

The hyperxmp package^{*}

Scott Pakin
scott+hyxmp@pakin.org

July 4, 2016

Abstract

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

1 Introduction

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

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

This is too abstract! Give me an example. Consider a \LaTeX document with three authors: Jack Napier, Edward Nigma, and Harvey Dent. The generated PDF file will contain, among other information, the following stanza of XMP code embedded within it:

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

^{*}This document corresponds to `hyperxmp` v3.0, dated 2016/07/04.

```

</rdf:Seq>
</dc:creator>

```

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

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

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

- authors (`dc:creator`)
- base URL (`xmp:BaseURL`)
- contact address (`lptc4xmpCore:CreatorContactInfo/CiAdrExtadr`,
`lptc4xmpCore:CreatorContactInfo/CiAdrCity`,
`lptc4xmpCore:CreatorContactInfo/CiAdrRegion`,
`lptc4xmpCore:CreatorContactInfo/CiAdrPcode`, and
`lptc4xmpCore:CreatorContactInfo/CiAdrCtry`)
- contact email address(es) (`lptc4xmpCore:CreatorContactInfo/CiEmailWork`)
- contact telephone number(s) (`lptc4xmpCore:CreatorContactInfo/CiTelWork`)
- contact URL(s) (`lptc4xmpCore:CreatorContactInfo/CiUrlWork`)
- copyright (`dc:rights` and `xmpRights:Marked`)
- date (`dc:date`, `xmp:CreateDate`, `xmp:ModifyDate`, and `xmp:MetadataDate`)
- document identifier (`xmpMM:DocumentID`)
- document instance identifier (`xmpMM:InstanceID`)
- document type (`dc:type`)
- file format (`dc:format`)
- keywords (`pdf:Keywords` and `dc:subject`)
- language (`dc:language`)
- L^AT_EX file name (`dc:source`)
- license URL (`xmpRights:WebStatement`)

- metadata writer (`photoshop:CaptionWriter`)
- PDF version (`pdf:PDFVersion`)
- PDF-generating tool (`pdf:Producer` and `xmp:CreatorTool`)
- PDF/A compliance level and version (`pdfaid:part` and `pdfaid:conformance`)
- primary author’s position/title (`photoshop:AuthorsPosition`)
- summary (`dc:description`)
- title (`dc:title`)

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`
- `pdfkeywords`
- `pdflang`
- `pdfproducer`

- pdfsubject
- pdftitle

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

- pdfaconformance
- pdfapart
- pdfauthortitle
- pdfcaptionwriter
- pdfcontactaddress
- pdfcontactcity
- pdfcontactcountry
- pdfcontactemail
- pdfcontactphone
- pdfcontactpostcode
- pdfcontactregion
- pdfcontacturl
- pdfcopyright
- pdfdate
- pdflicenseurl
- pdfmetalang
- pdftype

The two most obscure—but alphabetically first—of the above, **pdfaconformance** and **pdfapart**, are used in conjunction with **hyperref**’s **pdfa** option to claim a particular PDF/A standard by which the document abides. They default to **pdfapart=1** and **pdfaconformance=B**, indicating the PDF/A-1B standard. These can be changed (with caution) to assert that the document abides by a different standard (e.g., PDF/A-2U).

pdfauthortitle indicates the primary author’s position or title. **pdfcaptionwriter** specifies the name of the person who added the metadata to the document. The next eight items describe how to contact the person or institution responsible for the document (the “contact”). **pdfcontactaddress** is the contact’s street address and can include the institution name if the contact is an institution;

`pdfcontactcity` is the contact’s city. `pdfcontactcountry` is the contact’s country; `pdfcontactemail` is the contact’s email address (or multiple, comma-separated email addresses); `pdfcontactphone` is the contact’s telephone number (or multiple, comma-separated telephone numbers); `pdfcontactpostcode` is the contact’s postal code; `pdfcontactregion` is the contact’s state or province; and `pdfcontacturl` is the contact’s URL (or multiple, comma-separated URLs).

`pdfcopyright` defines the copyright text. `pdfdate` specifies the document date. It is analogous to the L^AT_EX `\date` command, and, like `\date`, defaults to the date the document was built. However, `pdfdate` must be specified in `YYYY-MM-DDThh:mm:ss.ff+TT:tt` format as per the W3C’s recommendation [12]. For example, 14 hours, 15 minutes, 9.26 seconds past midnight U.S. Mountain Daylight Time (UTC-6) on the 23rd day of September in the year 2014 should be written as `2014-09-23T14:15:09.26-06:00`. This can be truncated to `2014-09-23T14:15:09-06:00` or `2014-09-23T14:15-06:00` or `2014-09-23` or `2014-09` or `2014` but no other subsets. `hyperxmp` does not validate `pdfdate`’s argument, but an invalid format may confuse a PDF reader.

`pdflicenseurl` identifies a URL that points to the document’s license agreement. `pdfmetalang` indicates the natural language in which the metadata is written, typically as an IETF language tag [8], 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 is always included in addition to the specified metadata language, as the user reading the document may not have specified a language preference.

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

It is usually more convenient to provide values for those options using `hyperref`’s `\hypersetup` command than on the `\usepackage` command line. See the `hyperref` manual for more information. The following is a sample L^AT_EX document that provides values for most of the metadata options that `hyperxmp` recognizes:

```
\documentclass{article}
\usepackage{hyperxmp}
\usepackage{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},
  pdfauthor={Albert Einstein},
  pdfauthortitle={Technical Assistant, 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
  },
  pdflang={en},
  baseurl={http://mirror.ctan.org/macros/latex/contrib/hyperxmp/}
}
\begin{document}
\maketitle
A profound formal difference exists between the theoretical
concepts that physicists have formed about gases and other
ponderable bodies, and Maxwell's theory of electromagnetic
processes in so-called empty space\dots
\end{document}

```

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

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

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

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

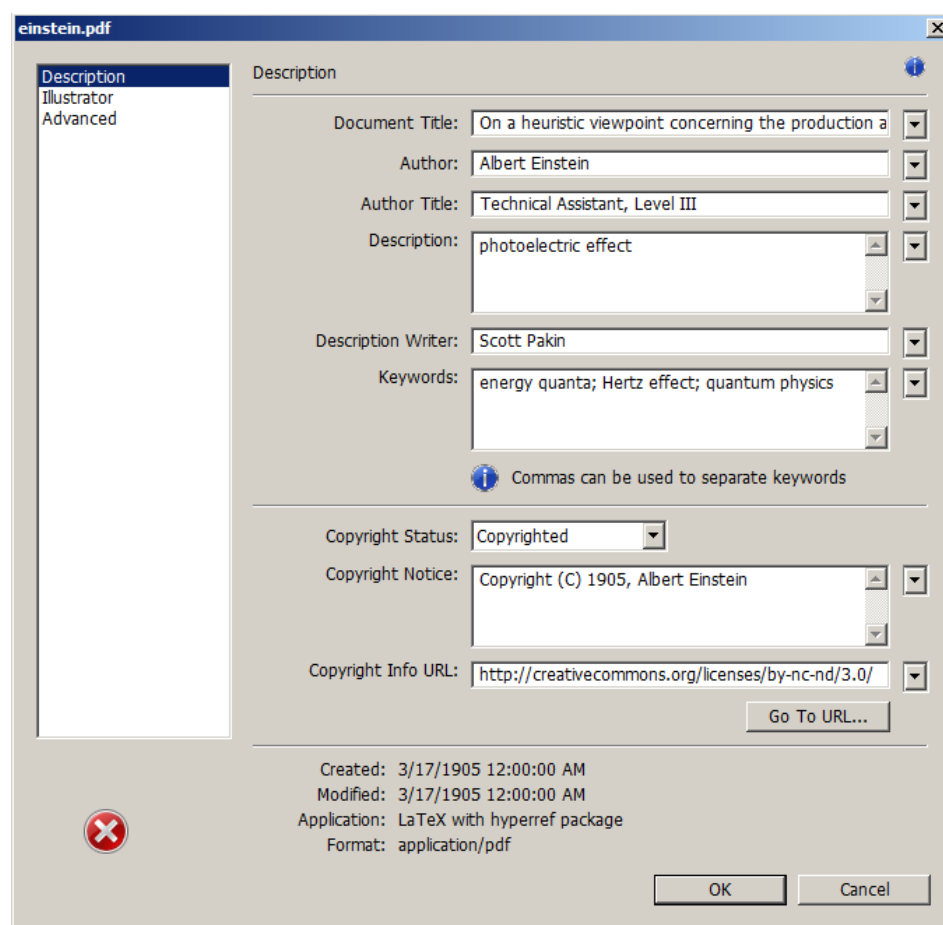


Figure 1: XMP metadata as it appears in Adobe Acrobat

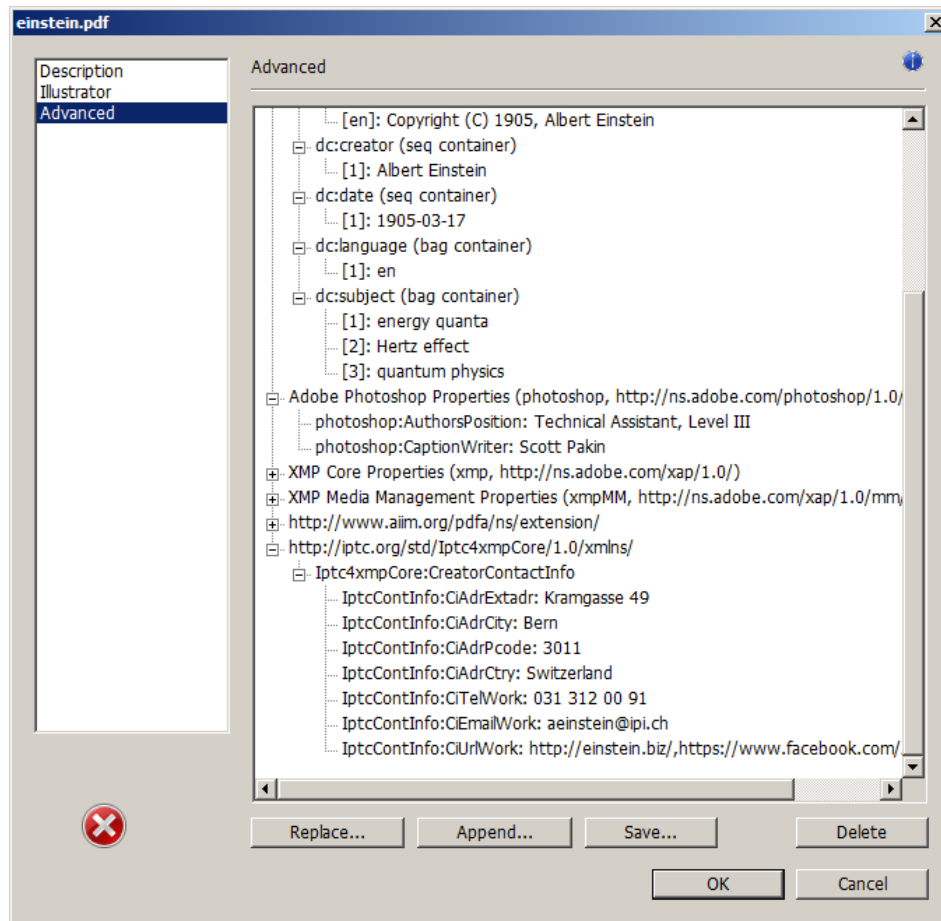


Figure 2: Additional XMP metadata as it appears in Adobe Acrobat

Note 1: Acrobat Author bug A bug in Adobe Acrobat—at least in versions 10.0.1 and earlier—causes that PDF reader to confuse the XMP and non-XMP author lists when displaying the document’s metadata. Specifically, the first author is displayed as the concatenated list of authors from the non-XMP data (`Author`) while the remaining authors are displayed from the XMP data (`dc:creator`). For example, suppose that a document’s authors are Jack Napier, Edward Nigma, and Harvey Dent. When displaying the document properties, Adobe Acrobat replaces “Jack Napier” with a single author named “Jack Napier, Edward Nigma, Harvey Dent” and leaves “Edward Nigma” and “Harvey Dent” as the second and third authors, respectively.

`\XMPTruncateList` The `hyperxmp` package provides a workaround for this bug in the form of the `\XMPTruncateList` macro. `\XMPTruncateList` takes the name of a list (a `hyperref` option name) and replaces the list with the value of its first element. Currently,

the only meaningful usage is to put

```
\XMPTruncateList{pdfauthor}
```

in your document’s preamble. This will cause Adobe Acrobat to properly display all of the authors but at the cost of other PDF readers likely displaying only the first author.

Note 2: Acrobat multiline-field bug The IPTC Photo Metadata schema states that “the [contact] address is a multiline field” [7]. `hyperxmp` converts commas in `pdfcontactaddress`’s argument to line breaks in the generated XML. Unfortunately, A bug in Adobe Acrobat—at least in versions 10.0.1 and earlier—causes that PDF reader to discard line breaks in the contact address. Interestingly, `\xmplinesep` Adobe Illustrator CS5 correctly displays the contact address. If you find Adobe Acrobat’s behavior bothersome, you can redefine the `\xmplinesep` macro as a string to use as an address-line separator. For example, the following replaces all commas appearing in `pdfcontactaddress`’s argument with semicolons:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3 Implementation

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

3.1 Initial preparation

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

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

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

`\hyxmp@driver`

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

3.2 Integration with `hyperref`

An important design decision underlying `hyperxmp` is that the package should integrate seamlessly with `hyperref`. To that end, `hyperxmp` takes its XMP metadata from `hyperref`’s `baseurl`, `pdfauthor`, `pdfkeywords`, `pdflang`, `pdfproducer`, `pdfsubject`, and `pdftitle` options. It also introduces a number of new options, which are listed on page 4. For consistency with `hyperref`’s document-metadata naming conventions (which are in turn based on L^AT_EX’s document-metadata naming conventions), we do not prefix metadata-related macro names with our package-specific `\hyxmp@` prefix. That is, we use names like `\@pdfcopyright` instead of `\hyxmp@pdfcopyright`.

We load a bunch of helper packages: `koptions` for package-option processing, `pdfescape` and `stringenc` for re-encoding Unicode strings, `intcalc` for performing integer calculations (division and modulo), and `ifxetex` for detecting Xe_{La}TeX.

```
10 \RequirePackage{koptions}
11 \RequirePackage{pdfescape}
12 \RequirePackage{stringenc}
13 \RequirePackage{intcalc}
14 \RequirePackage{ifxetex}
```

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

```
15 \newcommand{\hyxmp@pdfstringdef}[2]{%
16   \let\hyxmp@textunderscore=\textunderscore
17   \let\textunderscore=\hyxmp@uscore
18   \pdfstringdef{#1}{#2}%
19   \let\textunderscore=\hyxmp@textunderscore
20 }
```

`\@pdfdatetime` Prepare to store the document’s date and (optionally) time.

```
21 \def\@pdfdatetime{}
22 \define@key{Hyp}{pdfdate}{\hyxmp@pdfstringdef\@pdfdatetime{#1}}
```

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

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

`\@pdftype` Prepare to store the document’s logical type, which defaults to “Text”.

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

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

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

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

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

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

```
31 \def\@pdfcaptionwriter{}
32 \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.

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

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

```
35 \def\@pdfapart{1}
36 \define@key{Hyp}{pdfapart}{\hyxmp@pdfstringdef\@pdfapart{#1}}
```

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

```
37 \def\@pdfaconformance{B}
38 \define@key{Hyp}{pdfaconformance}{\hyxmp@pdfstringdef\@pdfaconformance{#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.

```
39 \def\@pdfcontactaddress{}
40 \define@key{Hyp}{pdfcontactaddress}{%
41   \let\xmpcomma=\hyxmp@comma
42   \def\xmpquote##1{##1}%
43   \hyxmp@pdfstringdef\@pdfcontactaddress{#1}%
44   \def\xmpcomma{,}%
45   \let\xmpquote=\relax
46 }
```

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

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

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

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

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

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

<code>\@pdfcontactcountry</code>	<p>Prepare to store the country of the document's contact person/institution.</p> <pre> 53 \def\@pdfcontactcountry{} 54 \define@key{Hyp}{pdfcontactcountry}{\hyxmp@pdfstringdef\@pdfcontactcountry{#1}} </pre>
<code>\@pdfcontactphone</code>	<p>Prepare to store the telephone number of the document's contact person/institution.</p> <pre> 55 \def\@pdfcontactphone{} 56 \define@key{Hyp}{pdfcontactphone}{\hyxmp@pdfstringdef\@pdfcontactphone{#1}} </pre>
<code>\@pdfcontactemail</code>	<p>Prepare to store the email address of the document's contact person/institution.</p> <pre> 57 \def\@pdfcontactemail{} 58 \define@key{Hyp}{pdfcontactemail}{\hyxmp@pdfstringdef\@pdfcontactemail{#1}} </pre>
<code>\@pdfcontacturl</code>	<p>Prepare to store the URL of the document's contact person/institution.</p> <pre> 59 \def\@pdfcontacturl{} 60 \define@key{Hyp}{pdfcontacturl}{\hyxmp@pdfstringdef\@pdfcontacturl{#1}} </pre>

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`.

<code>\hyxmp@pdfauthor</code> <code>\hyxmp@pdfkeywords</code>	<p>Prepare to store the name of the author and a list of keywords.</p> <pre> 61 \def\hyxmp@pdfauthor{} 62 \def\hyxmp@pdfkeywords{} </pre>
<code>\hyxmp@redefine@Hyp</code>	<p>If not already redefined, redefine <code>hyperref</code>'s <code>pdfauthor</code> and <code>pdfkeywords</code> options to properly handle <code>\xmpcomma</code> and <code>\xmpquote</code>.</p> <pre> 63 \newcommand*{\hyxmp@redefine@Hyp}{% </pre>
<code>\hyxmp@Hyp@pdfauthor</code>	<p>Store the old definition of <code>\KV@Hyp@pdfauthor</code> in <code>\hyxmp@Hyp@pdfauthor</code>, but only if we see that <code>\KV@Hyp@pdfauthor</code> is defined and <code>\hyxmp@Hyp@pdfauthor</code> isn't. Otherwise, we'd be defining <code>\hyxmp@Hyp@pdfauthor</code> in terms of itself and creating an infinite loop.</p> <pre> 64 \@ifundefined{KV@Hyp@pdfauthor}{}{% 65 \@ifundefined{hyxmp@Hyp@pdfauthor}{% 66 \expandafter\let\expandafter\hyxmp@Hyp@pdfauthor 67 \csname KV@Hyp@pdfauthor\endcsname 68 }{}% 69 }% </pre>

\KV@Hyp@pdfauthor \xmpcomma \xmpquote \hyxmp@and \and \hyxmp@pdfauthor \@pdfauthor	<p>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.</p> <pre> 70 \define@key{Hyp}{pdfauthor}{% 71 \let\xmpcomma=\hyxmp@comma 72 \def\xmpquote####1{####1}% 73 \let\hyxmp@and=\and 74 \def\and{,}% 75 \hyxmp@Hyp@pdfauthor{##1}% 76 \global\let\hyxmp@pdfauthor=\@pdfauthor 77 \def\and{and\space}% 78 \def\xmpcomma{,%} 79 \def\xmpquote####1{"####1"% 80 \hyxmp@Hyp@pdfauthor{##1}% 81 \def\xmpcomma{,%} 82 \let\xmpquote=\relax 83 \let\and=\hyxmp@and 84 }%</pre>
\hyxmp@Hyp@pdfkeywords	<p>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.</p> <pre> 85 \@ifundefined{KV@Hyp@pdfkeywords}{}{% 86 \@ifundefined{hyxmp@Hyp@pdfkeywords}{% 87 \expandafter\let\expandafter\hyxmp@Hyp@pdfkeywords 88 \csname KV@Hyp@pdfkeywords\endcsname 89 }{}% 90 }%</pre>
\KV@Hyp@pdfkeywords \xmpcomma \xmpquote \hyxmp@pdfkeywords \@pdfkeywords	<p>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).</p> <pre> 91 \define@key{Hyp}{pdfkeywords}{% 92 \let\xmpcomma=\hyxmp@comma 93 \def\xmpquote####1{####1}% 94 \hyxmp@Hyp@pdfkeywords{##1}% 95 \global\let\hyxmp@pdfkeywords=\@pdfkeywords</pre>

```

96     \def\xmpcomma{,}%
97     \def\xmpquote####1{"####1"}%
98     \hyxmp@Hyp@pdfkeywords{##1}%
99     \def\xmpcomma{,}%
100    \let\xmpquote=\relax
101  }%
102 }

\hyxmp@ProcessKeyvalOptions  Redefine kvoptions's \ProcessOptions command to invoke \hyxmp@redefine@Hyp
\ProcessKeyvalOptions        before performing its normal option processing.
103 \let\hyxmp@ProcessKeyvalOptions=\ProcessKeyvalOptions
104 \renewcommand*{\ProcessKeyvalOptions}{%
105   \hyxmp@redefine@Hyp
106   \hyxmp@ProcessKeyvalOptions
107 }

\hyxmp@hypersetup  Redefine hyperref's \hypersetup command to invoke \hyxmp@redefine@Hyp before
\hypersetup        performing its normal option processing.
108 \let\hyxmp@hypersetup=\hypersetup
109 \def\hypersetup{%
110   \hyxmp@redefine@Hyp
111   \hyxmp@hypersetup
112 }

\hyxmp@find@metadata  Issue a warning message if the author failed to specify any metadata at all. This
\hyxmp@concat@metadata excludes metadata that is included automatically such as the current timestamp.
Note that we don't consider \@pdfmetalang as metadata as that value is meaningful
only when used in conjunction with other information. We also don't examine
\@pdfapart or \@pdfaconformance because those have nonempty default values.
113 \newcommand*{\hyxmp@find@metadata}{%
114   \edef\hyxmp@concat@metadata{%
115     \@baseurl
116     \@pdfauthor
117     \@pdfauthortitle
118     \@pdfcaptionwriter
119     \@pdfcontactaddress
120     \@pdfcontactcity
121     \@pdfcontactcountry
122     \@pdfcontactemail
123     \@pdfcontactphone
124     \@pdfcontactpostcode
125     \@pdfcontactregion
126     \@pdfcontacturl
127     \@pdfcopyright
128     \@pdftype
129     \@pdfdatetime
130     \@pdfkeywords
131     \@pdflang
132     \@pdflicenseurl

```



```

133     \@pdfsubject
134     \@pdftitle
135 }%
136 \ifx\hyxmp@concat@metadata\@empty
137     \PackageWarningNoLine{hyperxmp}{%
138 \jobname.tex did not specify any metadata to\MessageBreak
139 include in the XMP packet.\space\space Please see the\MessageBreak
140 hyperxmp documentation for instructions on how to\MessageBreak
141 provide metadata values to hyperxmp}%
142 \fi
143 }

```

Rather than load `hyperref` ourselves we let the author do it then verify he actually did. This approach gives the author the flexibility to load `hyperxmp` and `hyperref` in either order and to call `\hypersetup` anywhere in the document's preamble, not just before `hyperxmp` is loaded.

```

144 \AtBeginDocument{%
145     \@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`.

```

146     \ifx\@pdflang\relax
147         \let\@pdflang=\@empty
148     \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.

```

149     \ifx\@pdflang\@empty
150         \let\@pdfmetalang=\hyxmp@x@default
151     \else
152         \edef\@pdfmetalang{\@pdflang}%
153     \fi
154     \hyxmp@xmlify\@pdfmetalang

```

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

```

155     \ifx\@pdfdatetime\@empty
156     \else
157         \edef\hyxmp@today{\@pdfdatetime}%
158     \fi

```

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

```

159     \ifx\@pdftitle\@empty

```

```

160     \ifx\@title\@empty
161     \else
162         \hypersetup{pdftitle={\@title}}}%
163     \fi
164 \fi
165 \ifx\@pdfauthor\@empty
166     \ifx\@author\@empty
167     \else
168         \hypersetup{pdfauthor={\@author}}}%
169     \fi
170 \fi

```

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`.

```

171     \hyxmp@at@end{%
172         \hyxmp@find@metadata
173         \hyxmp@embed@packet
174     }%
175 }{%
176     \PackageWarningNoLine{hyperxmp}{%
177 \jobname.tex failed to include a\MessageBreak
178 \string\usepackage\string{hyperref\string}
179 in the preamble.\MessageBreak
180 Consequently, all hyperxmp functionality will be\MessageBreak
181 disabled}%
182 }%
183 }

```

3.3 Manipulating author-supplied data

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

3.3.1 List manipulation

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

`\hyxmp@commas@to@list` 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.)

```

184 \newcommand*{\hyxmp@commas@to@list}[2]{%
185     \gdef#1{%

```

```

186 \expandafter\hyxmp@commas@to@list@i\expandafter#1#2,,%
187 }

```

`\hyxmp@commas@to@list@i` Recursively construct macro #1 from comma-separated list #2. Stop if #2 is empty.

```

\next 188 \def\hyxmp@commas@to@list@i#1#2,{%
189 \gdef\hyxmp@sublist{#2}%
190 \ifx\hyxmp@sublist\@empty
191 \let\next=\relax
192 \else
193 \hyxmp@trimspaces\hyxmp@sublist
194 \@cons{#1}{\hyxmp@sublist}%
195 \def\next{\hyxmp@commas@to@list@i{#1}}%
196 \fi
197 \next
198 }

```

`\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.

```

199 \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.

```

200 \bgroup
201 \catcode'\^^C=11
202 \gdef\hyxmp@comma{^^C}
203 \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.

```

204 \bgroup
205 \catcode'\^^U=11
206 \gdef\hyxmp@uscore{^^U}
207 \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.

```

208 \let\xmpquote=\relax

```

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

```

209 \bgroup
210 \catcode'\~=12%
211 \gdef\xmptilde{~}%
212 \egroup

```

`\XMPTruncateList` As a workaround for Adobe Acrobat’s inability to display author lists correctly (cf. “Acrobat Author bug” on page 7) 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. It’s sad that this is necessary, though.

```

213 \newcommand{\XMPTruncateList}[1]{%
214 \edef\hyxmp@temp@str{\csname hyxmp@#1\endcsname}%
215 \hyxmp@commas@to@list{\hyxmp@temp@list}{\hyxmp@temp@str}%
216 \def\@elt##1{%
217 \expandafter\gdef\csname @#1\endcsname{##1}%
218 \let\@elt=\gobble
219 }
220 \hyxmp@temp@list
221 }}

```

3.3.2 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@trunc`—was taken almost verbatim from a solution to an *Around the Bend* puzzle [6]. Inline comments are also taken from the solution text.

```

222 \catcode'\Q=3

```

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

```

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

```

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

```

224 \begingroup

```

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

```

225 \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.

```

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

```

Transfer the trimmed text back into `#1`.

```

227 \edef#1{\the\toks0}%
228 }

```

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

```
229 \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.

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

3.3.3 Converting text to XML

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

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

```
232 \newif\ifhyxmp@unicodetex
233 \ifnum64='^^^^0040\relax
234   \hyxmp@unicodetextrue
235 \else
236   \hyxmp@unicodetexfalse
237 \fi
```

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

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

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

```
239 \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.

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

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

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

```

241 \newcommand*{\hyxmp@xmlify}[1]{%
242   \gdef\hyxmp@xmlified{%
    Escaped PDF string → PDFDocEncoding/Unicode
243   \EdefUnescapeString\hyxmp@text{#1}%
244   \ifhyxmp@unicodetex
    PDFDocEncoding/Unicode → UTF-32BE
245     \hyxmp@is@unicode\hyxmp@text{%
246       \StringEncodingConvert
247       \hyxmp@text\hyxmp@text{utf16be}{utf32be}%
248     }{%
249       \ifxetex
250         \hyxmp@xetex@crap
251       \else
252         \StringEncodingConvert
253         \hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
254       \fi
255     }%
    UTF-32BE → UTF-32BE as hex string
256     \EdefEscapeHex\hyxmp@text{\hyxmp@text}%
    UTF-32BE → XML in ASCII
257     \edef\hyxmp@text{%
258       \expandafter
259       \expandafter\hyxmp@toxml@unicodetex\hyxmp@text
260       \relax\relax\relax\relax\relax\relax\relax\relax
261     \else
    PDFDocEncoding/Unicode → UTF-8
262     \hyxmp@is@unicode\hyxmp@text{%
263       \StringEncodingConvert
264       \hyxmp@text\hyxmp@text{utf16be}{utf8}%
265     }{%
266       \StringEncodingConvert
267       \hyxmp@text\hyxmp@text{pdfdoc}{utf8}%
268     }%
    UTF-8 → UTF-8 as hex string
269     \EdefEscapeHex\hyxmp@text{\hyxmp@text}%
    UTF-8 as hex string → XML in UTF-8 as hex string
270     \edef\hyxmp@text{%
271       \expandafter\hyxmp@toxml\hyxmp@text\@empty\@empty
272     }%
    XML in UTF-8 as hex string → XML in UTF-8
273     \EdefUnescapeHex\hyxmp@text{\hyxmp@text}%
274     \fi
275     \global\let\hyxmp@xmlified\hyxmp@text

```

276 }

`\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.

```

277 \begingroup
278   \lccode'\<=254 %
279   \lccode'\>=255 %
280   \catcode254=12 %
281   \catcode255=12 %
282 \lowercase{\endgroup
283   \def\hyxmp@is@unicode#1{%
284     \expandafter\hyxmp@@is@unicode#1<>\@nil
285   }%
286   \def\hyxmp@@is@unicode#1<>#2\@nil{%
287     \ifx\#1\%
288       \expandafter\@firstoftwo
289     \else
290       \expandafter\@secondoftwo
291     \fi
292   }%
293 }
```

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

```

294 \def\hyxmp@toxml#1#2{%
295   \ifx#1\@empty
296   \else
297     \ifnum"#1#2='\& %
298       26616D703B% &
299     \else\ifnum"#1#2='\< %
300       266C743B% <
301     \else\ifnum"#1#2='\> %
302       2667743B% >
303   \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.

```

304   \@ifundefined{pdfmark}{%
305     #1#2%
306   }{%
```

```

307     \ifnum"#1#2='\'( %
308         5C28% \'(
309     \else\ifnum"#1#2='\' %
310         5C29% \)
311     \else
312         #1#2%
313     \fi\fi
314 }%
315 \fi\fi\fi
316 \expandafter\hyxmp@toxml
317 \fi
318 }

```

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

```

319 \def\hyxmp@toxml@unicodetex#1#2#3#4#5#6#7#8{%
320     \ifx#1\relax
321     \else
322         \ifnum"#1#2#3#4#5#6#7#8>127 %
323             \uccode'\*="#1#2#3#4#5#6#7#8\relax
324             \uppercase{%
325                 \edef\hyxmp@text{\hyxmp@text *}%
326             }%
327         \else\ifnum"#7#8='\'< %
328             \edef\hyxmp@text{\hyxmp@text &lt;}%
329         \else\ifnum"#7#8='\'& %
330             \edef\hyxmp@text{\hyxmp@text &}%
331         \else\ifnum"#7#8='\'> %
332             \edef\hyxmp@text{\hyxmp@text &gt;}%
333         \else\ifnum"#7#8='\' %
334             \edef\hyxmp@text{\hyxmp@text\space}%
335         \else
336             \uccode'\*="#7#8\relax
337             \uppercase{%
338                 \edef\hyxmp@text{\hyxmp@text *}%
339             }%
340         \fi\fi\fi\fi\fi
341     \expandafter\hyxmp@toxml@unicodetex
342 \fi
343 }

```

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

```

344 \def\hyxmp@skipzeros#1{%
345     \ifx#10%
346         \expandafter\hyxmp@skipzeros
347     \fi
348 }

```

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

`\hyxmp@xetex@crap`

`\hyxmp@try`

`\hyxmp@crap@result`

`\hyxmp@text`


```

349 \begingroup
350 \def\x#1{\endgroup
351   \def\hyxmp@xetex@crap{%
352     \edef\hyxmp@try{%
353       \expandafter\hyxmp@SpaceOther\hyxmp@text#1\@nil
354     }%
355     \let\hyxmp@crap@result=N%
356     \expandafter\hyxmp@crap@test\hyxmp@try\relax
357     \ifx\hyxmp@crap@result Y%
358       \let\hyxmp@text\@empty
359       \expandafter\hyxmp@crap@convert\hyxmp@try\relax
360     \else
361       \StringEncodingConvert\hyxmp@text\hyxmp@text{pdfdoc}{utf32be}%
362     \fi
363   }%
364 }
365 \x{ }

```

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

```

366 \begingroup
367 \catcode'\~=12 %
368 \lccode'\~=\' %
369 \lowercase{\endgroup
370 \def\hyxmp@SpaceOther#1 #2\@nil{%
371   #1%
372   \ifx\relax#2\relax
373     \expandafter\@gobble
374   \else
375     ~%
376     \expandafter\@firstofone
377   \fi
378   {\hyxmp@SpaceOther#2\@nil}%
379 }%
380 }

```

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

```

381 \def\hyxmp@crap@test#1{%
382   \ifx#1\relax
383   \else
384     \ifnum'#1>127 %
385       \let\hyxmp@crap@result=Y%
386       \expandafter\expandafter\expandafter\hyxmp@skiptorelax
387     \else
388       \expandafter\expandafter\expandafter\hyxmp@crap@test
389     \fi
390   \fi
391 }

```

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

```

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

```

```

\hyxmp@crap@convert  Convert a hexadecimal string to a number.
\hyxmp@num 393 \def\hyxmp@crap@convert#1{%
\hyxmp@text 394 \ifx#1\relax
395 \else
396 \edef\hyxmp@num{\number'#1}%
397 \ifnum\hyxmp@num>"FFFFFF %
398 \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"1000000}\relax
399 \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
400 \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"1000000}}%
401 \else
402 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
403 \fi
404 \ifnum\hyxmp@num>"FFFF %
405 \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"10000}\relax
406 \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
407 \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"10000}}%
408 \else
409 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
410 \fi
411 \ifnum\hyxmp@num>"FF %
412 \lccode'\!=\intcalcdDiv{\hyxmp@num}{\number"100}\relax
413 \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
414 \edef\hyxmp@num{\intcalcmMod{\hyxmp@num}{\number"100}}%
415 \else
416 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
417 \fi
418 \ifnum\hyxmp@num>0 %
419 \lccode'\!=\hyxmp@num\relax
420 \lowercase{\edef\hyxmp@text{\hyxmp@text!}}%
421 \else
422 \edef\hyxmp@text{\hyxmp@text\hyxmp@zero}%
423 \fi
424 \expandafter\hyxmp@crap@convert
425 \fi
426 }

```

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

```

427 \begingroup
428 \catcode0=12 %
429 \gdef\hyxmp@zero{^^00}%
430 \endgroup

```

3.4 UUID generation

We use a linear congruential generator to produce pseudorandom version 4 UUIDs [9]. 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.

```
431 \def\hyxmp@modulo@a#1{%
432   \@tempcntb=\@tempcnta
433   \divide\@tempcntb by #1
434   \multiply\@tempcntb by #1
435   \advance\@tempcnta by -\@tempcntb
436 }
```

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

```
\hyxmp@big@prime@ii 437 \def\hyxmp@big@prime{536870923}
438 \def\hyxmp@big@prime@ii{536870027}
```

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

```
\hyxmp@one@token 439 \def\hyxmp@seed@rng#1{%
440   \@tempcnta=\hyxmp@big@prime
441   \futurelet\hyxmp@one@token\hyxmp@seed@rng@i#1\@empty
442 }
```

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

```
\next 443 \def\hyxmp@seed@rng@i{%
444   \ifx\hyxmp@one@token\@empty
445     \let\next=\relax
446   \else
447     \def\next##1{%
448       \multiply\@tempcnta by 3
449       \advance\@tempcnta by '##1
450       \hyxmp@modulo@a{\hyxmp@big@prime}%
451       \futurelet\hyxmp@one@token\hyxmp@seed@rng@i
452     }%
453   \fi
454 \next
455 }
```

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

```
456 \def\hyxmp@set@rand@num{%
457   \@tempcnta=\hyxmp@rand@num
458   \multiply\@tempcnta by 3
459   \advance\@tempcnta by \hyxmp@big@prime@ii
460   \hyxmp@modulo@a{\hyxmp@big@prime}%
461   \xdef\hyxmp@rand@num{the\@tempcnta}%
462 }
```

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

```

463 \def\hyxmp@append@hex#1{%
464   \hyxmp@set@rand@num
465   \@tempcnta=\hyxmp@rand@num
466   \hyxmp@modulo@a{16}%
467   \ifnum\@tempcnta<10
468     \xdef#1{#1\the\@tempcnta}%
469   \else

```

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

```

470   \advance\@tempcnta by -10
471   \ifcase\@tempcnta
472     \xdef#1{#1a}%
473     \or\xdef#1{#1b}%
474     \or\xdef#1{#1c}%
475     \or\xdef#1{#1d}%
476     \or\xdef#1{#1e}%
477     \or\xdef#1{#1f}%
478   \fi
479 \fi
480 }

```

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

```

481 \def\hyxmp@append@hex@iii#1{%
482   \hyxmp@append@hex#1%
483   \hyxmp@append@hex#1%
484   \hyxmp@append@hex#1%
485 }

```

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

```

486 \def\hyxmp@append@hex@iv#1{%
487   \hyxmp@append@hex@iii#1%
488   \hyxmp@append@hex#1%
489 }

```

\hyxmp@create@uuid As per the definition of a version 4 UUID [9], define macro #1 as a UUID of the form “uuid:xxxxxxx-xxx-4xxx-yxx-xxxxxxxx” 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.

```

490 \def\hyxmp@create@uuid#1{%
491   \def#1{uuid:}%
492   \hyxmp@append@hex@iv#1%
493   \hyxmp@append@hex@iv#1%
494   \g@addto@macro#1{-}%
495   \hyxmp@append@hex@iv#1%
496   \g@addto@macro#1{-4}%
497   \hyxmp@append@hex@iii#1%
498   \g@addto@macro#1{-}%

```

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

```

499 \hyxmp@set@rand@num
500 \@tempcnta=\hyxmp@rand@num
501 \hyxmp@modulo@a{4}%
502 \ifcase\@tempcnta
503   \g@addto@macro#1{8}%
504   \or\g@addto@macro#1{9}%
505   \or\g@addto@macro#1{a}%
506   \or\g@addto@macro#1{b}%
507 \fi
508 \hyxmp@append@hex@iii#1%
509 \g@addto@macro#1{-}%
510 \hyxmp@append@hex@iv#1%
511 \hyxmp@append@hex@iv#1%
512 \hyxmp@append@hex@iv#1%
513 }

```

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

```

514 \newcommand*{\hyxmp@def@DocumentID}{%
515   \edef\hyxmp@seed@string{\jobname:\@pdftitle:\@pdfauthor}%
516   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
517   \edef\hyxmp@rand@num{\the\@tempcnta}%
518   \hyxmp@create@uuid\hyxmp@DocumentID
519 }

```

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

```

520 \newcommand*{\hyxmp@def@InstanceID}{%
521   \edef\hyxmp@seed@string{\jobname:\@pdftitle:\@pdfauthor:\hyxmp@today}%
522   \expandafter\hyxmp@seed@rng\expandafter{\hyxmp@seed@string}%
523   \edef\hyxmp@rand@num{\the\@tempcnta}%
524   \hyxmp@create@uuid\hyxmp@InstanceID
525 }

```

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 3.5.8), and PDF/A Identification (Section 3.5.9). The `\hyxmp@construct@packet` macro (Section 3.5.10) 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.

```

526 \newcommand*{\hyxmp@add@to+xml}[1]{%
527   \bgroup
528     \@tempcnta=0
529     \ifhyxmp@unicodetex
530       \@tempcntb=65536%
531     \else
532       \@tempcntb=256%
533     \fi
534     \loop
535       \lccode\@tempcnta=\@tempcnta
536       \advance\@tempcnta by 1
537       \ifnum\@tempcnta<\@tempcntb
538         \repeat
539         \lccode'\_='\ \relax
540         \lccode'\^C='\,\relax
541         \lccode'\^U='\_\relax
542         \lowercase{\xdef\hyxmp@new+xml{#1}}%
543         \xdef\hyxmp+xml{\hyxmp+xml\hyxmp@new+xml}%
544     \egroup
545 }
```

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

```

546 \bgroup
547 \catcode'\#=11
548 \gdef\hyxmp@hash{#}
549 \egroup
```

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

```

550 \bgroup
551 \xdef\hyxmp+xml{%
552   \hyxmp@add@to+xml{%
553     -----^^J%
554   }
555   \xdef\hyxmp@padding{\hyxmp+xml}%
556 \egroup
557 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
558 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
559 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
```

```

560 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}
561 \xdef\hyxmp@padding{\hyxmp@padding\hyxmp@padding}

\hyxmp@pdf@to@xmp@date Convert a timestamp from PDF's D:YYYYMMDDhhmmss-TT'tt' format
(e.g., D:20160704110036-06'00') to XMP's YYYY-MM-DDThh:mm:ss+TT:tt for-
mat (e.g., 2016-07-04T11:00:36-06:00) [4]. This macro is fully expandable.
562 \def\hyxmp@pdf@to@xmp@date#1:#2#3#4#5#6#7#8#9{%
563   #2#3#4#5-#6#7-#8#9%
564   \hyxmp@parse@time
565 }

\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.
566 \def\hyxmp@parse@time#1#2#3#4#5#6{%
567   T#1#2:#3#4:#5#6%
568   \hyxmp@parse@tz@char
569 }

\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+ $x$ , includ-
ing Asia, Oceania, and most of Europe), "-" for western timezones (UTC- $x$ , pri-
marily 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.
570 \def\hyxmp@parse@tz@char#1{%
571   #1%
572   \ifx#1-%
573     \expandafter\hyxmp@parse@tz
574   \else
575     \ifx#1+%
576       \expandafter\hyxmp@parse@tz
577     \fi
578   \fi
579 }

\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.
580 \def\hyxmp@parse@tz#1'#2' {%
581   #1:#2%
582 }

\hyxmp@today@define Use TEX's \year, \month, and \day primitives to define a given macro as today's
date in YYYY-MM-DD format.
583 \def\hyxmp@today@define#1{%

```

```

584 \xdef#1{\the\year}%
585 \ifnum\month<10
586   \xdef#1{#1-0\the\month}%
587 \else
588   \xdef#1{#1-\the\month}%
589 \fi
590 \ifnum\day<10
591   \xdef#1{#1-0\the\day}%
592 \else
593   \xdef#1{#1-\the\day}%
594 \fi
595 }

```

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

```

596 \@ifundefined{pdffeedback}{%
597   \@ifundefined{pdfcreationdate}{%
    Case 1: Neither \pdffeedback nor \pdfcreationdate is defined (XeLaTeX and
    regular LATEX).
598     \hyxmp@today@define\hyxmp@today
599   }{%
    Case 2: \pdfcreationdate is defined (pdfLaTeX and pre-0.85 LuaLaTeX).
600     \edef\hyxmp@today{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%
601   }%
602 }{%
    Case 3: \pdffeedback is defined (LuaLaTeX 0.85+).
603   \edef\hyxmp@today{\expandafter\hyxmp@pdf@to@xmp@date\pdffeedback creationdate}%
604 }

```

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

```

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

```

3.5.2 The Adobe PDF schema

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

```

606 \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.

```

607   \hyxmp@add@to+xml{%

```



```

608 -----<rdf:Description rdf:about="^^J%
609 -----xmlns:pdf="http://ns.adobe.com/pdf/1.3/">^^J%
610  }%
611   \hyxmp@add@simple@var{pdf:Keywords}{@pdfkeywords}%
612   \hyxmp@add@simple@var{pdf:Producer}{@pdfproducer}%
613   \@ifundefined{pdfvariable}{%
614     \@ifundefined{pdfminorversion}{%

```

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

```

615   }{%

```

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

```

616     \hyxmp@add@simple{pdf:PDFVersion}{1.\the\pdfminorversion}%
617   }%
618   }{%

```

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

```

619     \hyxmp@add@simple{pdf:PDFVersion}{1.\the\pdfvariable minorversion}%
620   }%
621   \hyxmp@add@to@xml{%
622 -----</rdf:Description>^^J%
623   }%
624 }

```

`\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.

```

625 \newcommand*{\hyxmp@add@simple}[2]{%
626   \edef\hyxmp@string{#2}%
627   \ifx\hyxmp@string\empty
628     \else
629       \hyxmp@xmlify{\hyxmp@string}%
630       \hyxmp@add@to@xml{%
631 -----<#1>\hyxmp@xmlified</#1>^^J%
632   }%
633   \fi
634 }

```

`\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.

```

635 \newcommand*{\hyxmp@add@simple@var}[2]{%
636   \expandafter\ifx\csname#2\endcsname\relax
637     \else
638       \hyxmp@xmlify{\csname#2\endcsname}%
639       \hyxmp@add@to@xml{%

```

```

640 -----<#1>\hyxmp@xmlified</#1>^^J%
641  }%
642 \fi
643 }

```

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.

```

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

```

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

```

645   \@tempswafalse
646   \ifx#3\@empty
647   \else
648     \@tempswatrue
649   \fi
650   #1
651   \@tempswatrue
652 \fi

```

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

```

653 \if@tempswa
654   \hyxmp@xmlify{#3}%
655   \hyxmp@add@to@xml{%
656 -----<dc:#2>^^J%
657 -----<rdf:Alt>^^J%
658   }%
659   \ifx\@pdfmetalang\hyxmp@x@default
660   \else
661     \hyxmp@add@to@xml{%
662 -----<rdf:li xml:lang="\@pdfmetalang">\hyxmp@xmlified</rdf:li>^^J%
663   }%
664   \fi
665   \hyxmp@add@to@xml{%
666 -----<rdf:li xml:lang="\hyxmp@x@default">\hyxmp@xmlified</rdf:li>^^J%
667 -----</rdf:Alt>^^J%
668 -----</dc:#2>^^J%
669   }%
670 \fi
671 }%

```

`\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.

```

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

```

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

```

673  \@tempswafalse
674  \ifx#4\@empty
675  \else
676    \@tempswatrue
677  \fi
678  #1
679    \@tempswatrue
680  \fi

```

Append the corresponding XML only if \@tempswatrue.

```

681  \if@tempwa
682    \hyxmp@add@to@xml{%
683  -----<dc:#2>^^J%
684  -----<rdf:#3>^^J%
685    }%
686    \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.

```

687    \hyxmp@xmlify{#4}%
688    \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
689    \def\@elt##1{%
690      \hyxmp@add@to@xml{%
691  -----<rdf:li>##1</rdf:li>^^J%
692      }%
693    }%
694    \hyxmp@list
695    \egroup
696    \hyxmp@add@to@xml{%
697  -----</rdf:#3>^^J%
698  -----</dc:#2>^^J%
699    }%
700  \fi
701 }

```

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

```

702 \newcommand*{\hyxmp@dc@schema}{%
703   \hyxmp@add@to@xml{%
704     -----<rdf:Description rdf:about="^^J%
705     -----_xmlns:dc="http://purl.org/dc/elements/1.1/">^^J%

```

```

706 -----<dc:format>application/pdf</dc:format>^^J%
707  }%
708 \hyxmp@rdf@dc[\ifHy@pdfa]{title}{\@pdftitle}%
709 \hyxmp@rdf@dc[\ifHy@pdfa]{description}{\@pdfsubject}%
710 \hyxmp@rdf@dc{rights}{\@pdfcopyright}%
711 \hyxmp@list@to@xml[\ifHy@pdfa]{creator}{Seq}{\hyxmp@pdfauthor}%
712 \hyxmp@list@to@xml{subject}{Bag}{\hyxmp@pdfkeywords}%
713 \hyxmp@list@to@xml{date}{Seq}{\hyxmp@today}%
714 \hyxmp@list@to@xml{language}{Bag}{\@pdflang}%
715 \hyxmp@list@to@xml{type}{Bag}{\@pdftype}%
716 \hyxmp@add@simple{dc:source}{\jobname.tex}%
717 \hyxmp@add@to@xml{%
718 -----</rdf:Description>^^J%
719  }%
720 }

```

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.

```

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

```

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

```

722 \let\hyxmp@rights=\@empty
723 \ifx\@pdflicenseurl\@empty
724 \else
725 \def\hyxmp@rights{YES}%
726 \fi
727 \ifx\@pdfcopyright\@empty
728 \else
729 \def\hyxmp@rights{YES}%
730 \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.

```

731 \ifx\hyxmp@rights\@empty
732 \else

```

Header

```

733 \hyxmp@add@to@xml{%
734 -----<rdf:Description rdf:about=""^^J%
735 -----xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/">^^J%
736  }%

```

Copyright indication

```

737 \ifx\@pdfcopyright\@empty
738 \else

```

```

739      \hyxmp@add@to@xml{%
740 -----<xmpRights:Marked>True</xmpRights:Marked>^^J%
741      }%
742      \fi

License URL
743      \hyxmp@add@simple{xmpRights:WebStatement}{\@pdflicenseurl}%

Trailer
744      \hyxmp@add@to@xml{%
745 -----</rdf:Description>^^J%
746      }%
747      \fi
748 }

```

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.

```

749 \gdef\hyxmp@mm@schema{%
750   \hyxmp@def@DocumentID
751   \hyxmp@def@InstanceID
752   \hyxmp@add@to@xml{%
753 -----<rdf:Description rdf:about=""^^J%
754 -----_xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/"^^J%
755 -----<xmpMM:DocumentID>\hyxmp@DocumentID</xmpMM:DocumentID>^^J%
756 -----<xmpMM:InstanceID>\hyxmp@InstanceID</xmpMM:InstanceID>^^J%
757 -----</rdf:Description>^^J%
758   }%
759 }

```

3.5.6 The XMP Basic schema

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

```

760 \newcommand*{\hyxmp@define@createdate}{%
761   \@ifundefined{pdffeedback}{%
762     \@ifundefined{pdfcreationdate}{%

Case 1: Neither \pdffeedback nor \pdfcreationdate is defined (X $\text{\TeX}$  and
regular L $\text{\TeX}$ ).
763     \hyxmp@today@define\hyxmp@createdate
764     }%

Case 2: \pdfcreationdate is defined (pdf $\text{\TeX}$  and pre-0.85 Lua $\text{\TeX}$ ).
765     \edef\hyxmp@createdate{\expandafter\hyxmp@pdf@to@xmp@date\pdfcreationdate}%

```

```

766     }%
767 }{
  Case 3: \pdffeedback is defined (LuaLATEX 0.85+).
768     \edef\hyxmp@createdate{\expandafter\hyxmp@pdf@to@xmp@date\pdffeedback creationdate}%
769 }%
770 }

```

`\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`.

```

771 \newcommand*{\hyxmp@xmp@basic@schema}{%
772   \hyxmp@add@to+xml{%
773     -----<rdf:Description rdf:about=""^^J%
774     -----xmlns:xmp="http://ns.adobe.com/xap/1.0/">^^J%
775   }%
776   \hyxmp@define@createdate
777   \hyxmp@add@simple{xmp:CreateDate}{\hyxmp@createdate}%
778   \hyxmp@add@simple{xmp:ModifyDate}{\hyxmp@createdate}%
779   \hyxmp@add@simple{xmp:MetadataDate}{\hyxmp@createdate}%
780   \hyxmp@add@simple{xmp:CreatorTool}{\@pdfcreator}%
781   \hyxmp@add@simple{xmp:BaseURL}{\@baseurl}%
782   \hyxmp@add@to+xml{%
783     -----</rdf:Description>^^J%
784   }%
785 }

```

3.5.7 The Photoshop schema

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

```

786 \gdef\hyxmp@photoshop@schema{%
787   \edef\hyxmp@photoshop@data{\@pdfauthortitle\@pdfcaptionwriter}%
788   \ifx\hyxmp@photoshop@data\@empty
789     \else
790       \hyxmp@add@to+xml{%
791         -----<rdf:Description rdf:about=""^^J%
792         -----xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/">^^J%
793       }%
794       \fi
795       \hyxmp@add@simple{photoshop:AuthorsPosition}{\@pdfauthortitle}%
796       \hyxmp@add@simple{photoshop:CaptionWriter}{\@pdfcaptionwriter}%
797       \ifx\hyxmp@photoshop@data\@empty
798         \else
799           \hyxmp@add@to+xml{%
800             -----</rdf:Description>^^J%
801           }%
802         \fi
803 }

```

3.5.8 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.

```
804 \begingroup
805   \catcode'\&=12
806   \catcode'\#=12
807   \gdef\xmplinesep{&#xA;}
808 \endgroup
```

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

```
809 \newcommand*{\hyxmp@list@to@lines}[2]{%
810   \ifx#2\@empty
811   \else
812     \bgroup
813     \hyxmp@add@to@xml{%
814       -----<#1>%
815       }%
```

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

```
816   \def\@elt@first##1{%
817     \hyxmp@add@to@xml{##1}%
818     \let\@elt=\@elt@rest
819   }%
```

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

```
820   \def\@elt@rest##1{%
821     \hyxmp@add@to@xml{\xmplinesep##1}%
822   }%
```

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

```
823   \let\@elt=\@elt@first
824   \hyxmp@xmlify{#2}%
825   \hyxmp@commas@to@list\hyxmp@list{\hyxmp@xmlified}%
826   \hyxmp@list
827   \hyxmp@add@to@xml{</#1>^J}%
828 \egroup
829 \fi
830 }
```

`\hyxmp@photometa@schema` Add properties defined by the IPTC Photo Metadata schema [7] to the `\hyxmp@xml` macro. We currently support only the contact-information details structure, viz. the `lptc4xmpCore:CreatorContactInfo/CiAdrExtadr`, `lptc4xmpCore:CreatorContactInfo/CiAdrCity`, `lptc4xmpCore:CreatorContactInfo/CiAdrRegion`, `lptc4xmpCore:CreatorContactInfo/CiAdrPcode`,

Iptc4xmpCore:CreatorContactInfo/CiAdrCtry, Iptc4xmpCore:CreatorContactInfo/
CiTelWork, Iptc4xmpCore:CreatorContactInfo/CiEmailWork, and
Iptc4xmpCore:CreatorContactInfo/CiUrlWork properties.

```

831 \gdef\hyxmp@photometa@schema{%
832   \edef\hyxmp@photometa@data{%
833     \@pdfcontactaddress
834     \@pdfcontactcity
835     \@pdfcontactregion
836     \@pdfcontactpostcode
837     \@pdfcontactcountry
838     \@pdfcontactphone
839     \@pdfcontactemail
840     \@pdfcontacturl
841   }%
842   \ifx\hyxmp@photometa@data\@empty
843   \else
844     \hyxmp@iptc@extensions
845     \hyxmp@add@to@xml{%
846       <rdf:Description rdf:about=""^^J%
847       -----xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/">^^J%
848       -----<Iptc4xmpCore:CreatorContactInfo rdf:parseType="Resource">^^J%
849     }%
850   \fi
851   \hyxmp@list@to@lines{Iptc4xmpCore:CiAdrExtadr}{\@pdfcontactaddress}%
852   \hyxmp@add@simple{Iptc4xmpCore:CiAdrCity}{\@pdfcontactcity}%
853   \hyxmp@add@simple{Iptc4xmpCore:CiAdrRegion}{\@pdfcontactregion}%
854   \hyxmp@add@simple{Iptc4xmpCore:CiAdrPcode}{\@pdfcontactpostcode}%
855   \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” [7]. 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.

```

856   \bgroup
857   \def\xmplinesep{,%
858     \hyxmp@list@to@lines{Iptc4xmpCore:CiTelWork}{\@pdfcontactphone}%
859     \hyxmp@list@to@lines{Iptc4xmpCore:CiEmailWork}{\@pdfcontactemail}%
860     \hyxmp@list@to@lines{Iptc4xmpCore:CiUrlWork}{\@pdfcontacturl}%
861   \egroup
862   \ifx\hyxmp@photometa@data\@empty
863   \else
864     \hyxmp@add@to@xml{%
865       -----</Iptc4xmpCore:CreatorContactInfo>^^J%
866       -----</rdf:Description>^^J%
867     }%
868   \fi

```


869 }

\hyxmp@iptc@extensions Because IPTC metadata are not recognized by the PDF/A standard, PDF/A conversion would normally fail for documents that utilize \pdfcontactaddress, \pdfcontactcity, etc. However, there exists a technique, described in a PDF Association technical note [11], for describing nonstandard XMP metadata within the XMP packet itself. We use that technique here to describe all of the metadata that \hyxmp@photometa@schema can produce. Doing so enables the document to be converted to PDF/A format.

```
870 \newcommand*{\hyxmp@iptc@extensions}{%
871   \hyxmp@add@to+xml{%
872     <rdf:Description rdf:about=""^^J%
873       xmlns:pdfaExtension="http://www.aiim.org/pdfa/ns/extension/"^^J%
874       xmlns:pdfaSchema="http://www.aiim.org/pdfa/ns/schema\hyxmp@hash"^^J%
875       xmlns:pdfaProperty="http://www.aiim.org/pdfa/ns/property\hyxmp@hash"^^J%
876       xmlns:pdfaType="http://www.aiim.org/pdfa/ns/type\hyxmp@hash"^^J%
877       xmlns:pdfaField="http://www.aiim.org/pdfa/ns/field\hyxmp@hash"^^J%
878     <pdfaExtension:schemas>^^J%
879     <rdf:Bag>^^J%
880       <rdf:li rdf:parseType="Resource">^^J%
881         <pdfaSchema:schema>IPTC Core Schema</pdfaSchema:schema>^^J%
882         <pdfaSchema:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaSchema:
883         <pdfaSchema:prefix>Iptc4xmpCore</pdfaSchema:prefix>^^J%
884         <pdfaSchema:property>^^J%
885         <rdf:Seq>^^J%
886           <rdf:li rdf:parseType="Resource">^^J%
887             <pdfaProperty:name>CreatorContactInfo</pdfaProperty:name>^^J%
888             <pdfaProperty:valueType>contactinfo</pdfaProperty:valueType>^^J%
889             <pdfaProperty:category>external</pdfaProperty:category>^^J%
890             <pdfaProperty:description>contact information for the document's creator</p
891             </rdf:li>^^J%
892           </rdf:Seq>^^J%
893         </pdfaSchema:property>^^J%
894         <pdfaSchema:valueType>^^J%
895       </rdf:Seq>^^J%
896     <rdf:li rdf:parseType="Resource">^^J%
897       <pdfaType:type>contactinfo</pdfaType:type>^^J%
898       <pdfaType:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaType
899       <pdfaType:prefix>Iptc4xmpCore</pdfaType:prefix>^^J%
900       <pdfaType:description>contact information</pdfaType:description>^^J%
901       <pdfaType:field>^^J%
902     </rdf:Seq>^^J%
903   }%
904   \hyxmp@text@resource{CiAdrExtadr}{contact address}%
905   \hyxmp@text@resource{CiAdrCity}{contact city}%
906   \hyxmp@text@resource{CiAdrRegion}{contact region}%
907   \hyxmp@text@resource{CiAdrPcode}{contact postal code}%
908   \hyxmp@text@resource{CiAdrCtry}{contact country}%
909   \hyxmp@text@resource{CiTelWork}{contact telephone number}%
```

```

910 \hyxmp@text@resource{CiEmailWork}{contact email address}%
911 \hyxmp@text@resource{CiUrlWork}{contact url}%
912 \hyxmp@add@to@xml{%
913 -----</rdf:Seq>^^J%
914 -----</pdfaType:field>^^J%
915 -----</rdf:li>^^J%
916 -----</rdf:Seq>^^J%
917 -----</pdfaSchema:valueType>^^J%
918 -----</rdf:li>^^J%
919 -----</rdf:Bag>^^J%
920 -----</pdfaExtension:schemas>^^J%
921 -----</rdf:Description>^^J%
922 }%
923 }

```

`\hyxmp@text@resource` Output a single Text resource given its name and description.

```

924 \newcommand*{\hyxmp@text@resource}[2]{%
925   \hyxmp@add@to@xml{%
926 -----<rdf:li rdf:parseType="Resource">^^J%
927 -----<pdfaField:name>#1</pdfaField:name>^^J%
928 -----<pdfaField:valueType>Text</pdfaField:valueType>^^J%
929 -----<pdfaField:description>#2</pdfaField:description>^^J%
930 -----</rdf:li>^^J%
931   }
932 }

```

3.5.9 The PDF/A Identification schema

`\hyxmp@pdfa@id@schema` Add properties defined by the PDF/A Identification schema [10] 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`.

```

933 \newcommand*{\hyxmp@pdfa@id@schema}{%
934   \ifHy@pdfa
935     \hyxmp@add@to@xml{%
936 -----<rdf:Description rdf:about=""^^J%
937 -----_xmlns:pdfaid="http://www.aiim.org/pdfa/ns/id/">^^J%
938     }%
939     \hyxmp@add@simple{pdfaid:part}{\@pdfapart}%
940     \hyxmp@add@simple{pdfaid:conformance}{\@pdfaconformance}%
941     \hyxmp@add@to@xml{%
942 -----</rdf:Description>^^J%
943     }%
944   \fi
945 }

```

3.5.10 Combining schemata into an XMP packet

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

```

946 \begingroup
947   \ifhyxmp@unicodetex
948     \lccode'\!="FEFF %
949     \lowercase{%
950       \gdef\hyxmp@bom{!}
951     }%
952   \else
953     \catcode'\^^ef=12
954     \catcode'\^^bb=12
955     \catcode'\^^bf=12
956     \gdef\hyxmp@bom{\^^ef^^bb^^bf}%
957   \fi
958 \endgroup

```

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

```

959 \def\hyxmp@construct@packet{%
960   \gdef\hyxmp+xml{%
961     \hyxmp@add@to+xml{<?xpacket begin="\hyxmp@bom" %
962 id="W5M0MpCehiHzreSzNTczkc9d"?>^^J%
963 <x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="3.1-702">^^J%
964 ___<rdf:RDF
965 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\hyxmp@hash">^^J%
966   }%
967   \hyxmp@pdf@schema
968   \hyxmp@xmpRights@schema
969   \hyxmp@dc@schema
970   \hyxmp@photoshop@schema
971   \hyxmp@photometa@schema
972   \hyxmp@xmp@basic@schema
973   \hyxmp@pdfa@id@schema
974   \hyxmp@mm@schema
975   \hyxmp@add@to+xml{%
976 ___</rdf:RDF>^^J%
977 </x:xmpmeta>^^J%
978 \hyxmp@padding
979 <?xpacket end="w"?>^^J%
980   }%
981 }

```

3.6 Embedding the XMP packet

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

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

```

982 \newcommand*{\hyxmp@embed@packet}{%

```

```

983 \hyxmp@construct@packet
984 \def\hyxmp@driver{hpdfTEX}%
985 \ifx\hyxmp@driver\Hy@driver
986   \hyxmp@embed@packet@pdfTEX
987 \else
988   \def\hyxmp@driver{hLUAteX}%
989   \ifx\hyxmp@driver\Hy@driver
990     \hyxmp@embed@packet@LUAteX
991   \else
992     \def\hyxmp@driver{hdvipdfm}%
993     \ifx\hyxmp@driver\Hy@driver
994       \hyxmp@embed@packet@dvipdfm
995     \else
996       \def\hyxmp@driver{hxetex}%
997       \ifx\hyxmp@driver\Hy@driver
998         \hyxmp@embed@packet@xetex
999       \else
1000        \@ifundefined{pdfmark}{%
1001          \PackageWarningNoLine{hyperxmp}{%
1002            Unrecognized hyperref driver ‘\Hy@driver’.\MessageBreak
1003            \jobname.tex’s XMP metadata will *not* be\MessageBreak
1004            embedded in the resulting file}%
1005        }{%
1006          \hyxmp@embed@packet@pdfmark
1007        }%
1008      \fi
1009    \fi
1010  \fi
1011 \fi
1012 }

```

3.6.1 Embedding using pdfT_EX

`\hyxmp@embed@packet@pdfTEX` Embed the XMP packet using pdfT_EX primitives, which are supported by both pdfT_EX and pre-0.85 LuaT_EX.

```

1013 \newcommand*{\hyxmp@embed@packet@pdfTEX}{%
1014   \bgroup
1015     \pdfcompresslevel=0
1016     \immediate\pdfobj stream attr {%
1017       /Type /Metadata
1018       /Subtype /XML
1019     }{\hyxmp@xml}%
1020     \pdfcatalog {/Metadata \the\pdflastobj\space 0 R}%
1021   \egroup
1022 }

```

3.6.2 Embedding using LuaTeX 0.85+

If we're running Lua^ATeX 0.85+ we apparently need to use Lua directly to leave uncompressed the PDF object holding the XMP packet. The `luacode` package facilitates the inclusion of Lua code in a LaTeX document.

```
1023 \expandafter\ifx\csname pdffeedback\endcsname\relax
1024 \else
1025   \RequirePackage{luacode}%
```

`hyxmp_embed_packet` We define a Lua function, `hyxmp_embed_packet`, that embeds a string as an uncompressed PDF object of type `Metadata`.

```
1026   \begin{luacode*}
1027     function hyxmp_embed_packet (contents)
1028       local n = pdf.obj {
1029         type = "stream",
1030         attr = [[
1031 /Type /Metadata
1032 /Subtype /XML
1033 ]],
1034         compresslevel = 0,
1035         string = contents
1036       }
1037       pdf.refobj(n)
1038     end
1039   \end{luacode*}
1040 \fi
```

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

```
1041 \newcommand*{\hyxmp@embed@packet@luatex}{%
1042   \luadirect{hyxmp_embed_packet(\luastring{\hyxmp@xml})}%
1043   \pdfextension catalog {/Metadata \the\numexpr\pdffeedback lastobj\relax\space 0 R}%
1044 }
```

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.

```
1045 \newcommand*{\hyxmp@embed@packet@pdfmark}{%
1046   \pdfmark{%
1047     pdfmark=/NamespacePush
1048   }%
1049   \pdfmark{%
1050     pdfmark=/OBJ,
1051     Raw={/_objdef \string{hyxmp@Metadata\string} /type /stream}%
1052   }%
1053   \pdfmark{%
1054     pdfmark=/PUT,
1055     Raw={\string{hyxmp@Metadata\string}
```

```

1056      2 dict begin
1057        /Type /Metadata def
1058        /Subtype /XML def
1059        currentdict
1060      end
1061    }%
1062  }%
1063  \pdfmark{%
1064    pdfmark=/PUT,
1065    Raw={\string{hyxmp@Metadata\string} (\hyxmp+xml)}%
1066  }%
1067  \pdfmark{%
1068    pdfmark=/Metadata,
1069    Raw={\string{Catalog\string} \string{hyxmp@Metadata\string}}%
1070  }%
1071  \pdfmark{%
1072    pdfmark=/NamespacePop
1073  }%
1074 }

```

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.

```

1075 \newcommand*{\hyxmp@embed@packet@dvipdfm}{%
1076   \hyxmp@string@len{\hyxmp+xml}%
1077   \special{pdf: object @hyxmp@Metadata
1078     <<
1079       /Type /Metadata
1080       /Subtype /XML
1081       /Length \the\@tempcnta
1082     >>
1083     stream^^J\hyxmp+xml endstream%
1084   }%
1085   \special{pdf: docview
1086     <<
1087       /Metadata @hyxmp@Metadata
1088     >>
1089   }%
1090 }

```

`\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.

```

1091 \newcommand*{\hyxmp@string@len}[1]{%
1092   \@tempcnta=0

```

```

1093 \expandafter\hyxmp@count@spaces#1 {} %
1094 \expandafter\hyxmp@count@non@spaces#1{}%
1095 }

```

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

```

1096 \def\hyxmp@count@spaces#1 {%
1097   \def\hyxmp@one@token{#1}%
1098   \ifx\hyxmp@one@token\@empty
1099     \advance\@tempcnta by -1
1100   \else
1101     \advance\@tempcnta by 1
1102     \expandafter\hyxmp@count@spaces
1103   \fi
1104 }

```

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

```

1105 \newcommand*{\hyxmp@count@non@spaces}[1]{%
1106   \def\hyxmp@one@token{#1}%
1107   \ifx\hyxmp@one@token\@empty
1108   \else
1109     \advance\@tempcnta by 1
1110     \expandafter\hyxmp@count@non@spaces
1111   \fi
1112 }

```

3.6.5 Embedding using X_qT_EX

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

```

1113 \newcommand*{\hyxmp@embed@packet@xetex}{%
1114   \special{pdf:stream @hyxmp@Metadata (\hyxmp@xml)
1115     <<
1116       /Type /Metadata
1117       /Subtype /XML
1118     >>
1119   }%
1120   \special{pdf:put @catalog
1121     <<
1122       /Metadata @hyxmp@Metadata
1123     >>
1124   }%
1125 }

```

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.

```
1126 \catcode'\="=\hyxmp@dq@code
```

4 Future Work

Help wanted 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 , \XeTeX , 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 5–6. 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/" x:xmptk="3.1-702">
  <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/"
      <pdf:Keywords>
        energy quanta, Hertz effect, quantum physics
      </pdf:Keywords>
      <pdf:Producer>pdfTeX-1.40.10</pdf:Producer>
    </rdf:Description>
    <rdf:Description rdf:about=""
      xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/"
      <xmpRights:Marked>True</xmpRights:Marked>
      <xmpRights:WebStatement>
        http://creativecommons.org/licenses/by-nc-nd/3.0/
      </xmpRights:WebStatement>
    </rdf:Description>
    <rdf:Description rdf:about=""
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      <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: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: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>
</rdf:Description>
<rdf:Description rdf:about=""
  xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/">
  <photoshop:AuthorsPosition>
    Technical Assistant, Level III
  </photoshop:AuthorsPosition>
  <photoshop:CaptionWriter>Scott Pakin</photoshop:CaptionWriter>
</rdf:Description>
<rdf:Description rdf:about=""
  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>
    <rdf:li rdf:parseType="Resource">
      <pdfaSchema:schema>IPTC Core Schema</pdfaSchema:schema>
      <pdfaSchema:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaSchema:namespaceURI>
      <pdfaSchema:prefix>Iptc4xmpCore</pdfaSchema:prefix>
      <pdfaSchema:property>
        <rdf:Seq>
          <rdf:li rdf:parseType="Resource">
            <pdfaProperty:name>CreatorContactInfo</pdfaProperty:name>
            <pdfaProperty:valueType>contactinfo</pdfaProperty:valueType>
            <pdfaProperty:category>external</pdfaProperty:category>
            <pdfaProperty:description>contact information for the document's
          </rdf:li>
        </rdf:Seq>
      </pdfaSchema:property>
      <pdfaSchema:valueType>
        <rdf:Seq>
          <rdf:li rdf:parseType="Resource">
            <pdfaType:type>contactinfo</pdfaType:type>
            <pdfaType:namespaceURI>http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/</pdfaType:namespaceURI>
            <pdfaType:prefix>Iptc4xmpCore</pdfaType:prefix>
            <pdfaType:description>contact information</pdfaType:description>
            <pdfaType:field>
              <rdf:Seq>

```

```

<rdf:li rdf:parseType="Resource">
  <pdfaField:name>CiAdrExtadr</pdfaField:name>
  <pdfaField:valueType>Text</pdfaField:valueType>
  <pdfaField:description>contact address</pdfaField:description>
</rdf:li>
<rdf:li rdf:parseType="Resource">
  <pdfaField:name>CiAdrCity</pdfaField:name>
  <pdfaField:valueType>Text</pdfaField:valueType>
  <pdfaField:description>contact city</pdfaField:description>
</rdf:li>
<rdf:li rdf:parseType="Resource">
  <pdfaField:name>CiAdrRegion</pdfaField:name>
  <pdfaField:valueType>Text</pdfaField:valueType>
  <pdfaField:description>contact region</pdfaField:description>
</rdf:li>
<rdf:li rdf:parseType="Resource">
  <pdfaField:name>CiAdrPcode</pdfaField:name>
  <pdfaField:valueType>Text</pdfaField:valueType>
  <pdfaField:description>contact postal code</pdfaField:description>
</rdf:li>
<rdf:li rdf:parseType="Resource">
  <pdfaField:name>CiAdrCtry</pdfaField:name>
  <pdfaField:valueType>Text</pdfaField:valueType>
  <pdfaField:description>contact country</pdfaField:description>
</rdf:li>
<rdf:li rdf:parseType="Resource">
  <pdfaField:name>CiTelWork</pdfaField:name>
  <pdfaField:valueType>Text</pdfaField:valueType>
  <pdfaField:description>contact telephone number</pdfaField:description>
</rdf:li>
<rdf:li rdf:parseType="Resource">
  <pdfaField:name>CiEmailWork</pdfaField:name>
  <pdfaField:valueType>Text</pdfaField:valueType>
  <pdfaField:description>contact email address</pdfaField:description>
</rdf:li>
<rdf:li rdf:parseType="Resource">
  <pdfaField:name>CiUrlWork</pdfaField:name>
  <pdfaField:valueType>Text</pdfaField:valueType>
  <pdfaField:description>contact url</pdfaField:description>
</rdf:li>
</rdf:Seq>
</pdfaType:field>
</rdf:li>
</rdf:Seq>
</pdfaSchema:valueType>
</rdf:li>

```

```

        </rdf:Bag>
    </pdfaExtension:schemas>
</rdf:Description>
<rdf:Description rdf:about=""
    xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/">
    <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>
</rdf:Description>
<rdf:Description rdf:about=""
    xmlns:xmp="http://ns.adobe.com/xap/1.0/">
    <xmp:CreateDate>2016-07-04T11:00:36-06:00</xmp:CreateDate>
    <xmp:ModifyDate>2016-07-04T11:00:36-06:00</xmp:ModifyDate>
    <xmp:MetadataDate>2016-07-04T11:00:36-06:00</xmp:MetadataDate>
    <xmp:CreatorTool>LaTeX with hyperref package</xmp:CreatorTool>
    <xmp:BaseURL>
        http://mirror.ctan.org/macros/latex/contrib/hyperxmp/
    </xmp:BaseURL>
</rdf:Description>
<rdf:Description rdf:about=""
    xmlns:xmpMM="http://ns.adobe.com/xap/1.0/mm/">
    <xmpMM:DocumentID>
        uuid:0595fdce-41dc-e4c4-6c418dc4ce46
    </xmpMM:DocumentID>
    <xmpMM:InstanceID>
        uuid:efd754c4-1d7f-200a-ef754ce413ea
    </xmpMM:InstanceID>
</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 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.
- [8] Internet Assigned Numbers Authority. Language subtag registry, January 11, 2011. Available from <http://www.iana.org/assignments/language-subtag-registry>.
- [9] 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>.
- [10] PDF/A Competence Center, Berlin, Germany. *TechNote 0008: Pre-defined XMP Properties in PDF/A-1*, March 20, 2008. Available from http://www.pdfa.org/wp-content/uploads/2011/08/tn0008_predefined_xmp_properties_in_pdfa-1_2008-03-20.pdf.
- [11] 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.

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

Change History

v1.0	General: Initial version	1	v1.5	General: Made the XMP inclusion more robust. Thanks to Heiko Oberdiek for the bug report and suggested modifications.	11
v1.1	<code>\hyxmp@construct@packet</code> : Explicitly set the category codes of characters <code><EF></code> , <code><BB></code> , and <code><BF></code> to “letter”. Thanks to Daniel Schömer for the bug report	43	v2.0	General: Added support for the XMP Basic schema and miscellaneous other bits of metadata	1
v1.2	General: Added support for the X _Y TeX backend (<code>xdvipdfmx</code>)	1		Heiko Oberdiek’s major rewrite of the code to better support native-Unicode T _E X implementations (X _Y TeX and LuaT _E X)	1
	Added support for the Photoshop schema	1		New <code>\AtBeginDocument</code> code from Heiko Oberdiek to properly encode <code>\@pdfmetalang</code>	17
	Made the package compatible with <code>ngerman</code> . Thanks to Tobias Mueller for the bug report.	11		<code>\hyxmp@add@to@xml</code> : Updated also to replace commas	30
v1.3	General: Introduced the <code>pdfmetalang</code> package option, which enables an author to specify the language in which he wrote the document’s metadata	17		<code>\hyxmp@bom</code> : Added by Heiko Oberdiek	42
	<code>\hyxmp@reencode</code> : Introduced this macro to re-encode Unicode strings as 8-bit strings before manipulating them into XMP schema. This change addresses a bug reported by Martin Münch	21		<code>\hyxmp@comma</code> : Added this macro	19
v1.4	<code>\hyxmp@mm@schema</code> : Renamed the <code>xapMM</code> namespace prefix to <code>xmpMM</code>	37		<code>\hyxmp@construct@packet</code> : Modified by Heiko Oberdiek to use an appropriate BOM representation via <code>\hyxmp@bom</code>	43
	<code>\hyxmp@rdf@dc</code> : Included metadata in the <code>x-default</code> language regardless of the specified metadata language	34		<code>\hyxmp@crap@convert</code> : Added by Heiko Oberdiek	26
	<code>\hyxmp@xmpRights@schema</code> : Renamed the <code>xapRights</code> namespace prefix to <code>xmpRights</code>	36		<code>\hyxmp@crap@test</code> : Added by Heiko Oberdiek	25
				<code>\hyxmp@dc@schema</code> : Added support for <code>dc:language</code> and <code>dc:source</code>	35
				<code>\hyxmp@is@unicode</code> : Added by Heiko Oberdiek	23
				<code>\hyxmp@list@to@xml</code> : Modified by Heiko Oberdiek to use the new Unicode-processing macros	34
				<code>\hyxmp@photoshop@schema</code> : Simplified using <code>\hyxmp@add@simple</code>	38
				<code>\hyxmp@ProcessKeyvalOptions</code> : Added this macro	16

\hyxmp@reencode: Replaced with an empty macro by Heiko Oberdiek	21	\hyxmp@redefine@Hyp: Added this macro	14
\hyxmp@skiptorelax: Added by Heiko Oberdiek	25	v2.2	
\hyxmp@skipzeros: Added by Heiko Oberdiek	24	General: Added support for the IPTC Photo Metadata schema	1
\hyxmp@SpaceOther: Added by Heiko Oberdiek	25	\hyxmp@iptc@extensions: Added this macro to support PDF/A generation	41
\hyxmp@string: Added this macro	33	\hyxmp@list@to@lines: Added this macro	39
\hyxmp@toxml: Added by Heiko Oberdiek	23	\hyxmp@photometa@schema: Added this macro	39
Escaped parentheses written with pdfmarks to prevent dvips from line-wrapping the XMP packet	23	\hyxmp@text@resource: Added this macro	42
\hyxmp@toxml@unicodetex: Added by Heiko Oberdiek	24	\xmpcomma: Changed the default from \relax to an ordinary comma	19
\hyxmp@xetex@crap: Added by Heiko Oberdiek	24	\xmplinesep: Added this macro	39
\hyxmp@xmllify: Completely rewritten by Heiko Oberdiek to better support Unicode-enabled T _E X programs	21	v2.3	
\hyxmp@xmp@basic@schema: Added this macro	38	\hyxmp@iptc@extensions: Gave the lptc4xmpCore:CreatorContactInfo fields a unique pdfaType:prefix to better support conversion of the document to PDF/A	41
\hyxmp@xmpRights@schema: Modified to include xmpRights:Marked only when pdfcopyright is specified and xmpRights:WebStatement only when pdflicenseurl is specified	36	v2.3a	
\hyxmp@zero: Added by Heiko Oberdiek	26	General: Bug fix: Redefine \@pdflang as \@empty when hyperref has set it to \relax ..	17
\ifhyxmp@unicodetex: Added by Heiko Oberdiek	21	v2.3b	
\ProcessKeyvalOptions: Added this macro	16	\XMPTruncateList: Made all definitions local to avoid spurious Too many unprocessed floats errors when running with memoir ..	20
\xmpcomma: Added this macro	19	v2.4	
\xmpquote: Added this macro	19	General: Added support for the PDF/A Identification schema, as requested by Florian Breitwieser	1
\XMPTruncateList: Added this macro	20	\hyxmp@add@simple@var: Added this macro	33
v2.1		\hyxmp@create@uuid: Modified this macro to produce a proper version 4 (random or pseudorandom) UUID	28
General: Enabled hyperxmp and hyperref to be loaded in either order. This addresses a bug report by Yury Donskoy	14	\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	35
\hypersetup: Added this macro	16		
\hyxmp@hypersetup: Added this macro	16		

<code>\hyxmp@parse@time</code> : Added this macro	31	v2.8	<code>\hyxmp@add@to@xml</code> : Corrected inadvertent lowercasing of non-Latin characters when run under \TeX or \LaTeX (bug reported by Leonid Sinev) . . .	30
<code>\hyxmp@parse@tz</code> : Added this macro	31			
<code>\hyxmp@parse@tz@char</code> : Added this macro	31			
<code>\hyxmp@pdf@schema</code> : Made <code>\hyxmp@pdf@schema</code> <i>always</i> include the Adobe PDF schema, even when empty. Florian Breitwieser noted that this is necessary for PDF/A-1b compliance	32	v2.9	General: Force inclusion of <code>dc:creator</code> , <code>dc:title</code> , and <code>dc:description</code> —even if empty—when <code>hyperref</code> is loaded with the <code>pdfa</code> option (suggested by Leonid Sinev)	1
<code>\hyxmp@pdf@to@xmp@date</code> : Added this macro	31		Introduced the <code>pdftype</code> package option, which enables an author to specify the type of document being produced	1
<code>\hyxmp@pdfa@id@schema</code> : Added this macro	42		<code>\hyxmp@pdfa@id@schema</code> : Let the author specify the PDF/A part and conformance IDs, as requested by Leonid Sinev	42
<code>\hyxmp@today</code> : Modified the code to parse the time and timezone from <code>\pdfcreationdate</code> , as proposed by Florian Breitwieser	32		<code>\hyxmp@photometa@schema</code> : Use <code>lptc4xmpCore</code> instead of <code>lptc4ContInfo</code> as the contact-information metadata prefix. Leonid Sinev reports that Acrobat’s PDF/A validator seems to prefer <code>lptc4xmpCore</code>	39
<code>\hyxmp@today@define</code> : Added this macro	31			
<code>\xmp tilde</code> : Added this macro	20			
v2.5				
General: Enabled “ <code>_</code> ” to work within email addresses, as requested by Leonid Sinev	1			
<code>\hyxmp@add@to@xml</code> : Updated also to replace underscores and to modify only the text being added, not the already-modified text	30	v3.0	General: Made the code compatible with \LaTeX 0.85. Thanks to Robert Schlicht, Leonid Sinev, and David Carlisle for bug reports and to Leonid Sinev for helping test the new <code>hyperxmp</code> code	1
<code>\hyxmp@textunderscore</code> : Added this macro	12		<code>\hyxmp@embed@packet@luatex</code> : Added this macro	45
<code>\hyxmp@uscore</code> : Added this macro	19		<code>\hyxmp@today@define</code> : Modified to accept the name of a macro to define	31
v2.6			<code>\hyxmp@xmp@basic@schema</code> : Made the <code>XMP CreateDate</code> , <code>ModifyDate</code> , and <code>MetadataDate</code> match the PDF <code>CreationDate</code>	38
General: Added support for a new <code>pdfdate</code> key to explicitly specify the document date (and optionally time)	1		<code>hyxmp_embed_packet</code> : Added this Lua function	45
v2.7				
General: Automatically use <code>\title</code> and <code>\author</code> if <code>pdftitle</code> and <code>pdfauthor</code> are left unspecified. Thanks to Maciej Radziejewski for the suggestion	17			

Index

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

Symbols		
\#	547, 806	
\&	297, 329, 805	
\,	540	
\@author	166, 168	
\@baseurl	115, 781	
\@elt	<u>213</u> , <u>687</u> , 818, <u>823</u>	
\@elt@first	<u>816</u>	
\@elt@rest	818, <u>820</u>	
\@firstofone	376	
\@firstoftwo	288	
\@gobble	218, 373	
\@pdfaformance <u>37</u>	940	
\@pdfapart	<u>35</u> , 939	
\@pdfauthor	<u>70</u> , 116, 165, 515, 521	
\@pdfauthortitle	<u>29</u> , 117, 787, 795	
\@pdfcaptionwriter	<u>31</u> , 118, 787, 796	
\@pdfcontactaddress	<u>39</u> , 119, 833, 851	
\@pdfcontactcity	<u>47</u> , 120, 834, 852	
\@pdfcontactcountry	<u>53</u> , 121, 837, 855	
\@pdfcontactemail	<u>57</u> , 122, 839, 859	
\@pdfcontactphone	<u>55</u> , 123, 838, 858	
\@pdfcontactpostcode	<u>51</u> , 124, 836, 854	
\@pdfcontactregion	<u>49</u> , 125, 835, 853	
\@pdfcontacturl	<u>59</u> , 126, 840, 860	
\@pdfcopyright	<u>23</u> , 127, 710, 727, 737	
\@pdfcreator	780	
\@pdfdatetime	<u>21</u> , 129, 155, 157	
\@pdfkeywords	<u>91</u> , 130	
\@pdflang	131, 146, 147, 149, 152, 714	
\@pdflicenseurl	<u>27</u> , 132, 723, 743	
\@pdfmetalang <u>33</u>	150, 152, 154, 659, 662	
\@pdfsubject	133, 709	
\@pdftitle	134, 159, 515, 521, 708	
\@pdftype	<u>25</u> , 128, 715	
\@secondoftwo	290	
\@tempswafalse	645, 673	
\@tempswatrue	648, 651, 676, 679	
\@title	160, 162	
\^	201, 205, 233, 540, 541, 953–955	
_	539, 541	
\~	210, 367, 368	
_	333, 368, 539	
A		
\and	<u>70</u>	
ASCII	12, 22	
\AtBeginDocument	144	
\AtEndDocument	5	
\AtEndDvi	8	
atendddvi	11	
Author	7, 8, 20	
B		
Bag	55	
baseurl (option)	3, 10, 11, 38	
BOM	42, 54	
C		
CiAdrCity	2, 39	
CiAdrCtry	2, 40	
CiAdrExtadr	2, 39	
CiAdrPcode	2, 39	
CiAdrRegion	2, 39	
CiEmailWork	2, 40	
CiTelWork	2, 40	
CiUrlWork	2, 40	
CreateDate	37, 56	
CreationDate	56	
D		
Date	32	
\day	590, 591, 593	
dc:creator	2, 8, 35, 56	
dc:date	2, 35	
dc:description	3, 35, 56	
dc:format	2	
dc:language	2, 35, 54, 55	
dc:rights	2, 35	
dc:source	2, 35, 54	
dc:subject	2, 35	
dc:title	3, 35, 56	
dc:type	2	
\define@key 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 48, 50, 52, 54, 56, 58, 60, 70, 91		
dvipdf (option)	45	
dvipdfm	46	
dvips (option)	45	
dvips	6, 7, 23, 55	
dvipsone (option)	45	
dviwindo (option)	45	
E		
\EdefEscapeHex	256, 269	
\EdefUnescapeHex	273	
\EdefUnescapeString	243	
ETX	19, 21	
G		
Ghostscript	7	
H		
\Hy@driver	4, 985, 989, 993, 997, 1002	

hyperref 1, 3–5, 8, 10–12, 14, 16– 18, 32, 43, 45, 55, 56	\hyxmp@construct@packet 959, 983	\hyxmp@is@unicode 245, 262, 277
\hypersetup 108, 162, 168	\hyxmp@count@non@spaces 1094, 1105	\hyxmp@legal 722
hyperxmp 1–5, 8–14, 17–19, 21, 27, 32, 48, 55, 56	\hyxmp@count@spaces 1093, 1096	\hyxmp@list 688, 694, 825, 826
\hyxmp@is@unicode . 277	\hyxmp@crap@convert 359, 393	\hyxmp@list@to@lines . 809, 851, 858–860
\hyxmp@add@simple . . . 616, 619, 625, 716, 743, 777– 781, 795, 796, 852–855, 939, 940	\hyxmp@crap@result 349, 385	\hyxmp@list@to@xml 672, 711–715
\hyxmp@add@simple@var 611, 612, 635	\hyxmp@crap@test 356, 381	\hyxmp@mm@schema 749, 974
\hyxmp@add@to@xml 526, 552, 607, 621, 630, 639, 655, 661, 665, 682, 690, 696, 703, 717, 733, 739, 744, 752, 772, 782, 790, 799, 813, 817, 821, 827, 845, 864, 871, 912, 925, 935, 941, 961, 975	\hyxmp@create@uuid 490, 518, 524	\hyxmp@modulo@a 431, 450, 460, 466, 501
\hyxmp@add@simple@var 611, 612, 635	\hyxmp@createdate 760, 777–779	\hyxmp@new@xml 542, 543
\hyxmp@add@to@xml 526, 552, 607, 621, 630, 639, 655, 661, 665, 682, 690, 696, 703, 717, 733, 739, 744, 752, 772, 782, 790, 799, 813, 817, 821, 827, 845, 864, 871, 912, 925, 935, 941, 961, 975	\hyxmp@dc@schema 702, 969	\hyxmp@num 393
\hyxmp@and 70	\hyxmp@def@DocumentID 514, 750	\hyxmp@one@token . . . 439, 443, 1097, 1098, 1106, 1107
\hyxmp@append@hex . . . 463, 482–484, 488	\hyxmp@def@InstanceID 520, 751	\hyxmp@padding 550, 978
\hyxmp@append@hex@iii . 481, 487, 497, 508	\hyxmp@define@createdate 760, 776	\hyxmp@parse@time 564, 566
\hyxmp@append@hex@iv 486, 492, 493, 495, 510–512	\hyxmp@DocumentID 514, 755	\hyxmp@parse@tz 573, 576, 580
\hyxmp@at@end . . . 3, 171	\hyxmp@dq@code . 1, 1126	\hyxmp@parse@tz@char 568, 570
\hyxmp@big@prime 437, 440, 450, 460	\hyxmp@driver . . . 3, 982	\hyxmp@pdf@schema 606, 967
\hyxmp@big@prime@ii 437, 459	\hyxmp@embed@packet 173, 982	\hyxmp@pdf@to@xmp@date 562, 600, 603, 765, 768
\hyxmp@bom 946, 961	\hyxmp@embed@packet@dvi 994, 1075	\hyxmp@pdfa@id@schema 933, 973
\hyxmp@comma 41, 71, 92, 200	\hyxmp@embed@packet@luatex 990, 1041	\hyxmp@pdfauthor 61, 70, 711
\hyxmp@commas@to@list . 184, 215, 688, 825	\hyxmp@embed@packet@pdfmark 1006, 1045	\hyxmp@pdfkeywords 61, 91, 712
\hyxmp@commas@to@list@i 186, 188	\hyxmp@embed@packet@pdftex 986, 1013	\hyxmp@pdfstringdef 15, 22, 24, 26, 28, 30, 32, 34, 36, 38, 43, 48, 50, 52, 54, 56, 58, 60
\hyxmp@concat@metadata 113	\hyxmp@embed@packet@xetex 998, 1113	\hyxmp@photometa@data 831
	\hyxmp@find@metadata 113, 172	\hyxmp@photometa@schema 831, 971
	\hyxmp@hash 546, 874–877, 965	\hyxmp@photoshop@data 786
	\hyxmp@Hyp@pdfauthor 64	\hyxmp@photoshop@schema 786, 970
	\hyxmp@Hyp@pdfkeywords 85	\hyxmp@ProcessKeyvalOptions 103
	\hyxmp@hypersetup . . 108	
	\hyxmp@InstanceID 520, 756	
	\hyxmp@iptc@extensions 844, 870	

<code>\hyxmp@rand@num</code> 456 , 465, 500, 517, 523	<code>\hyxmp@unicodetextrue</code> 232	<code>\KV@Hyp@pdfauthor</code> .. 70
<code>\hyxmp@rdf@dc</code> 644 , 708–710	<code>\hyxmp@uscore</code> .. 17, 204	<code>\KV@Hyp@pdfkeywords</code> 91
<code>\hyxmp@redefine@Hyp</code> 63 , 105, 110	<code>\hyxmp@x@default</code> 150, 605, 659, 666	<code>kvoptions</code> 12, 16
<code>\hyxmp@reencode</code> ... 238	<code>\hyxmp@xetex@crap</code> 250, 349	L
<code>\hyxmp@rights</code> 722, 725, 729, 731	<code>\hyxmp+xml</code> 543, 550 , 959 , 1019, 1042, 1065, 1076, 1083, 1114	<code>LF</code> 39
<code>\hyxmp@seed@rng</code> 439 , 516, 522	<code>\hyxmp+xmlified</code> 241 , 631, 640, 662, 666, 688, 825	<code>Lua</code> 45, 56
<code>\hyxmp@seed@rng@i</code> 441, 443	<code>\hyxmp+xmlify</code> 154, 241 , 629, 638, 654, 687, 824	<code>luacode</code> 45
<code>\hyxmp@seed@string</code> . . 515, 516, 521, 522	<code>\hyxmp@xmp@basic@schema</code> 771 , 972	<code>\luadirect</code> 1042
<code>\hyxmp@set@rand@num</code> 456 , 464, 499	<code>\hyxmp@xmpRights@schema</code> 721 , 968	<code>Lua^AT_EX</code> 6, 9, 32, 33, 37, 38, 45, 56
<code>\hyxmp@skiptorelax</code> 386, 392	<code>\hyxmp@zero</code> 402, 409, 416, 422, 427	<code>\luastring</code> 1042
<code>\hyxmp@skipzeros</code> ... 344	<code>hyxmp_embed_packet</code> 1026	<code>LuaT_EX</code> 21, 24, 44, 45, 48, 54, 56
<code>\hyxmp@SpaceOther</code> 353, 366	I	M
<code>\hyxmp@string</code> 625	<code>IETF</code> 5	<code>memoir</code> 55
<code>\hyxmp@string@len</code> 1076, 1091	<code>\if@tempswa</code> ... 653, 681	<code>Metadata</code> .. 7, 43, 45, 47
<code>\hyxmp@sublist</code> 189, 190, 193, 194	<code>\iffalse</code> 644, 672	<code>MetadataDate</code> 37, 56
<code>\hyxmp@temp@list</code> ... 213	<code>\ifHy@pdfa</code> 708, 709, 711, 934	<code>ModifyDate</code> 37, 56
<code>\hyxmp@temp@str</code> ... 213	<code>\ifhyxmp@unicodetex</code> . 232 , 244, 529, 947	<code>\month</code> 585, 586, 588
<code>\hyxmp@text</code> 241 , 319 , 349 , 393	<code>ifxetex</code> 12	N
<code>\hyxmp@text@resource</code> 904–911, 924	<code>\ifxetex</code> 249	<code>NAK</code> 12, 19, 21
<code>\hyxmp@textunderscore</code> 15	<code>Info</code> 7	<code>nativepdf (option)</code> ... 45
<code>\hyxmp@today</code> 157, 521, 596 , 713	<code>intcalc</code> 12	<code>\newif</code> 232
<code>\hyxmp@today@define</code> 583 , 598, 763	<code>\intcalcDiv</code> 398, 405, 412	<code>\next</code> 188 , 443
<code>\hyxmp@toxml</code> .. 271, 294	<code>\intcalcMod</code> 400, 407, 414	<code>ngerman</code> 11, 54
<code>\hyxmp@toxml@unicodetex</code> 259, 319	<code>IPTC</code> . 9, 13, 29, 39–41, 55	<code>\number</code> 396, 398, 400, 405, 407, 412, 414
<code>\hyxmp@trimb</code> .. 226, 229	<code>lptc4xmpCore:CreatorContactInfo</code> 2, 39, 40, 55	<code>\numexpr</code> 1043
<code>\hyxmp@trimc</code> .. 229, 230	<code>ISO</code> 13	O
<code>\hyxmp@trimspaces</code> 193, 222	J	<code>options</code>
<code>\hyxmp@try</code> 349	<code>\jobname</code> .. 138, 177, 515, 521, 716, 1003	<code>baseurl</code> . 3, 10, 11, 38
<code>\hyxmp@unicodetexfalse</code> 232	K	<code>dvipdf</code> 45
	<code>Keywords</code> 7, 32	<code>dvips</code> 45
		<code>dvipsone</code> 45
		<code>dviwindo</code> 45
		<code>nativepdf</code> 45
		<code>pdfa</code> 4, 56
		<code>pdfaconformance</code> 4, 42
		<code>pdfapart</code> 4, 42
		<code>pdfauthor</code> .. 3, 10, 11, 14, 15, 17, 35, 56
		<code>pdfauthortitle</code> ... 4, 10
		<code>pdfcaptionwriter</code> ... 4
		<code>pdfcontactaddress</code> 4, 9
		<code>pdfcontactcity</code> .. 4, 5
		<code>pdfcontactcountry</code> 4, 5

UTF-8	22	32, 34, 35, 39, 40, 43	xmpMM:DocumentID	2, 26, 37
UUID	26, 28, 29, 55	XMP 1–3, 5, 7–11, 17–20, 23, 26, 29–33, 36–38, 41, 44–48, 54–56	xmpMM:InstanceID	2, 26, 37
V				
\vfuzz	230	xmp:BaseURL	2	\xmpquote
vtexpdfmark (option)	45	xmp:CreateDate	2	42, 45, <u>70</u> , <u>91</u> , <u>208</u>
X				
\x	<u>349</u>	xmp:CreatorTool	3	xmpRights:Marked 2, 36, 55
xdvipdfmx	9, 47	xmp:MetadataDate	2	xmpRights:WebStatement
X _Y LaTeX 6, 9, 32, 33, 37, 56		xmp:ModifyDate	2	2, 36, 55
X _Y TeX 12, 21, 24, 47, 48, 54		\xmpcomma		\xmptilde
XML	1, 2, 9, 18, 21–24, 29, 30,	41, 44, <u>70</u> , <u>91</u> , <u>199</u>		<u>209</u>
		xmpincl	3	\XMPTruncateList
		\xmplinesep <u>804</u> , 821, <u>856</u>		<u>213</u>
Y				
		\year	584	