

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

```
\gdef\hyxmp@photoshop@schema{%
  \edef\hyxmp@photoshop@data{\@pdfauthortitle\@pdfcaptionwriter}%
  \hyxmp@add@simple{photoshop:AuthorsPosition}{\@pdfauthortitle}%
  \hyxmp@add@simple{photoshop:CaptionWriter}{\@pdfcaptionwriter}%
}
```

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.

```
\newcommand*{\hyxmp@pdfa@id@schema}{%
  \ifHy@pdfa
    \hyxmp@add@simple{pdfaid:part}{\@pdfapart}%
    \hyxmp@add@simple{pdfaid:conformance}{\@pdfaconformance}%
  \fi
}
```

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

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

`\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.

```
\newcommand*{\hyxmp@pdfx@id@schema}{%
  \@tempcnta=0\hyxmp@pdfx@major\relax
  \ifnum\@tempcnta=0
  \else
    \ifnum\@tempcnta=1
      \hyxmp@add@simple{pdfx:GTS_PDFXVersion}{PDF/X-1:2001}%
      \hyxmp@add@simple{pdfx:GTS_PDFXConformance}{\@pdfxstandard}%
    \else
      \ifnum\@tempcnta<4
        \hyxmp@add@simple{pdfx:GTS_PDFXVersion}{\@pdfxstandard}%
      \else
        \hyxmp@add@simple{pdfxid:GTS_PDFXVersion}{\@pdfxstandard}%
      \fi
    \fi
  \fi
}
```

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.

```
\begingroup
  \catcode`\&=12
  \catcode`\#=12
  \gdef\xmplinesep{&#xA;}
\endgroup
```

`\hyxmp@list@to@lines` Given a property (`#1`) and a macro containing a comma-separated list (`#2`), replace commas with `\xmplinesep`. Do nothing it the list is empty.

```
\newcommand*{\hyxmp@list@to@lines}[2]{%
  \@ifnotmtargexp{#2}{%
    \bgroup
      \hyxmp@add@to@xml{%
        \hyxmp@extra@indent______<#1>%
      }%
```

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

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

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

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

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

```
      \let\@elt=\@elt@first
      \hyxmp@xmlify{#2}%
      \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
      \hyxmp@list
      \hyxmp@add@to@xml{</#1>^^J}%
    \egroup
  }%
}
```

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

```
\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.

```
  \ifx\hyxmp@iptc@data\@empty
  \else
    \hyxmp@add@to@xml{%
______<Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">^^J%
    }%
```

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

```
    \bgroup
```

```
      \edef\hyxmp@extra@indent{\hyxmp@extra@indent\space\space}%
      \hyxmp@list@to@lines{Iptc4xmpCore:CiAdrExtadr}{\@pdfcontactaddress}%
      \hyxmp@add@simple{Iptc4xmpCore:CiAdrCity}{\@pdfcontactcity}%
      \hyxmp@add@simple{Iptc4xmpCore:CiAdrRegion}{\@pdfcontactregion}%
      \hyxmp@add@simple{Iptc4xmpCore:CiAdrPcode}{\@pdfcontactpostcode}%
      \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.

```
      \def\xmplinesep{,}%
      \hyxmp@list@to@lines{Iptc4xmpCore:CiTelWork}{\@pdfcontactphone}%
      \hyxmp@list@to@lines{Iptc4xmpCore:CiEmailWork}{\@pdfcontactemail}%
      \hyxmp@list@to@lines{Iptc4xmpCore:CiUrlWork}{\@pdfcontacturl}%
    \egroup
    \hyxmp@add@to@xml{%
␣␣␣␣␣␣</Iptc4xmpCore:CreatorContactInfo>^^J%
    }%
  \fi
}
```

3.5.10 The PRISM Basic Metadata schema

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

```
\newcommand*{\hyxmp@prism@schema}{%
  \ifx\hyxmp@prism@data\@empty
  \else
    \hyxmp@add@simple{prism:complianceProfile}{three}%
  \fi
  \hyxmp@add@simple@lang{prism:subtitle}{\@pdfsubtitle}%
  \hyxmp@add@simple@lang{prism:publicationName}{\@pdfpublication}%
  \hyxmp@add@simple{prism:aggregationType}{\@pdfpubtype}%
  \hyxmp@add@simple@lang{prism:bookEdition}{\@pdfbookedition}%
  \hyxmp@add@simple{prism:volume}{\@pdfvolumenum}%
  \hyxmp@add@simple{prism:number}{\@pdfissuenum}%
  \hyxmp@add@simple{prism:pageRange}{\@pdfpagerange}%
  \hyxmp@add@simple{prism:isbn}{\@pdfisbn}%
  \hyxmp@add@simple{prism:issn}{\@pdfissn}%
  \hyxmp@add@simple{prism:eIssn}{\@pdfeissn}%
  \hyxmp@add@simple{prism:doi}{\@pdfdoi}%
  \hyxmp@add@simple{prism:url}{\@pdfurl}%
  \hyxmp@add@simple{prism:byteCount}{\@pdfbytes}%
  \hyxmp@add@simple{prism:pageCount}{\@pdfnumpages}%
}
```

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.

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

`\hyxmp@iptc@data`

```
  \edef\hyxmp@iptc@data{%
    \@pdfcontactaddress
    \@pdfcontactcity
    \@pdfcontactregion
    \@pdfcontactpostcode
    \@pdfcontactcountry
    \@pdfcontactphone
    \@pdfcontactemail
    \@pdfcontacturl
  }%
}%
```

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

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

`\hyxmp@prism@data`

```
  \edef\hyxmp@prism@data{%
    \@pdfbookedition
    \@pdfbytes
    \@pdfdoi
    \@pdfeissn
    \@pdfisbn
    \@pdfissn
    \@pdfissuenum
    \@pdfnumpages
    \@pdfpagerange
    \@pdfpublication
    \@pdfpubtype
    \@pdfsubtitle
    \@pdfurl
    \@pdfvolumenum
  }%
}%
```

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

```
\newcommand*{\hyxmp@begin@extension@decls}{%
  \hyxmp@add@to@xml{%
______<pdfaExtension:schemas>^^J%
________<rdf:Bag>^^J%
  }%
}
```

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

```
\newcommand*{\hyxmp@end@extension@decls}{%
  \hyxmp@add@to@xml{%
________</rdf:Bag>^^J%
______</pdfaExtension:schemas>^^J%
  }%
}
```

`\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.

```
\newcommand*{\hyxmp@begin@ext@decl}[3]{%
  \hyxmp@add@to@xml{%
__________<rdf:li rdf:parseType="Resource">^^J%
____________<pdfaSchema:schema>#1</pdfaSchema:schema>^^J%
____________<pdfaSchema:prefix>#2</pdfaSchema:prefix>^^J%
____________<pdfaSchema:namespaceURI>#3</pdfaSchema:namespaceURI>^^J%
____________<pdfaSchema:property>^^J%
______________<rdf:Seq>^^J%
  }%
}%
```

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

```
\newcommand*{\hyxmp@end@ext@decl}{%
  \hyxmp@add@to@xml{%
______________</rdf:Seq>^^J%
____________</pdfaSchema:property>^^J%
__________</rdf:li>^^J%
  }%
}%
```

`\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.

```
\newcommand{\hyxmp@declare@property}[4][Text]{%
  \hyxmp@add@to@xml{%
________________<rdf:li rdf:parseType="Resource">^^J%
__________________<pdfaProperty:name>}%
  \xdef\hyxmp@xml{\hyxmp@xml#2}%
  \hyxmp@add@to@xml{</pdfaProperty:name>^^J%
__________________<pdfaProperty:valueType>#1</pdfaProperty:valueType>^^J%
_________________<pdfaProperty:category>#3</pdfaProperty:category>^^J%
_________________<pdfaProperty:description>#4</pdfaProperty:description>^^J%
_______________</rdf:li>^^J%
  }%
}%
```

`\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.

```
\newcommand{\hyxmp@declare@field}[3][Text]{%
  \hyxmp@add@to@xml{%
____________________<rdf:li rdf:parseType="Resource">^^J%
______________________<pdfaField:name>#2</pdfaField:name>^^J%
______________________<pdfaField:valueType>#1</pdfaField:valueType>^^J%
______________________<pdfaField:description>#3</pdfaField:description>^^J%
____________________</rdf:li>^^J%
  }%
}
```

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

```
\newcommand*{\hyxmp@pdf@extensions}{%
  \hyxmp@begin@ext@decl
      {Adobe PDF Schema}%
      {pdf}%
      {http://ns.adobe.com/pdf/1.3/}%
  \hyxmp@declare@property
      {Trapped}%
      {internal}%
      {Indication if the document has been modified to include trapping information}%
  \hyxmp@end@ext@decl
}%
```

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

```
\newcommand*{\hyxmp@mm@extensions}{%
  \hyxmp@begin@ext@decl
      {XMP Media Management Schema}%
      {xmpMM}%
      {http://ns.adobe.com/xap/1.0/mm/}%
  \hyxmp@declare@property
      [URI]
      {DocumentID}%
      {internal}%
      {UUID based identifier for all versions and renditions of a document}%
  \hyxmp@declare@property
      [URI]
      {InstanceID}%
      {internal}%
      {UUID based identifier for specific incarnation of a document}%
  \hyxmp@declare@property
```

```
      {VersionID}%
      {internal}%
      {Document version identifier}%
  \hyxmp@declare@property
      {RenditionClass}%
      {internal}%
      {The manner in which a document is rendered}%
  \hyxmp@end@ext@decl
}%
```

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

```
\newcommand*{\hyxmp@pdfa@id@extensions}{%
  \hyxmp@begin@ext@decl
      {PDF/A Identification Schema}%
      {pdfaid}%
      {http://www.aiim.org/pdfa/ns/id/}%
  \hyxmp@declare@property
      [Integer]%
      {part}%
      {internal}%
      {Part of PDF/A standard}%
  \hyxmp@declare@property
      {conformance}%
      {internal}%
      {Conformance level of PDF/A standard}%
  \hyxmp@end@ext@decl
}%
```

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

```
\newcommand*{\hyxmp@pdfua@id@extensions}{%
  \hyxmp@begin@ext@decl
      {PDF/UA Universal Accessibility Schema}%
      {pdfuaid}%
      {http://www.aiim.org/pdfua/ns/id/}%
  \hyxmp@declare@property
      [Integer]%
      {part}%
      {internal}%
      {Part of ISO 14289 standard}%
  \hyxmp@end@ext@decl
}%
```

`\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.

```
\newcommand*{\hyxmp@pdfx@id@extensions}{%
  \ifx\hyxmp@pdfx@major\empty
  \else
    \hyxmp@begin@ext@decl
        {Adobe Document Info PDF/X eXtension Schema}%
        {pdfx}%
        {http://ns.adobe.com/pdfx/1.3/}%
    \hyxmp@declare@property
        {GTS_PDFXVersion}%
        {internal}%
        {ID of PDF/X standard}%
    \hyxmp@declare@property
        {GTS_PDFXConformance}%
        {internal}%
        {Conformance level of PDF/X standard}%
    \hyxmp@end@ext@decl
  \fi
```

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

```
  \@tempcnta=0\hyxmp@pdfx@major\relax
  \ifnum\@tempcnta>3
    \hyxmp@begin@ext@decl
        {PDF/X ID Schema}%
        {pdfxid}%
        {http://www.npes.org/pdfx/ns/id/}%
    \hyxmp@declare@property
        {GTS_PDFXVersion}%
        {internal}%
        {ID of PDF/X standard}%
    \hyxmp@end@ext@decl
  \fi
}%
```

`\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.

```
\newcommand*{\hyxmp@iptc@extensions}{%
  \hyxmp@begin@ext@decl
      {IPTC Core Schema}%
      {Iptc4xmpCore}%
      {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}%
  \hyxmp@declare@property
      [ContactInfo]
      {CreatorContactInfo}
      {external}
      {Document creator's contact information}
```

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

```
  \hyxmp@add@to@xml{%
______________</rdf:Seq>^^J%
____________</pdfaSchema:property>^^J%
____________<pdfaSchema:valueType>^^J%
______________<rdf:Seq>^^J%
________________<rdf:li rdf:parseType="Resource">^^J%
__________________<pdfaType:type>ContactInfo</pdfaType:type>^^J%
__________________<pdfaType:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaType:n
__________________<pdfaType:prefix>Iptc4xmpCore</pdfaType:prefix>^^J%
__________________<pdfaType:description>%
                   Basic set of information to get in contact with a person%
                 </pdfaType:description>^^J%
__________________<pdfaType:field>^^J%
____________________<rdf:Seq>^^J%
  }%
  \hyxmp@declare@field
      {CiAdrCity}%
      {Contact information city}%
  \hyxmp@declare@field
      {CiAdrCtry}%
      {Contact information country}%
  \hyxmp@declare@field
      {CiAdrExtadr}%
      {Contact information address}%
  \hyxmp@declare@field
      {CiAdrPcode}%
      {Contact information local postal code}%
  \hyxmp@declare@field
      {CiAdrRegion}%
      {Contact information regional information such as state or province}%
  \hyxmp@declare@field
      {CiEmailWork}%
      {Contact information email address(es)}%
  \hyxmp@declare@field
      {CiTelWork}%
      {Contact information telephone number(s)}%
  \hyxmp@declare@field
      {CiUrlWork}%
      {Contact information Web URL(s)}%
  \hyxmp@add@to@xml{%
____________________</rdf:Seq>^^J%
__________________</pdfaType:field>^^J%
________________</rdf:li>^^J%
______________</rdf:Seq>^^J%
____________</pdfaSchema:valueType>^^J%
__________</rdf:li>^^J%
  }%
}
```

`\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.

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

```
    {PRISM Basic Metadata}%
    {prism}%
    {http://prismstandard.org/namespaces/basic/2.1/}%
\hyxmp@declare@property
    {complianceProfile}%
    {internal}%
    {PRISM specification compliance profile to which this document adheres}%
\hyxmp@declare@property
    {publicationName}%
    {external}%
    {Publication name}%
\hyxmp@declare@property
    {aggregationType}%
    {external}%
    {Publication type}%
\hyxmp@declare@property
    {bookEdition}%
    {external}%
    {Edition of the book in which the document was published}%
\hyxmp@declare@property
    {volume}%
    {external}%
    {Publication volume number}%
\hyxmp@declare@property
    {number}%
    {external}%
    {Publication issue number within a volume}%
\hyxmp@declare@property
    {pageRange}%
    {external}%
    {Page range for the document within the print version of its publication}%
\hyxmp@declare@property
    {issn}%
    {external}%
    {ISSN for the printed publication in which the document was published}%
\hyxmp@declare@property
    {eIssn}%
    {external}%
    {ISSN for the electronic publication in which the document was published}%
\hyxmp@declare@property
    {isbn}%
    {external}%
    {ISBN for the publication in which the document was published}%
\hyxmp@declare@property
    {doi}%
    {external}%
    {Digital Object Identifier for the document}%
\hyxmp@declare@property
    [URL]
    {url}%
    {external}%
    {URL at which the document can be found}%
  \hyxmp@declare@property
    [Integer]
    {byteCount}%
    {internal}%
    {Approximate file size in octets}%
  \hyxmp@declare@property
    [Integer]
    {pageCount}%
    {internal}%
    {Number of pages in the print version of the document}%
  \hyxmp@declare@property
    {subtitle}%
    {external}%
    {Document's subtitle}%
  \hyxmp@end@ext@decl
}%
```

`\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.

```
\newcommand*{\hyxmp@declare@extensions}{%
  \hyxmp@begin@extension@decls
```

Declare the Adobe PDF schema (always present).

```
  \hyxmp@pdf@extensions
```

Declare the XMP Media Management extensions (always present).

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

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

```
  \ifHy@pdfa
    \hyxmp@pdfa@id@extensions
  \fi
```

Conditionally declare the PDF/UA Universal Accessibility extensions.

```
  \ifx\@pdfuapart\@empty
  \else
    \hyxmp@pdfua@id@extensions
  \fi
```

`\next` Conditionally declare the PDF/X extensions.

```
  \ifx\@pdfxversion\@empty
  \else
    \hyxmp@pdfx@id@extensions
  \fi
```

Conditionally declare IPTC photo metadata extensions.

```
  \ifx\hyxmp@iptc@data\@empty
  \else
    \hyxmp@iptc@extensions
  \fi
```

Conditionally declare PRISM basic metadata extensions.

```
  \ifx\hyxmp@prism@data\@empty
  \else
    \hyxmp@prism@extensions
  \fi
```

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

3.5.12 Combining schemata into an XMP packet

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

```
\begingroup
  \ifhyxmp@unicodetex
    \lccode`\!="FEFF %
    \lowercase{%
      \gdef\hyxmp@bom{!}
    }%
  \else
    \catcode`\^^ef=12
    \catcode`\^^bb=12
    \catcode`\^^bf=12
    \gdef\hyxmp@bom{^^ef^^bb^^bf}%
  \fi
\endgroup
```

`\hyxmp@construct@packet` `\hyxmp@xml` Successively add XML data to `\hyxmp@xml` until we have something we can insert into the document's PDF catalog.

```
\def\hyxmp@construct@packet{%
  \gdef\hyxmp@xml{}%
  \hyxmp@add@to@xml{<?xpacket begin="\hyxmp@bom" %
id="W5M0MpCehiHzreSzNTczkc9d"?>^^J%
<x:xmpmeta xmlns:x="adobe:ns:meta/">^^J%
__<rdf:RDF %
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\hyxmp@hash">^^J%
____<rdf:Description rdf:about=""^^J%
```

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

```
_____________________xmlns:pdf="http://ns.adobe.com/pdf/1.3/"^^J%
_____________________xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/"^^J%
_____________________xmlns:dc="http://purl.org/dc/elements/1.1/"^^J%
_____________________xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/"^^J%
_____________________xmlns:xmp="http://ns.adobe.com/xap/1.0/"^^J%
_____________________xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"^^J%
_____________________xmlns:stEvt="http://ns.adobe.com/xap/1.0/sType/ResourceEvent\hyxmp@hash"^^
_____________________xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/"^^J%
```

```
_____________________xmlns:pdfuaid="http://www.aiim.org/pdfua/ns/id/"^^J%
_____________________xmlns:pdfx="http://ns.adobe.com/pdfx/1.3/"^^J%
_____________________xmlns:pdfxid="http://www.npes.org/pdfx/ns/id/"^^J%
_____________________xmlns:prism="http://prismstandard.org/namespaces/basic/2.1/"^^J%
_____________________xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"^^J%
_____________________xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"^^J%
_____________________xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema\hyxmp@hash"^^J%
_____________________xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property\hyxmp@hash"^^J%
_____________________xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type\hyxmp@hash"^^J%
_____________________xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field\hyxmp@hash">^^J%
  }%
```

Declare non-standard schemata.

```
  \hyxmp@check@iptc@data
  \hyxmp@check@prism@data
  \hyxmp@declare@extensions
```

Insert all the metadata we know how to insert.

```
  \hyxmp@pdf@schema
  \hyxmp@xmpRights@schema
  \hyxmp@dc@schema
  \hyxmp@photoshop@schema
  \hyxmp@xmp@basic@schema
  \hyxmp@pdfa@id@schema
  \hyxmp@pdfua@id@schema
  \hyxmp@pdfx@id@schema
  \hyxmp@mm@schema
  \hyxmp@iptc@schema
  \hyxmp@prism@schema
  \hyxmp@add@to@xml{%
____</rdf:Description>^^J%
__</rdf:RDF>^^J%
</x:xmpmeta>^^J%
\hyxmp@padding
<?xpacket end="w"?>^^J%
  }%
}
```

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`
`\hyxmp@driver`

Determine which hyperref driver is in use and invoke the appropriate embedding function.

```
\newcommand*{\hyxmp@embed@packet}{%
  \hyxmp@construct@packet
  \def\hyxmp@driver{hpdftex}%
  \ifx\hyxmp@driver\Hy@driver
```

```
    \hyxmp@embed@packet@pdftex
  \else
    \def\hyxmp@driver{hluatex}%
    \ifx\hyxmp@driver\Hy@driver
      \hyxmp@embed@packet@luatex
    \else
      \def\hyxmp@driver{hdvipdfm}%
      \ifx\hyxmp@driver\Hy@driver
        \hyxmp@embed@packet@dvipdfm
      \else
        \def\hyxmp@driver{hxetex}%
        \ifx\hyxmp@driver\Hy@driver
          \hyxmp@embed@packet@xetex
        \else
          \@ifundefined{pdfmark}{%
            \PackageWarningNoLine{hyperxmp}{%
              Unrecognized hyperref driver `\Hy@driver'.\MessageBreak
              \jobname.tex's XMP metadata will *not* be\MessageBreak
              embedded in the resulting file}%
          }{%
            \hyxmp@embed@packet@pdfmark
          }%
        \fi
      \fi
    \fi
  \fi
}
```

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.

```
\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.

```
\newcommand*{\hyxmp@embed@packet@pdftex}{%
  \bgroup
    \ifluatex
    \else
      \pdfcompresslevel=0
    \fi
    \immediate\pdfobj \ifluatex uncompressed\fi stream attr {%
      /Type /Metadata
      /Subtype /XML
    }{\hyxmp@xml}%
    \pdfcatalog {/Metadata \the\pdflastobj\space 0 R}%
  \egroup
}
```

3.6.2 Embedding using LuaTeX 0.85+

`\hyxmp@embed@packet@luatex`

Embed the XMP packet using LuaTeX 0.85+ primitives.

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

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.

```
\newcommand*{\hyxmp@embed@packet@pdfmark}{%
  \pdfmark{%
    pdfmark=/NamespacePush
  }%
  \pdfmark{%
    pdfmark=/OBJ,
    Raw={/_objdef \string{hyxmp@Metadata\string} /type /stream}%
  }%
  \pdfmark{%
    pdfmark=/PUT,
    Raw={\string{hyxmp@Metadata\string}
      2 dict begin
        /Type /Metadata def
        /Subtype /XML def
        currentdict
      end
    }%
  }%
  \pdfmark{%
```

```
    pdfmark=/PUT,
    Raw={\string{hyxmp@Metadata\string} (\hyxmp@xml)}%
  }%
  \pdfmark{%
    pdfmark=/Metadata,
    Raw={\string{Catalog\string} \string{hyxmp@Metadata\string}}%
  }%
  \pdfmark{%
    pdfmark=/NamespacePop
  }%
}
```

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.

```
\newcommand*{\hyxmp@embed@packet@dvipdfm}{%
  \hyxmp@string@len{\hyxmp@xml}%
  \special{pdf: object @hyxmp@Metadata
    <<
      /Type /Metadata
      /Subtype /XML
      /Length \the\@tempcnta
    >>
    stream^^J\hyxmp@xml endstream%
  }%
  \special{pdf: docview
    <<
      /Metadata @hyxmp@Metadata
    >>
  }%
}
```

`\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.

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

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

```
\def\hyxmp@count@spaces#1 {%
  \def\hyxmp@one@token{#1}%
  \ifx\hyxmp@one@token\@empty
    \advance\@tempcnta by -1
  \else
    \advance\@tempcnta by 1
    \expandafter\hyxmp@count@spaces
  \fi
}
```

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

```
\newcommand*{\hyxmp@count@non@spaces}[1]{%
  \def\hyxmp@one@token{#1}%
  \ifx\hyxmp@one@token\@empty
  \else
    \advance\@tempcnta by 1
    \expandafter\hyxmp@count@non@spaces
  \fi
}
```

3.6.5 Embedding using XƎTeX

`\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.

```
\newcommand*{\hyxmp@embed@packet@xetex}{%
  \special{pdf:stream @hyxmp@Metadata (\hyxmp@xml)
    <<
      /Type /Metadata
      /Subtype /XML
    >>
  }%
  \special{pdf:put @catalog
    <<
      /Metadata @hyxmp@Metadata
    >>
  }%
}
```

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.

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

4 Help Wanted

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

A Sample XMP Packet

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

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

⋮

[over 200 lines of boilerplate definitions not shown]

⋮

```
  </rdf:Bag>
</pdfaExtension:schemas>
<pdf:Keywords>
  energy quanta, Hertz effect, quantum physics
</pdf:Keywords>
<pdf:Producer>
  pdfTeX, Version 3.14159265-2.6-1.40.20 (TeX Live 2019/Debian)
</pdf:Producer>
<pdf:PDFVersion>1.5</pdf:PDFVersion>
<xmpRights:Marked>True</xmpRights:Marked>
<xmpRights:WebStatement>
  http://creativecommons.org/licenses/by-nc-nd/3.0/
</xmpRights:WebStatement>
<dc:format>application/pdf</dc:format>
<dc:title>
  <rdf:Alt>
    <rdf:li xml:lang="en">
      On a heuristic viewpoint concerning the production
      and transformation of light
    </rdf:li>
    <rdf:li xml:lang="x-default">
      On a heuristic viewpoint concerning the production
      and transformation of light
    </rdf:li>
    <rdf:li xml:lang="de">
      Über einen die Erzeugung und Verwandlung des Lichtes
      betreffenden heuristischen Gesichtspunkt
    </rdf:li>
  </rdf:Alt>
</dc:title>
<dc:description>
  <rdf:Alt>
    <rdf:li xml:lang="en">photoelectric effect</rdf:li>
    <rdf:li xml:lang="x-default">photoelectric effect</rdf:li>
  </rdf:Alt>
</dc:description>
<dc:rights>
  <rdf:Alt>
    <rdf:li xml:lang="en">
      Copyright (C) 1905, Albert Einstein
    </rdf:li>
    <rdf:li xml:lang="x-default">
      Copyright (C) 1905, Albert Einstein
    </rdf:li>
  </rdf:Alt>
</dc:rights>
```

```
<dc:publisher>
  <rdf:Bag>
    <rdf:li>Wiley-VCH</rdf:li>
  </rdf:Bag>
</dc:publisher>
<dc:creator>
  <rdf:Seq>
    <rdf:li>Albert Einstein</rdf:li>
  </rdf:Seq>
</dc:creator>
<dc:subject>
  <rdf:Bag>
    <rdf:li>energy quanta</rdf:li>
    <rdf:li>Hertz effect</rdf:li>
    <rdf:li>quantum physics</rdf:li>
  </rdf:Bag>
</dc:subject>
<dc:date>
  <rdf:Seq>
    <rdf:li>1905-03-17</rdf:li>
  </rdf:Seq>
</dc:date>
<dc:language>
  <rdf:Bag>
    <rdf:li>en</rdf:li>
  </rdf:Bag>
</dc:language>
<dc:type>
  <rdf:Bag>
    <rdf:li>Text</rdf:li>
  </rdf:Bag>
</dc:type>
<dc:source>einstein.tex</dc:source>
<photoshop:AuthorsPosition>
  Technical Assistant, Level III
</photoshop:AuthorsPosition>
<photoshop:CaptionWriter>Scott Pakin</photoshop:CaptionWriter>
<xmp:CreateDate>2019-03-16T23:07:38-06:00</xmp:CreateDate>
<xmp:ModifyDate>2019-03-16T23:07:38-06:00</xmp:ModifyDate>
<xmp:MetadataDate>2019-03-16T23:07:38-06:00</xmp:MetadataDate>
<xmp:CreatorTool>LaTeX with hyperref package</xmp:CreatorTool>
<xmpMM:DocumentID>
  uuid:6d1ac9ec-4ff2-515a-954b-648eeb4853b0
</xmpMM:DocumentID>
<xmpMM:InstanceID>
  uuid:3e4c4182-b182-46c9-995f-754c41d13390
```

```
   </xmpMM:InstanceID>
   <xmpMM:VersionID>2.998e8</xmpMM:VersionID>
   <xmpMM:RenditionClass>default</xmpMM:RenditionClass>
   <Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">
    <Iptc4xmpCore:CiAdrExtadr>Kramgasse 49</Iptc4xmpCore:CiAdrExtadr>
    <Iptc4xmpCore:CiAdrCity>Bern</Iptc4xmpCore:CiAdrCity>
    <Iptc4xmpCore:CiAdrPcode>3011</Iptc4xmpCore:CiAdrPcode>
    <Iptc4xmpCore:CiAdrCtry>Switzerland</Iptc4xmpCore:CiAdrCtry>
    <Iptc4xmpCore:CiTelWork>031 312 00 91</Iptc4xmpCore:CiTelWork>
    <Iptc4xmpCore:CiEmailWork>aeinstein@ipi.ch</Iptc4xmpCore:CiEmailWork>
    <Iptc4xmpCore:CiUrlWork>
     http://einstein.biz/,
     https://www.facebook.com/AlbertEinstein
    </Iptc4xmpCore:CiUrlWork>
   </Iptc4xmpCore:CreatorContactInfo>
   <prism:complianceProfile>three</prism:complianceProfile>
   <prism:subtitle xml:lang="en-US">
    Putting that bum Maxwell in his place
   </prism:subtitle>
   <prism:publicationName xml:lang="de">
    Annalen der Physik
   </prism:publicationName>
   <prism:aggregationType>journal</prism:aggregationType>
   <prism:volume>322</prism:volume>
   <prism:number>6</prism:number>
   <prism:pageRange>132-148</prism:pageRange>
   <prism:issn>0003-3804</prism:issn>
   <prism:eIssn>1521-3889</prism:eIssn>
   <prism:doi>10.1002/andp.19053220607</prism:doi>
   <prism:url>
    http://www.physik.uni-augsburg.de/annalen/history/einstein-papers/190517132-
   </prism:url>
   <prism:byteCount>59846</prism:byteCount>
   <prism:pageCount>17</prism:pageCount>
  </rdf:Description>
 </rdf:RDF>
</x:xmpmeta>
<?xpacket end="w"?>
```

References

[1] Adobe Systems, Inc., San Jose, California. *Adobe Acrobat X SDK Help, pdfmark Reference*. Available from `http://www.adobe.com/devnet/acrobat/documentation.html`.

[2] Adobe Systems, Inc. *PostScript Language Reference Manual*. Addison-Wesley, 2nd edition, January 1996, ISBN: 0-201-18127-4.

[3] Adobe Systems, Inc., San Jose, California. *Document Management—Portable Document Format—Part 1: PDF 1.7*, July 2008. ISO 32000-1 standard document. Available from `http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/PDF32000_2008.pdf`.

[4] Adobe Systems, Inc., San Jose, California. *XMP Specification Part 1: Data model, Serialization, and Core Properties*, April 2012. Available from `http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/xmp/pdfs/cc-201306/XMPSpecificationPart1.pdf`.

[5] DCMI Usage Board *DCMI Metadata Terms*, June 14, 2012. Available from `http://dublincore.org/documents/dcmi-terms/`.

[6] Michael Downes. Around the bend #15, answers, 4th (last) installment. `comp.text.tex` newsgroup posting, January 3, 1994. Archived by Google at `http://groups.google.com/group/comp.text.tex/msg/7da7643b9e8f3b48`.

[7] International Digital Enterprise Alliance, Inc. *Publishing Requirements for Industry Standard Metadata, Version 3.0: PRISM Basic Metadata Specification*, October 12, 2012. Available from `http://www.prismstandard.org/specifications/3.0/PRISM_Basic_Metadata_3.0.htm`.

[8] International Digital Enterprise Alliance, Inc. *Publishing Requirements for Industry Standard Metadata, Version 3.0: PRISM Controlled Vocabularies Specification*, October 4, 2012. Available from `http://www.prismstandard.org/specifications/3.0/PRISM_CV_Spec_3.0.pdf`.

[9] International Press Telecommunications Council. *IPTC Photo Metadata: Core 1.1/Extension 1.1*, July 2010. Revision 1. Available from `http://www.iptc.org/std/photometadata/specification/IPTC-PhotoMetadata-201007_1.pdf`.

[10] Internet Assigned Numbers Authority. Language subtag registry, January 11, 2011. Available from `http://www.iana.org/assignments/language-subtag-registry`.

[11] Paul J. Leach, Michael Mealling, and Rich Salz. A Universally Unique IDentifier (UUID) URN namespace. Request for Comments 4122, Internet Engineering Task Force, Network Working Group, July 2005. Category: Standards Track. Available from `http://www.ietf.org/rfc/rfc4122.txt`.

[12] PDF/A Competence Center, Berlin, Germany. *TechNote 0008: Predefined XMP Properties in PDF/A-1*, March 20, 2008. Available from `http://www.pdfa.org/wp-content/uploads/2011/08/tn0008_predefined_xmp_properties_in_pdfa-1_2008-03-20.pdf`.

[13] PDF/A Competence Center, Berlin, Germany. *TechNote 0009: XMP Extension Schemas in PDF/A-1*, March 20, 2008. Available from `http://www.pdfa.org/wp-content/uploads/2011/08/tn0009_xmp_extension_schemas_in_pdfa-1_2008-03-20.pdf`.

[14] Misha Wolf and Charles Wicksteed. Date and time formats. Note NOTE-datetime, World Wide Web Consortium (W3C), September 15, 1997. Available from `http://www.w3.org/TR/NOTE-datetime`.

Change History

v1.0
- General: Initial version 1

v1.1
- `\hyxmp@construct@packet`: Explicitly set the category codes of characters ⟨*EF*⟩, ⟨*BB*⟩, and ⟨*BF*⟩ to “letter”. Thanks to Daniel Schömer for the bug report 69

v1.2
- General: Added support for the XƎLaTeX backend (`xdvipdfmx`) . . 1
  - Added support for the Photoshop schema 1
  - Made the package compatible with ngerman. Thanks to Tobias Mueller for the bug report. 17

v1.3
- General: Introduced the pdfmetalang package option, which enables an author to specify the language in which he wrote the document’s metadata 29

v1.4
- `\hyxmp@mm@schema`: Renamed the `xapMM` namespace prefix to `xmpMM` 56
- `\hyxmp@rdf@dc`: Included metadata in the `x-default` language regardless of the specified metadata language 53
- `\hyxmp@xmpRights@schema`: Renamed the `xapRights` namespace prefix to `xmpRights` 56

v1.5
- General: Made the XMP inclusion more robust. Thanks to Heiko Oberdiek for the bug report and suggested modifications. . 17

v2.0
- `\ProcessKeyvalOptions`: Added this macro 27
- `\XMPTruncateList`: Added this macro 32
- `\hyxmp@ProcessKeyvalOptions`: Added this macro 27
- `\hyxmp@SpaceOther`: Added by Heiko Oberdiek 43
- `\hyxmp@add@simple`: Added this macro 44
- `\hyxmp@add@to@xml`: Updated also to replace commas 50
- `\hyxmp@bom`: Added by Heiko Oberdiek 69
- `\hyxmp@comma`: Added this macro 32
- `\hyxmp@construct@packet`: Modified by Heiko Oberdiek to use an appropriate BOM representation via `\hyxmp@bom` 69
- `\hyxmp@crap@convert`: Added by Heiko Oberdiek 43
- `\hyxmp@crap@test`: Added by Heiko Oberdiek 43
- `\hyxmp@dc@schema`: Added support for dc:language and dc:source . 55
- `\hyxmp@is@unicode`: Added by Heiko Oberdiek 40
- `\hyxmp@list@to@xml`: Modified by Heiko Oberdiek to use the new

Unicode-processing macros 54

`\hyxmp@photoshop@schema`: Simplified using `\hyxmp@add@simple` 57

`\hyxmp@skiptorelax`: Added by Heiko Oberdiek 43

`\hyxmp@skipzeros`: Added by Heiko Oberdiek 42

`\hyxmp@toxml`: Added by Heiko Oberdiek 41

Escaped parentheses written with `pdfmark`s to prevent `dvips` from line-wrapping the XMP packet 41

`\hyxmp@toxml@unicodetex`: Added by Heiko Oberdiek 41

`\hyxmp@xetex@crap`: Added by Heiko Oberdiek 42

`\hyxmp@xmlify`: Completely rewritten by Heiko Oberdiek to better support Unicode-enabled TeX programs 39

`\hyxmp@xmp@basic@schema`: Added this macro 57

`\hyxmp@xmpRights@schema`: Modified to include xmpRights:Marked only when pdfcopyright is specified and xmpRights:WebStatement only when pdflicenseurl is specified 56

`\hyxmp@zero`: Added by Heiko Oberdiek 44

`\ifhyxmp@unicodetex`: Added by Heiko Oberdiek 39

`\xmpcomma`: Added this macro 32

`\xmpquote`: Added this macro 32

General: Added support for the XMP Basic schema and miscellaneous other bits of metadata 1

Heiko Oberdiek's major rewrite of the code to better support native-Unicode TeX implementations (XeTeX and LuaTeX) 1

New `\AtBeginDocument` code from Heiko Oberdiek to properly encode `\@pdfmetalang` 29

v2.1

`\hypersetup`: Added this macro 27

`\hyxmp@hypersetup`: Added this macro 27

`\hyxmp@redefine@Hyp`: Added this macro 26

General: Enabled hyperxmp and hyperref to be loaded in either order. This addresses a bug report by Yury Donskoy 25

v2.2

`\hyxmp@iptc@extensions`: Added this macro to support PDF/A generation 65

`\hyxmp@iptc@schema`: Added this macro 59

`\hyxmp@list@to@lines`: Added this macro 59

`\xmpcomma`: Changed the default from `\relax` to an ordinary comma 32

`\xmplinesep`: Added this macro 58

General: Added support for the IPTC Photo Metadata schema 1

v2.3

`\hyxmp@iptc@extensions`: Gave the Iptc4xmpCore:CreatorContactInfo fields a unique pdfaType:prefix to better support conversion of the document to PDF/A 65

v2.3a

General: Bug fix: Redefine `\@pdflang` as `\@empty` when hyperref has set it to `\relax` 29

v2.3b

`\XMPTruncateList`: Made all definitions local to avoid spurious `Too many unprocessed floats` errors when running with memoir 32

v2.4

`\hyxmp@add@simple@var`: Added this macro 45

`\hyxmp@create@uuid`: Modified this macro to produce a proper version 4 (random or pseudorandom) UUID 49

`\hyxmp@dc@schema`: Made dc:language a Bag instead of an individual item so as to conform to the latest XMP specifications, a detail identified by Florian Breitwieser 55

`\hyxmp@parse@time`: Added this macro 33

`\hyxmp@parse@tz`: Added this macro 34

`\hyxmp@parse@tz@char`: Added this macro 34

`\hyxmp@pdf@schema`: Made `\hyxmp@pdf@schema` *always* include the Adobe PDF schema, even when empty. Florian Breitwieser noted that this is necessary for PDF/A-1b compliance 52

`\hyxmp@pdf@to@xmp@date`: Added this macro 33

`\hyxmp@pdfa@id@schema`: Added this macro 58

`\hyxmp@today@xmp`: Modified the code to parse the time and timezone from `\pdfcreationdate`, as proposed by Florian Breitwieser 37

`\hyxmp@today@xmp@define`: Added this macro 36

`\hyxmp@xmp@to@pdf@date`: Added this macro 34

`\xmptilde`: Added this macro . . . 32

General: Added support for the PDF/A Identification schema, as requested by Florian Breitwieser 1

v2.5

`\hyxmp@add@to@xml`: Updated also to replace underscores and to modify only the text being added, not the already-modified text 50

`\hyxmp@textunderscore`: Added this macro 18

`\hyxmp@uscore`: Added this macro 32

General: Enabled "`\_`" to work within email addresses, as requested by Leonid Sinev 1

v2.6

General: Added support for a new pdfdate key to explicitly specify the document date (and optionally time) 1

v2.7

General: Automatically use `\title` and `\author` if pdftitle and pdfauthor are left unspecified. Thanks to Maciej Radziejewski for the suggestion 30

v2.8

`\hyxmp@add@to@xml`: Corrected inadvertent lowercasing of non-Latin characters when run under XeLaTeX or LuaLaTeX (bug reported by Leonid Sinev) 50

v2.9

`\hyxmp@iptc@schema`: Use Iptc4xmpCore instead of Iptc4ContInfo as the contact-information metadata prefix. Leonid Sinev reports that Acrobat's PDF/A validator seems to prefer Iptc4xmpCore . 59

`\hyxmp@pdfa@id@schema`: Let the author specify the PDF/A part and conformance IDs, as requested by Leonid Sinev . . . 58

General: Force inclusion of dc:creator, dc:title, and dc:description—even if empty—when hyperref is loaded with the pdfa option (suggested by Leonid Sinev) 1

Introduced the pdftype package option, which enables an author to specify the type of document being produced 1

v3.0

`\hyxmp@embed@packet@luatex`: Added this macro 72

`\hyxmp@today@xmp@define`: Modified to accept the name of a macro to define 36

`\hyxmp@xmp@basic@schema`: Made the XMP xmp:CreateDate, xmp:ModifyDate, and xmp:MetadataDate match the PDF CreationDate 57

General: Made the code compatible with LuaTeX 0.85. Thanks to Robert Schlicht, Leonid Sinev, and David Carlisle for bug reports and to Leonid Sinev for helping test the new hyperxmp code 1

v3.1

- `\hyxmp@embed@packet@luatex`: Updated to use `\pdfextension obj uncompressed` as suggested by Hans Hagen 72
- `\hyxmp@embed@packet@pdftex`: Leave the XMP packet—and only the XMP packet—uncompressed in both pdfTeX and pre-0.85 LuaTeX 71

v3.2

- `\hyxmp@as@pdf@date`: Added this macro 34
- `\hyxmp@as@xmp@date`: Added this macro 33
- `\hyxmp@today@xmp@define`: Modified to include hours and minutes 36
- `\hyxmp@xmp@basic@schema`: Honor hyperref's pdfcreationdate and pdfmoddate options plus a new pdfmetadate option. Leonid Sinev requested this additional control and helped test the resulting hyperxmp code 57

v3.3

- `\@pdfsource`: Added this macro and the corresponding pdfsource option, at Niklas Beisert's request 21
- `\XMPLangAlt`: Added this macro based on a request—and some code—by Niklas Beisert to support metadata expressed in multiple languages 46
- `\hyxmp@rdf@dc`: Bug fix: Output the metadata language as correct XML even when hyperref is loaded with the unicode option 53
- General: Don't overwrite an existing pdfmetalang with pdflang or `x-default`. This addresses a bug report by Niklas Beisert 29

v3.4

- `\hyxmp@seed@string`: Correctly handle an author field of all spaces. Bug reported by Gaëtan Leurent 49
- General: Use ifmtarg to test for empty arguments, including non-empty but all spaces 1

v3.5

- `\hyxmp@DocumentID`: Added the pdfdocumentid option, at Michael Osipov's request 21
- `\hyxmp@InstanceID`: Added the pdfinstanceid option, at Michael Osipov's request 21
- `\hyxmp@mm@schema`: Generate `\hyxmp@DocumentID` and `\hyxmp@InstanceID` only if the document does not already define these using the pdfdocumentid and pdfinstanceid options 56
- `\hyxmp@seed@string`: Seed with the TeX timestamp in addition to the document-specified timestamp 49

v4.0

- `\XMPTruncateList`: Deprecated this macro 32
- `\hyxmp@add@simple@lang`: Added this macro 45
- `\hyxmp@begin@ext@decl`: Added this macro 62
- `\hyxmp@declare@field`: Replaced `\hyxmp@declare@resource` with this macro 63
- `\hyxmp@declare@property`: Added this macro 62
- `\hyxmp@end@ext@decl`: Added this macro 62
- `\hyxmp@iptc@extensions`: Moved the header code from here into `\hyxmp@begin@extension@decls` and the trailer code from here into `\hyxmp@end@extension@decls` 65
  Rewrote to more closely honor the XMP specification 65
- `\hyxmp@iptc@schema`: Moved the definition of `\hyxmp@iptc@data`

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.