

The **l3pdfmeta** module  
PDF standards  
**L**A<sub>T</sub>E<sub>X</sub> PDF management testphase bundle

The L<sup>A</sup>T<sub>E</sub>X Project\*

Version 0.96r, released 2025-05-15

## 1 **l3pdfmeta** documentation

This module sets up some tools and commands needed for PDF standards in general. The goal is to collect the requirements and to provide code to check and fulfill them.

### 1.1 Verifying requirements of PDF standards

Standards like pdf/A set requirements on a PDF: Some things have to be in the PDF, e.g. the catalog has to contain a `/Lang` entry and a colorprofile and an `/OutputIntent`, some other things are forbidden or restricted, e.g. the action dictionary of an annotation should not contain Javascript.

The `l3pdfmeta` module collects a number of relevant requirements, tries to enforce the ones which can be enforced and offers some tools for package authors to test if an action is allowed in the standard or not.

This is work in progress and more tests will be added. But it should be noted that it will probably never be possible to prevent all forbidden actions or enforce all required ones or even to simply check all of them. The commands here don't replace a check with an external validator.

Verifying against a PDF-standard involves two different tasks:

- Check if you are allowed to ignore the requirement.
- Decide which action to take if the answer to the first question is NO.

The following conditionals address the first task. Because of the second task a return value `FALSE` means that the standard requires you to do some special action. `TRUE` means that you can ignore this requirement.<sup>1</sup>

In most cases it only matters if a requirement is in the standard, for example `Catalog_no_OCProperties` means “don't use `/OCProperties` in the catalog”. For a small number of requirements it is also needed to test a user value against a standard value. For example, `named_actions` restricts the allowed named actions in an annotation

---

\*E-mail: [latex-team@latex-project.org](mailto:latex-team@latex-project.org)

<sup>1</sup>One could also make the logic the other way round—there are arguments for both—but I had to decide.

of subtype `/Named`, in this case it is needed to check not only if the requirement is in the standard but also if the user value is in the allowed list.

---

```
\pdfmeta_standard_verify_p:n * \pdfmeta_standard_verify:n{<requirement>}
\pdfmeta_standard_verify:nTF *
```

---

This checks if `<requirement>` is listed in the standard. `FALSE` as result means that the requirement is in the standard and that probably some special action is required—which one depends on the requirement, see the descriptions below. `TRUE` means that the requirement is not there and so no special action is needed. This check can be used for simple requirements where neither a user nor a standard value is of importance.

---

```
\pdfmeta_standard_verify:nnTF \pdfmeta_standard_verify:nn{<requirement>}{<value>}
```

---

This checks if `<requirement>` is listed in the standard, if yes it tries to find a pre-defined test handler for the requirement and passes `<value>` and the value recorded in the standard to it. The handler returns `FALSE` if some special action is needed (e.g. if `<value>` violates the rule) and `TRUE` if no special action is needed. If no handler exists this commands works like `\pdfmeta_standard_verify:n`.

In some cases one needs to query the value in the standard, e.g. to correct a wrong minimal PDF version you need to know which version is required by `min_pdf_version`. For this two commands to access the value are provided:

---

```
\pdfmeta_standard_item:n * \pdfmeta_standard_item:n{<requirement>}
```

---

This retrieves the value of `<requirement>` and leaves it in the input. If the requirement isn't in the standard the result is empty, that means that requirements not in the standard and requirement without values can not be distinguished here.

---

```
\pdfmeta_standard_get:nN \pdfmeta_standard_get:nN{<requirement>} <tl var>
```

---

This retrieves the value of `<requirement>` and stores it in the `<token list variable>`. If the `<requirement>` is not found the special value `\q_no_value` is used. The `<token list variable>` is assigned locally.

The following describe the requirements which can be currently tested. Requirements with a value should use `\pdfmeta_standard_verify:nn` or `\pdfmeta_standard_verify:nnN` to test a local value against the standard. The rule numbers refer to <https://docs.verapdf.org/validation/pdfa-part1/>

### 1.1.1 Simple tests without handler

`outputintent_A` requires to embed a color profile and reference it in a `/Outputintent` and that all output intents reference the same colorprofile. The value stores the subtype. *This requirement is detected and fulfilled by `l3pdfmeta` if the provided interface in `\DocumentMetadata` is used, see below.*

`annot_flags` in annotations the `Print` flag should be true, `Hidden`, `Invisible`, `NoView` should be false. *This requirement is detected and set by `l3pdfmeta` for annotations created with the `l3pdfannot`. A new check is only needed if the flags are changed or if links are created by other means.*

`no_encryption` don't encrypt

**no\_external\_content** no /F, /FFilter, or /FDecodeParms in stream dictionaries

**no\_embed\_content** no /EF key in filespec, no /Type/EmbeddedFiles. *This will be checked in future by l3pdffiles for the files it embeds.* The restriction is set only for PDF/A-1 versions. PDF/A-2 and PDF/A-3 lifted this restriction: PDF/A-2 allows to embed other PDF documents conforming to either PDF/A-1 or PDF/A-2, and PDF/A-3 and PDF/A-4F allows any embedded files.

**only\_pdfa\_embed\_content** This is set for PDF/A-2a, PDF/A-2b, PDF/A-2u and PDF/A-4. I don't see a way to test the PDF/A-2 requirement so currently it will simply allow everything. Perhaps a test for at least the PDF-format will be added in future.

**Catalog\_no\_OCProperties** don't add /OCProperties to the catalog *l3pdfmeta removes this entry at the end of the document*

**Catalog\_OCProperties\_no\_AS** do not use /AS optional content configuration dictionary.

**Catalog\_EmbeddedFiles** ensure that an EmbeddedFiles name tree is in the catalog. This is required for PDF/A-4f.

**annot\_widget\_no\_AA** (rule 6.6.2-1) no AA dictionary in widget annotation, this will e.g. be checked by the new hyperref driver.

**annot\_widget\_no\_A\_AA** (rule 6.9-2) no A and AA dictionary in widget.

**form\_no\_AA** (6.9-3) no /AA dictionary in form field

**unicode** that is set in the U-standards, A-2u and A-3u and means that every text should be in unicode. This is not something that can be enforced or tested from TeX, but in a current LaTeX normally ToUnicode are set for all fonts.

**tagged** that is set in A-2a and A-3a and means that the pdf must be tagged. This is currently neither tested not enforced somewhere.

**no\_CharSet** CharSet is deprecated in pdf 2.0 and should not be used in A-4. l3pdfmeta will therefore suppress it for the engines pdftex and luatex (the other engines have no suitable option)

**omit\_CID** This avoids with PDF/A-2 and newer a failure because of with missing CID identifications (e.g. from rule ISO 19005-2:2011, Clause: 6.2.11.4.2) It has only with luatex an effect.

**Trailer\_no\_Info** The Info dictionary has been deprecated since quite some time. Metadata should be set with XMP-data instead. In PDF A-4 now the Info dictionary shall not be present in the trailer dictionary at all (unless there exists a PieceInfo entry in the Catalog). And if it is present it should only contain the /ModDate entry. In texlive 2023 the engines pdftex and luatex have primitives to suppress the dictionary and l3pdfmeta will make use of it.

### 1.1.2 Tests with values and special handlers

`min_pdf_version` stores the minimal PDF version needed for a standard. It should be checked against the current PDF version (`\pdf_version:`). A failure means that the version should be changed. Currently there is only one hard requirement which leads to a failure in a validator like verapdf: The A-4 standard should use PDF 2.0. As PDF A-1 is based on PDF 1.4 and PDF A-2 and A-3 are based on PDF 1.7 `l3pdfmeta` also sets these versions also as requirements. These requirements are checked by `l3pdfmeta` when the version is set with `\DocumentMetadata` and a warning is issued (but the version is not changed). More checks are only needed if the version is changed later.

`max_pdf_version` stores the maximal PDF version. It should be checked against the current PDF version (`\pdf_version:`). A failure means that the version should be changed. The check is currently relevant only for the A-1 to A-3 standards: PDF 2.0 leads to a failure in a validator like verapdf so the maximal version should be PDF 1.7. This requirement is checked by `l3pdfmeta` when the version is set with `\DocumentMetadata` and a warning is issued (but the version is not changed). More checks are only needed if the version is changed later.

`named_actions` this requirement restricts the list of allowed named actions to `NextPage`, `PrevPage`, `FirstPage`, `LastPage`. The check should supply the named action without slash (e.g. `View` (failure) or `NextPage` (pass)).

`annot_action_A` (rule 6.6.1-1) this requirement restricts the allowed subtypes of the `/A` dictionary of an action. The check should supply the user subtype without slash e.g. as `GoTo` (pass) or `Movie` (failure).

## 1.2 Colorprofiles and OutputIntent

The pdf/A standards require that a color profile is embedded and referenced in the catalog in the `/OutputIntent` array.

The problem is that the pdf/A standards also require, that if the PDF has more than one entry in the `/OutputIntent` array (which is allowed), their `/DestOutputProfile` should all reference the same color profile<sup>2</sup>.

Enforcing this fully is impossible if entries are added manually by users or packages with `\pdfmanagement_add:nnn {Catalog}{OutputIntents}{object reference}` as it is difficult to inspect and remove entries from the `/OutputIntent` array.

So we provide a dedicated interface to avoid the need of manual user settings and allow the code to handle the requirements of the standard. The interface doesn't handle yet all finer points for PDF/X standards, e.g. named profiles, it is meant as a starting point to get at least PDF/A validation here.

The interface looks like this

```
\DocumentMetadata
{
  %other options for example pdfstandard
  colorprofiles=
  {
    A = sRGB.icc, %or a or longer GTS_PDFa1 = sRGB.icc
```

---

<sup>2</sup>see rule 6.2.2-2 at <https://docs.verapdf.org/validation/pdfa-part1/>

```

    X = FOGRA39L_coated.icc, % or x or longer GTS_PDFX
    ISO_PDFE1 = whatever.icc
  }
}

```

sRGB.icc and FOGRA39L\_coated.icc (from the colorprofiles package) are predefined and will work directly<sup>3</sup>. whatever.icc will need special setup in the document preamble to declare the values for the `OutputIntent` dictionary, but the interface hasn't been added yet. This will be decided later.

If an A-standard is detected or set which requires that all `/DestOutputProfile` reference the same color profile, the setting is changed to the equivalent of

```

\DocumentMetadata
{
  %other options
  pdfstandard=A-2b,
  colorprofiles=
  {
    A = sRGB.icc, %or longer GTS_PDFA1 = sRGB.icc
    X = sRGB.icc,
    ISO_PDFE1 = sRGB.icc
  }
}

```

The pdf/A standards will use `A=sRGB.icc` by default, so this doesn't need to be declared explicitly.

### 1.3 Regression tests

When doing regression tests one has to set various metadata to fix values.

---

```
\pdfmeta_set_regression_data: \pdfmeta_set_regression_data:
```

This sets various metadata to values needed by the L<sup>A</sup>T<sub>E</sub>X regression tests. It also sets the seed for random functions. If a current l3backend is used and `\c_sys_timestamp_str` is available, the command does not set dates, but assumes that the environment variable `SOURCE_DATE_EPOCH` is used.

## 2 XMP-metadata

XMP-metadata are data in XML format embedded in a stream inside the PDF and referenced from the `/Catalog`. Such a XMP-metadata stream contains various document related data, is required by various PDF standards and can replace or extend the data in the `/Info` dictionary. In PDF 2.0 the `/Info` dictionary is actually deprecated and only XMP-metadata should be used for the metadata of the PDF.

---

<sup>3</sup>The `dvips` route will require that `ps2pdf` is called with `-dNOSAFER`, and that the color profiles are in the current folder as `ps2pdf` doesn't use `kpathsea` to find them.

The content of a XMP-metadata stream is not a fix set of data. Typically fields like the title, the author, the language and keywords will be there. But standards like e.g. ZUGferd (a standard for electronic bills) can require to add more fields, and it is also possible to define and add purely local data.

In some workflows (e.g. if dvips + ghostscript is used) a XMP-metadata stream with some standard content is added automatically by the backend, but normally it must be created with code.

For this task the packages `hyperxmp`, `xmpincl` or `pdfx` (which uses `xmpincl`) can be used, but all these packages are not compatible with the `pdfmanagement`<sup>4</sup>. The following code is meant as replacement for these packages.

`hyperxmp` uses `\hypersetup` as user interface to enter the XMP-metadata. This syntax is also supported by the new code<sup>5</sup>, so if `hyperref` has been loaded, e.g. `pdftitle=xxx` can be used to set the title. But XMP-metadata shouldn't require to use `hyperref` and in a future version an interface without `hyperref` will be added.

There is currently no full user interface command to extend the XMP-metadata with for example the code needed for ZUGferd, they will be added in a second step.

## 2.1 Debug option

The resulting XMP-packet can be written to an external file by activating a debug option

```
\DocumentMetadata{debug={xmp-export}}
%or
\DocumentMetadata{debug={xmp-export=true}}
%or
\DocumentMetadata{debug={xmp-export=filename}}
```

By default the data are written to `\jobname.xmpi`, if a `filename` is given, then `filename.xmpi` is used instead. `xmp-export=false` deactivates the export.

## 2.2 Encoding and escaping

XMP-metadata are stored as UTF-8 in the PDF. This mean if you open a PDF in an editor a content like “grüße” will be shown probably as “grÄ¼Äÿe”. As XMP-metadata are in XML format special chars like `<`, `>`, and `&` and „ must be escaped.

`hyperxmp` hooks into `hyperref` and passes all input through `\pdfstringdef`. This means a word like “hallo” is first converted by `\pdfstringdef` into `\376\377\000h\000a\0001\0001\000o` and then back to UTF-8 by `hyperxmp` and in the course of this action the XML-escapings are applied. `pdfx` uses `\pdfstringdef` together with a special fontencoding (similar to the PU-encoding of `hyperref`) for a similar aim. The code here is based on `\text_purify:n` followed by a few replacements for the escaping.

User data should normally be declared in the preamble (or even in the `\DocumentMetadata` command), and consist of rather simple text; `&` can be entered as `\&` (but directly `&` will normally work too), babel shorthands should not be used. Some data are interpreted as comma lists, in this cases commas which are part of the text should be protected by braces. In some cases a text in brackets like `[en]` is interpreted as language tag, if they are part of a text they should be protected by braces too. XMP-metadata are

<sup>4</sup>`hyperxmp` was partly compatible as the `pdfmanagement` contained some patches for it, but these patches have now been removed.

<sup>5</sup>with a number of changes which are discussed in more details below

stored uncompressed in the PDF so if you are unsure if a value has been passed correctly, open the PDF in an editor, copy the whole block and pass it to a validator, e.g. <https://www.w3.org/RDF/Validator/>.

## 2.3 User interfaces and differences to hyperxmp

### 2.3.1 PDF standards

The hyperxmp/hyperref keys `pdfapart`, `pdfaconformance`, `pdfuapart`, `pdfxstandard` and `pdfa` are ignored by this code. Standards must be set with the `pdfstandard` key of `\DocumentMetadata`. This key can be used more than once, e.g.

```
pdfstandard=A-2b,pdfstandard=X-4,pdfstandard=UA-1.
```

Note that using these keys doesn't mean that the document actually follows the standard. L<sup>A</sup>T<sub>E</sub>X can neither ensure nor check all requirements of a standard, and not everything it can do theoretically has already been implemented. When setting an **A** standard, the code will e.g. insert a color profile and warn if the PDF version doesn't fit, but **X** and **UA** currently only adds the relevant declarations to the XMP-metadata. It is up to the author to ensure and validate that the document actually follows the standard.

### 2.3.2 Declarations

PDF knows beside standards also a more generic method to declare conformance to some specification by adding a declaration, see <https://pdfa.org/wp-content/uploads/2019/09/PDF-Declarations.pdf>). Such declarations can be added as a simple url which identify the specification or with additional details regarding date and credentials. An example would be

```
\DocumentMetadata{}
\documentclass{article}
\ExplSyntaxOn
\pdfmeta_xmp_add_declaration:e {https://pdfa.org/declarations\c_hash_str iso32005}
\pdfmeta_xmp_add_declaration:ennnn
  {https://pdfa.org/declarations\c_hash_str wcag21A}{2023-11-20}{}
\pdfmeta_xmp_add_declaration:nnnnn
  {https://github.com/TikZlings/no-duck-harmed}
  {Ulrike~Fischer}{2023-11-20}{Bär}{https://github.com/u-fischer/bearwear}
\pdfmeta_xmp_add_declaration:nnnnn
  {https://github.com/TikZlings/no-duck-harmed}
  {Ulrike~Fischer}{2023-11-20}{Paulo}{https://github.com/cereda/sillypage}
\ExplSyntaxOff
\begin{document}
  text
\end{document}
```

### 2.3.3 Dates

- The dates `xmp:CreateDate`, `xmp:ModifyDate`, `xmp:MetadataDate` are normally set automatically to the current date/time when the compilation started. If they should be changed (e.g. for regression tests to produce reproducible documents) they

can be set with `\hypersetup` with the keys `pdfcreationdate`, `pdfmoddate` and `pdfmetadate`.

```
\hypersetup{pdfcreationdate=D:20010101205959-00'00'}
```

The format should be a full date/time in PDF format, so one of these (naturally the numbers can change):

```
D:20010101205959-00'00'  
D:20010101205959+00'00'  
D:20010101205959Z
```

- The date `dc:date` is an “author date” and so should normally be set to the same date as given by `\date`. This can be done with the key `pdfdate`<sup>6</sup>. The value should be a date in ISO 8601 format:

```
2022                %year  
2022-09-04          %year-month-day  
2022-09-04T19:20    %year-month-day hour:minutes  
2022-09-04T19:20:30 % year-month-day hour:minutes:second  
2022-09-04T19:20:30.45 % year-month-day hour:minutes:second with fraction  
2022-09-04T19:20+01:00 % with time zone designator  
2022-09-04T19:20-02:00 % time zone designator  
2022-09-04T19:20Z      % time zone designator
```

It is also possible to give the date as a full date in PDF format as described above. If not set the current date/time is used.

## 2.4 Language

The code assumes that a default language is always declared (as the `pdfmanagement` gives the `/Lang` entry in the catalog a default value) This language can be changed with the `\DocumentMetadata` key `lang` (preferred) but the `hyperref` key `pdflang` is also honored. Its value should be a simple language tag like `de` or `de-DE`.

The main language is also used in a number of attributes in the XMP data, if wanted a different language can be set here with the `hyperref/hyperxmp` key `pdfmetalang`.

A number of entries can be given a language tag. Such a language is given by using an “optional argument” before the text:

```
\hypersetup{pdftitle={ [en]english, [de]deutsch}}  
\hypersetup{pdfsubtitle={ [en]subtitle in english}}
```

## 2.5 Rights

The keys `pdfcopyright` and `pdflicenseurl` work similar as in `hyperxmp`. But differently to `hyperxmp` the code doesn't set the `xmpRights:Marked` property, as I have some doubts that one deduce its value simply by checking if the other keys have been used; if needed it can be added by using one of these settings (true means with copyright, false means public domain).

```
\AddToDocumentProperties[document]{copyright}{true}  
\AddToDocumentProperties[document]{copyright}{false}
```

<sup>6</sup>Extracting the value automatically from `\date` is not really possible as authors often put formatting or additional info in this command.



## 2.6 PDF related data

The PDF producer is for all engines by default built from the engine name and the engine version and doesn't use the banners as with `hyperxmp` and `pdfx`, it can be set manually with the `pdfproducer` key.

The key `pdftrapped` is ignored. `Trapped` is deprecated in PDF 2.0.

## 2.7 Document data

The authors should be given with the `pdfauthor` key, separated by commas. If an author contains a comma, protect/hide it by a brace.

## 2.8 User commands

The XMP-meta data are added automatically. This can be suppressed with the `\DocumentMetadata` key `xmp`.

---

`\pdfmeta_xmp_add:n` `\pdfmeta_xmp_add:n{<XML>}`

With this command additional XML code can be added to the Metadata. The content is added unchanged, and not sanitized.

---

`\pdfmeta_xmp_xmlns_new:nn` `\pdfmeta_xmp_xmlns_new:nn{<prefix>}{<uri>}`

With this command a xmlns name space can be added. The `<uri>` argument is expanded, a hash can be input with `\c_hash_str`.

With the two following commands PDF declarations can be added to the XMP metadata (see <https://pdfa.org/wp-content/uploads/2019/09/PDF-Declarations.pdf>).

---

`\pdfmeta_xmp_add_declaration:n` `\pdfmeta_xmp_add_declaration:n{<uri>}`  
`\pdfmeta_xmp_add_declaration:e`

This add a PDF declaration with the required `conformsTo` property to the XMP metadata. `<uri>` should not be empty and is a URI specifying the standard or profile referred to by the PDF Declaration. If the uri contains a hash, use `\c_hash_str` to escape it and use the `e` variant to expand it.

---

`\pdfmeta_xmp_add_declaration:nnnnn` `\pdfmeta_xmp_add_`  
`\pdfmeta_xmp_add_declaration:(ennnn|eeenn)` `declaration:nnnnn{<uri>}{<By>}{<Date>}{<Credentials>}{<Report>}`

This add a PDF declaration to the XMP metadata similar to `\pdfmeta_xmp_add_declaration:n`. With `<By>`, `<Date>`, `<Credentials>`, `<Report>` the optional fields `claimBy` (text), `claimDate` (iso date), `claimCredentials` (text) and `claimReport` (uri) of the `claimData` property can be given. If `\pdfmeta_xmp_add_declaration:nnnnn` is used twice with the same `<uri>` argument the `claimData` are concatenated. There is no check if the `claimData` are identical.

The following two commands can be used to extend the schema declarations in the XMP metadata. This is for example needed to implement a standard like ZUGferd/Factor X for invoices. A schema declaration should be added only once but as this task is probably not needed frequently only light guards are there to avoid duplicated entries.

---

```
\pdfmeta_xmp_schema_new:nnn \pdfmeta_xmp_schema_new:nnn{<text>}{<prefix>}{<uri>}
```

`<text>` is some string describing the schema, e.g. `PDF/A~Identification~Schema`, `<prefix>` is the unique prefix used by the schema. This prefix must be declared first with `\pdfmeta_xmp_xmlns_new:nn`. If a schema with this prefix has already been declared, it will currently be ignored with a warning. The `<uri>` is expanded, so a hash can for example be given as `\c_hash_str`.

---

```
\pdfmeta_xmp_property_new:nnnnn \pdfmeta_xmp_property_new:nnnnn{<schema  
prefix>}{<name>}{<type>}{<category>}{<description>}
```

If the new property already exists in the schema (as identified by the combination of `<schema prefix>` and `<name>`) the property is silently ignore. `<schema prefix>` is the prefix declared with the previous command. `schema`, e.g. `PDF/A~Identification~Schema`, `<name>` is a short string that identifies the property, e.g. `xmpMM` or `year`. It must be unique in the properties of a schema. `<type>` is e.g. `URI` or `Integer` or `Text`, `<category>` is e.g. `internal` or `external`, `<description>` is a free description string.

### 3 l3pdfmeta implementation

```
1 <@@=pdfmeta>  
2 <*header>  
3 \ProvidesExplPackage{l3pdfmeta}{2025-05-15}{0.96r}  
4 {PDF-Standards---LaTeX PDF management testphase bundle}  
5 </header>
```

Message for unknown standards

```
6 <*package>  
7 \msg_new:nnn {pdf }{unknown-standard}{The~standard~'#1'~is~unknown~and~has~been~ignored}
```

Message for not fitting pdf version

```
8 \msg_new:nnn {pdf }{wrong-pdfversion}  
9 {PDF~version~#1~is~too~#2~for~standard~'#3'.}
```

Messages for embedded files

```
10 \msg_new:nnn {pdf }{validation-failure}  
11 {  
12 PDF~standard~validation~failure.\  
13 #1  
14 }
```

```
\l__pdfmeta_tmpa_tl  
\l__pdfmeta_tmpb_tl  
\l__pdfmeta_tmpa_str  
\g__pdfmetatmpa_str  
\l__pdfmeta_tmpa_seq  
\l__pdfmeta_tmpb_seq  
15 \tl_new:N \l__pdfmeta_tmpa_tl  
16 \tl_new:N \l__pdfmeta_tmpb_tl  
17 \str_new:N \l__pdfmeta_tmpa_str  
18 \str_new:N \g__pdfmeta_tmpa_str  
19 \seq_new:N \l__pdfmeta_tmpa_seq  
20 \seq_new:N \l__pdfmeta_tmpb_seq
```

(End of definition for `\l__pdfmeta_tmpa_tl` and others.)

## 3.1 Standards (work in progress)

### 3.1.1 Tools and tests

This internal property will contain for now the settings for the document.

`\g__pdfmeta_standard_prop`

```
21 \prop_new:N \g__pdfmeta_standard_prop
```

*(End of definition for \g\_\_pdfmeta\_standard\_prop.)*

### 3.1.2 Functions to check a requirement

At first two commands to get the standard value if needed:

`\pdfmeta_standard_item:n`

```
22 \cs_new:Npn \pdfmeta_standard_item:n #1
23 {
24   \prop_item:Nn \g__pdfmeta_standard_prop {#1}
25 }
```

*(End of definition for \pdfmeta\_standard\_item:n. This function is documented on page 2.)*

`\pdfmeta_standard_get:nN`

```
26 \cs_new_protected:Npn \pdfmeta_standard_get:nN #1 #2
27 {
28   \prop_get:NnN \g__pdfmeta_standard_prop {#1} #2
29 }
```

*(End of definition for \pdfmeta\_standard\_get:nN. This function is documented on page 2.)*

Now two functions to check the requirement. A simple and one value/handler based.

`\pdfmeta_standard_verify_p:n`

This is a simple test is the requirement is in the prop.

`\pdfmeta_standard_verify:nTF`

```
30 \prg_new_conditional:Npnn \pdfmeta_standard_verify:n #1 {T,F,TF}
31 {
32   \prop_if_in:NnTF \g__pdfmeta_standard_prop {#1}
33   {
34     \prg_return_false:
35   }
36   {
37     \prg_return_true:
38   }
39 }
```

*(End of definition for \pdfmeta\_standard\_verify:nTF. This function is documented on page 2.)*

`\pdfmeta_standard_verify:nnTF`

This allows to test against a user value. It calls a test handler if this exists and passes the user and the standard value to it. The test handler should return true or false.

```
40 \prg_new_protected_conditional:Npnn \pdfmeta_standard_verify:nn #1 #2 {T,F,TF}
41 {
42   \prop_if_in:NnTF \g__pdfmeta_standard_prop {#1}
43   {
44     \cs_if_exist:cTF {__pdfmeta_standard_verify_handler_#1:nn}
45     {
46       \exp_args:Nnne
47       \use:c
```

```

48         {__pdfmeta_standard_verify_handler_#1:nn}
49         { #2 }
50         { \prop_item:Nn \g__pdfmeta_standard_prop {#1} }
51     }
52     {
53     \prg_return_false:
54     }
55 }
56 {
57     \prg_return_true:
58 }
59 }

```

(End of definition for \pdfmeta\_standard\_verify:nnTF. This function is documented on page 2.)

Now we setup a number of handlers.

The first actually ignores the user values and tests against the current pdf version. If this is smaller than the minimum we report a failure. #1 is the user value, #2 the reference value from the standard.

\_standard\_verify\_handler\_min\_pdf\_version:nn

```

60 %
61 \cs_new_protected:Npn \__pdfmeta_standard_verify_handler_min_pdf_version:nn #1 #2
62 {
63     \pdf_version_compare:NnTF <
64     { #2 }
65     {\prg_return_false:}
66     {\prg_return_true:}
67 }

```

(End of definition for \\_\_pdfmeta\_standard\_verify\_handler\_min\_pdf\_version:nn.)

The next is the counter part and checks that the version is not to high

\_standard\_verify\_handler\_max\_pdf\_version:nn

```

68 %
69 \cs_new_protected:Npn \__pdfmeta_standard_verify_handler_max_pdf_version:nn #1 #2
70 {
71     \pdf_version_compare:NnTF >
72     { #2 }
73     {\prg_return_false:}
74     {\prg_return_true:}
75 }

```

(End of definition for \\_\_pdfmeta\_standard\_verify\_handler\_max\_pdf\_version:nn.)

The next checks if the user value is in the list and returns a failure if not.

ta\_standard\_verify\_handler\_named\_actions:nn

```

76
77 \cs_new_protected:Npn \__pdfmeta_standard_verify_handler_named_actions:nn #1 #2
78 {
79     \tl_if_in:nnTF { #2 }{ #1 }
80     {\prg_return_true:}
81     {\prg_return_false:}
82 }

```

(End of definition for \\_\_pdfmeta\_standard\_verify\_handler\_named\_actions:nn.)

The next checks if the user value is in the list and returns a failure if not.

a\_standard\_verify\_handler\_annot\_action\_A:nn

```
83 \cs_new_protected:Npn \__pdfmeta_standard_verify_handler_annot_action_A:nn #1 #2
84 {
85   \tl_if_in:nnTF { #2 }{ #1 }
86   {\prg_return_true:}
87   {\prg_return_false:}
88 }
```

(End of definition for \\_\_pdfmeta\_standard\_verify\_handler\_annot\_action\_A:nn.)

This check is probably not needed, but for completeness

ard\_verify\_handler\_outputintent\_subtype:nn

```
89 \cs_new_protected:Npn \__pdfmeta_standard_verify_handler_outputintent_subtype:nn #1 #2
90 {
91   \tl_if_eq:nnTF { #2 }{ #1 }
92   {\prg_return_true:}
93   {\prg_return_false:}
94 }
```

(End of definition for \\_\_pdfmeta\_standard\_verify\_handler\_outputintent\_subtype:nn.)

### 3.1.3 Enforcing requirements

A number of requirements can sensibly be enforced by us.

**Annot flags** pdf/A require a number of settings here, we store them in a command which can be added to the property of the standard:

```
95 \cs_new_protected:Npn \__pdfmeta_verify_pdfa_annot_flags:
96 {
97   \bitset_set_true:Nn \l_pdfannot_F_bitset {Print}
98   \bitset_set_false:Nn \l_pdfannot_F_bitset {Hidden}
99   \bitset_set_false:Nn \l_pdfannot_F_bitset {Invisible}
100  \bitset_set_false:Nn \l_pdfannot_F_bitset {NoView}
101  \pdfannot_dict_put:nnn {link/URI}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
102  \pdfannot_dict_put:nnn {link/GoTo}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
103  \pdfannot_dict_put:nnn {link/GoToR}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
104  \pdfannot_dict_put:nnn {link/Launch}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
105  \pdfannot_dict_put:nnn {link/Named}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
106 }
```

At begin document this should be checked:

```
107 \hook_gput_code:nnn {begindocument} {pdf}
108 {
109   \pdfmeta_standard_verify:nF { annot_flags }
110   { \__pdfmeta_verify_pdfa_annot_flags: }
111   \pdfmeta_standard_verify:nF { Trailer_no_Info }
112   { \__pdf_backend_omit_info:n {1} }
113   \pdfmeta_standard_verify:nF { no_CharSet }
114   { \__pdf_backend_omit_charset:n {1} }
115   \pdfmeta_standard_verify:nF { omit_CID }
116   { \__pdf_backend_omit_cidset:n {1} }
117   \pdfmeta_standard_verify:nnF { min_pdf_version }
118   { \pdf_version: }
119   { \msg_warning:nneee {pdf}{wrong-pdfversion}
```

```

120     {\pdf_version:}{low}
121     {
122     \pdfmeta_standard_item:n{type}
123     -
124     \pdfmeta_standard_item:n{level}
125     }
126   }
127 \pdfmeta_standard_verify:nnF { max_pdf_version }
128 { \pdf_version: }
129 { \msg_warning:nneee {pdf}{wrong-pdfversion}
130   {\pdf_version:}{high}
131   {
132     \pdfmeta_standard_item:n{type}
133     -
134     \pdfmeta_standard_item:n{level}
135   }
136 }
137 }

```

### 3.1.4 pdf/A

We use global properties so that follow up standards can be copied and then adjusted. Some note about requirements for more standard can be found in info/pdfstandard.tex.

```

\g__pdfmeta_standard_pdf/A-1B_prop
\g__pdfmeta_standard_pdf/A-2A_prop
\g__pdfmeta_standard_pdf/A-2B_prop
\g__pdfmeta_standard_pdf/A-2U_prop
\g__pdfmeta_standard_pdf/A-3A_prop
\g__pdfmeta_standard_pdf/A-3B_prop
\g__pdfmeta_standard_pdf/A-3U_prop
\g__pdfmeta_standard_pdf/A-4_prop
\g__pdfmeta_standard_pdf/A-4F_prop
138 \prop_new:c { g__pdfmeta_standard_pdf/A-1B_prop }
139 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/A-1B_prop }
140 {
141   ,name           = pdf/A-1B
142   ,type           = A
143   ,level          = 1
144   ,conformance   = B
145   ,year           = 2005
146   ,min_pdf_version = 1.4      %minimum
147   ,max_pdf_version = 1.4      %minimum
148   ,no_encryption  =
149   ,no_external_content = % no F, FFilter, or FDecodeParms in stream dicts
150   ,no_embed_content = % no EF key in filespec, no /Type/EmbeddedFiles
151   ,max_string_size = 65535
152   ,max_array_size = 8191
153   ,max_dict_size  = 4095
154   ,max_obj_num    = 8388607
155   ,max_nest_qQ    = 28
156   ,named_actions  = {NextPage, PrevPage, FirstPage, LastPage}
157   ,annot_flags    =
158   %booleans. Only the existence of the key matter.
159   %If the entry is added it means a requirements is there
160   %(in most cases "don't use ...")
161   %
162   %=====
163   % Rule 6.1.13-1 CosDocument, isOptionalContentPresent == false
164   ,Catalog_no_OCProperties =
165   % Rule 6.9-4 The AS key shall not appear in any optional content configuration dictionary
166   % actually only starting with A-2 but doesn't harm here either

```

```

167 ,Catalog_OCProperties_no_AS=
168 %=====
169 % Rule 6.6.1-1: PDAAction, S == "GoTo" || S == "GoToR" || S == "Thread"
170 %           || S == "URI" || S == "Named" || S == "SubmitForm"
171 % means: no /S/Launch, /S/Sound, /S/Movie, /S/ResetForm, /S/ImportData,
172 %           /S/JavaScript, /S/Hide
173     ,annot_action_A           = {GoTo,GoToR,Thread,URI,Named,SubmitForm}
174 %=====
175 % Rule 6.6.2-1: PDAnnot, Subtype != "Widget" || AA_size == 0
176 % means: no AA dictionary
177     ,annot_widget_no_AA       =
178 %=====
179 % Rule 6.9-2: PDAnnot, Subtype != "Widget" || (A_size == 0 && AA_size == 0)
180 % (looks like a tightening of the previous rule)
181     ,annot_widget_no_A_AA     =
182 %=====
183 % Rule 6.9-1 PDAcroForm, NeedAppearances == null || NeedAppearances == false
184     ,form_no_NeedAppearances =
185 %=====
186 %Rule 6.9-3 PDFFormField, AA_size == 0
187     ,form_no_AA               =
188 %=====
189 % to be continued https://docs.verapdf.org/validation/pdfa-part1/
190 % - Outputintent/colorprofiles requirements
191 % an outputintent should be loaded and is unique.
192     ,outputintent_A           = {GTS_PDFA1}
193 % - no Alternates key in image dictionaries
194 % - no OPI, Ref, Subtype2 with PS key in xobjects
195 % - Interpolate = false in images
196 % - no TR, TR2 in ExtGstate
197 }
198
199 %A-2b =====
200 \prop_new:c { g__pdfmeta_standard_pdf/A-2B_prop }
201 \prop_gset_eq:cc
202   { g__pdfmeta_standard_pdf/A-2B_prop }
203   { g__pdfmeta_standard_pdf/A-1B_prop }
204 \prop_gput:cnn
205   { g__pdfmeta_standard_pdf/A-2B_prop }{name}{pdf/A-2B}
206 \prop_gput:cnn
207   { g__pdfmeta_standard_pdf/A-2B_prop }{year}{2011}
208 \prop_gput:cnn
209   { g__pdfmeta_standard_pdf/A-2B_prop }{level}{2}
210 % embedding files is allowed (with restrictions)
211 \prop_gremove:cn
212   { g__pdfmeta_standard_pdf/A-2B_prop }
213   { no_embed_content }
214 \prop_gput:cnn
215   { g__pdfmeta_standard_pdf/A-2B_prop }
216   { only_pdfa_embed_content }
217   {}
218 \prop_gput:cnn
219   { g__pdfmeta_standard_pdf/A-2B_prop }{max_pdf_version}{1.7}
220 \prop_gput:cnn

```

```

221 { g__pdfmeta_standard_pdf/A-2B_prop }{omit_CID}{}
222 % OCG layers are allowed (with restrictions)
223 \prop_gremove:cn
224 { g__pdfmeta_standard_pdf/A-2B_prop }
225 { Catalog_no_OCProperties }
226
227 %A-2u =====
228 \prop_new:c { g__pdfmeta_standard_pdf/A-2U_prop }
229 \prop_gset_eq:cc
230 { g__pdfmeta_standard_pdf/A-2U_prop }
231 { g__pdfmeta_standard_pdf/A-2B_prop }
232 \prop_gput:cnn
233 { g__pdfmeta_standard_pdf/A-2U_prop }{name}{pdf/A-2U}
234 \prop_gput:cnn
235 { g__pdfmeta_standard_pdf/A-2U_prop }{conformance}{U}
236 \prop_gput:cnn
237 { g__pdfmeta_standard_pdf/A-2U_prop }{unicode}{}
238
239 %A-2a =====
240 \prop_new:c { g__pdfmeta_standard_pdf/A-2A_prop }
241 \prop_gset_eq:cc
242 { g__pdfmeta_standard_pdf/A-2A_prop }
243 { g__pdfmeta_standard_pdf/A-2B_prop }
244 \prop_gput:cnn
245 { g__pdfmeta_standard_pdf/A-2A_prop }{name}{pdf/A-2A}
246 \prop_gput:cnn
247 { g__pdfmeta_standard_pdf/A-2A_prop }{conformance}{A}
248 \prop_gput:cnn
249 { g__pdfmeta_standard_pdf/A-2A_prop }{tagged}{}
250
251
252 %A-3b =====
253 \prop_new:c { g__pdfmeta_standard_pdf/A-3B_prop }
254 \prop_gset_eq:cc
255 { g__pdfmeta_standard_pdf/A-3B_prop }
256 { g__pdfmeta_standard_pdf/A-2B_prop }
257 \prop_gput:cnn
258 { g__pdfmeta_standard_pdf/A-3B_prop }{name}{pdf/A-3B}
259 \prop_gput:cnn
260 { g__pdfmeta_standard_pdf/A-3B_prop }{year}{2012}
261 \prop_gput:cnn
262 { g__pdfmeta_standard_pdf/A-3B_prop }{level}{3}
263 % embedding files is allowed
264 \prop_gremove:cn
265 { g__pdfmeta_standard_pdf/A-3B_prop }
266 { only_pdfa_embed_content }
267 %A-3u =====
268 \prop_new:c { g__pdfmeta_standard_pdf/A-3U_prop }
269 \prop_gset_eq:cc
270 { g__pdfmeta_standard_pdf/A-3U_prop }
271 { g__pdfmeta_standard_pdf/A-3B_prop }
272 \prop_gput:cnn
273 { g__pdfmeta_standard_pdf/A-3U_prop }{name}{pdf/A-3U}
274 \prop_gput:cnn

```



```

275 { g__pdfmeta_standard_pdf/A-3U_prop }{conformance}{U}
276 \prop_gput:cnn
277 { g__pdfmeta_standard_pdf/A-3U_prop }{unicode}{ }
278
279 %A-3a =====
280 \prop_new:c { g__pdfmeta_standard_pdf/A-3A_prop }
281 \prop_gset_eq:cc
282 { g__pdfmeta_standard_pdf/A-3A_prop }
283 { g__pdfmeta_standard_pdf/A-3B_prop }
284 \prop_gput:cnn
285 { g__pdfmeta_standard_pdf/A-3A_prop }{name}{pdf/A-3A}
286 \prop_gput:cnn
287 { g__pdfmeta_standard_pdf/A-3A_prop }{conformance}{A}
288 \prop_gput:cnn
289 { g__pdfmeta_standard_pdf/A-3A_prop }{tagged}{ }
290
291 %A-4 =====
292 \prop_new:c { g__pdfmeta_standard_pdf/A-4_prop }
293 \prop_gset_eq:cc
294 { g__pdfmeta_standard_pdf/A-4_prop }
295 { g__pdfmeta_standard_pdf/A-3U_prop }
296 \prop_gput:cnn
297 { g__pdfmeta_standard_pdf/A-4_prop }{name}{pdf/A-4}
298 \prop_gput:cnn
299 { g__pdfmeta_standard_pdf/A-4_prop }{level}{4}
300 \prop_gput:cnn
301 { g__pdfmeta_standard_pdf/A-4_prop }{min_pdf_version}{2.0}
302 \prop_gput:cnn
303 { g__pdfmeta_standard_pdf/A-4_prop }{year}{2020}
304 \prop_gput:cnn
305 { g__pdfmeta_standard_pdf/A-4_prop }{no_CharSet}{ }
306 \prop_gput:cnn
307 { g__pdfmeta_standard_pdf/A-4_prop }{Trailer_no_Info}{ }
308 \prop_gput:cnn
309 { g__pdfmeta_standard_pdf/A-4_prop }{only_pdfa_embed_content}{ }
310 \prop_gremove:cn
311 { g__pdfmeta_standard_pdf/A-4_prop }{conformance}
312 \prop_gremove:cn
313 { g__pdfmeta_standard_pdf/A-4_prop }{max_pdf_version}
314 \prop_gremove:cn
315 { g__pdfmeta_standard_pdf/A-4_prop }{Catalog_OCProperties_no_AS}
316 %A-4f =====
317 \prop_new:c { g__pdfmeta_standard_pdf/A-4F_prop }
318 \prop_gset_eq:cc
319 { g__pdfmeta_standard_pdf/A-4F_prop }
320 { g__pdfmeta_standard_pdf/A-4_prop }
321 \prop_gput:cnn
322 { g__pdfmeta_standard_pdf/A-4F_prop }{conformance}{F}
323 % containsEmbeddedFiles == true ISO 19005-4:2020, Clause: 6.9, Test number: 5
324 \prop_gput:cnn
325 { g__pdfmeta_standard_pdf/A-4F_prop }{Catalog_EmbeddedFiles}{ }
326 % can contain any file
327 \prop_gremove:cn
328 { g__pdfmeta_standard_pdf/A-4F_prop }{only_pdfa_embed_content}

```

(End of definition for `\g__pdfmeta_standard_pdf/A-1B_prop` and others.)

### 3.1.5 Embedded Files

Standard 4-AF is needed if we add AF files for tagging but it also requires an Embedded-Files name tree, so we test at the end if the name tree is empty and add a small readme if yes

```

329 \AddToHook{begindocument/end}
330 {
331   \pdfmeta_standard_verify:nF{Catalog_EmbeddedFiles}
332   {
333     \tl_gput_right:Nn\g__kernel_pdfmanagement_end_run_code_tl
334     {
335       \bool_if:NT \g__pdfmanagement_active_bool
336       {
337         \pdfdict_if_empty:nT { g__pdf_Core/Catalog/Names/EmbeddedFiles }
338         {
339           \group_begin:
340           \pdfdict_put:nne {l_pdffile/Filespec} {Desc}{(note~about~PDF/A-4F)}
341           \pdfdict_put:nnn { l_pdffile/Filespec }{AFRelationship} { /Unspecified }
342           \pdffile_embed_stream:nnN
343             {The~document~was~declared~to~be~of~type~PDF/A-4f~but~hasn't~any~attachments.~
344             LaTeX~therefore~added~this~dummy~file.}
345           {pdf-A4f.txt}
346           \l__pdfmeta_tmpa_tl
347           \exp_args:Nne \__pdf_backend_Names_gpush:nn{EmbeddedFiles}{(pdf-A4f)~\l__pdfmeta
348           \group_end:
349         }
350       }
351     }
352   }
353 }

```

Before writing the xml we check if there are embedded files we know of. For A-4 we adjust the standard to A-4F is needed.

```

354 \AddToHook{pdfmeta/xmp}
355 {
356   \pdfmeta_standard_verify:nF{no_embed_content}
357   {
358     \bool_lazy_or:nnT
359     { ! \int_if_zero_p:n { \g_pdffile_embed_pdfa_int } }
360     { ! \int_if_zero_p:n { \g_pdffile_embed_nonpdfa_int } }
361     {
362       \prop_get:NnNT\g__pdfmeta_standard_prop { name } \l__pdfmeta_tmpa_tl
363       {
364         \msg_warning:nne { pdf } { validation-failure }
365         {
366           Embedded-files-detected.\iow_newline:
367           This-is-not-allowed-in-standard~\l__pdfmeta_tmpa_tl
368         }
369       }
370     }
371   }
372   \pdfmeta_standard_verify:nF {only_pdfa_embed_content}

```

```

373 {
374   \int_if_zero:nF { \g_pdffile_embed_nonpdfa_int }
375   {
376     \prop_get:NnNT\g__pdfmeta_standard_prop { name }\l__pdfmeta_tmpa_tl
377     {
378       \str_if_eq:VnTF {\l__pdfmeta_tmpa_tl} { pdf/A-4 }
379       {
380         \prop_gset_eq:cc
381         { g__pdfmeta_standard_prop }
382         { g__pdfmeta_standard_pdf/A-4F_prop }
383         \msg_warning:nne { pdf } { validation-failure }
384         {
385           Embedded~non-PDF~files~detected.\iow_newline:
386           Switching~standard~from~PDF/A-4~to~PDF/A-4F
387         }
388       }
389       {
390         \msg_warning:nne { pdf } { validation-failure }
391         {
392           Embedded~non-PDF~files~detected.\iow_newline:
393           This~is~not~allowed~in~standard~\l__pdfmeta_tmpa_tl
394         }
395       }
396     }
397   }
398 }
399 }

```

### 3.1.6 Colorprofiles and Outputintents

The following provides a minimum of interface to add a color profile and an outputintent need for PDF/A for now. There will be need to extend it later, so we try for enough generality.

Adding a profile and an intent is technically easy:

1. Embed the profile as stream with

```
\pdf_object_unnamed_write:nn{fstream} {{/N~4}{XXX.icc}}
```

2. Write a /OutputIntent dictionary for this

```

\pdf_object_unnamed_write:ne {dict}
{
  /Type /OutputIntent
  /S /GTS_PDFA1 % or GTS_PDFX or ISO_PDFE1 or ...
  /DestOutputProfile \pdf_object_ref_last: % ref the color profile
  /OutputConditionIdentifier ...
  ... %more info
}

```

3. Reference the dictionary in the catalog:

```
\pdfmanagement_add:nne {Catalog}{OutputIntents}{\pdf_object_ref_last:}
```

But we need to do a bit more work, to get the interface right. The object for the profile should be named, to allow l3color to reuse it if needed. And we need container to store the profiles, to handle the standard requirements.

`\g_pdfmeta_outputintents_prop` This variable will hold the profiles for the subtypes. We assume that every subtype has only one color profile.

```

400 \prop_new:N \g_pdfmeta_outputintents_prop
(End of definition for \g_pdfmeta_outputintents_prop.)
Some keys to fill the property.
401 \keys_define:nn { document / metadata }
402 {
403   colorprofiles .code:n =
404   {
405     \keys_set:nn { document / metadata / colorprofiles }{#1}
406   }
407 }
408 \keys_define:nn { document / metadata / colorprofiles }
409 {
410   ,A .code:n =
411   {
412     \tl_if_blank:nF {#1}
413     {
414       \prop_gput:Nnn \g_pdfmeta_outputintents_prop
415       { GTS_PDFA1 } {#1}
416     }
417   }
418   ,a .code:n =
419   {
420     \tl_if_blank:nF {#1}
421     {
422       \prop_gput:Nnn \g_pdfmeta_outputintents_prop
423       { GTS_PDFA1 } {#1}
424     }
425   }
426   ,X .code:n =
427   {
428     \tl_if_blank:nF {#1}
429     {
430       \prop_gput:Nnn \g_pdfmeta_outputintents_prop
431       { GTS_PDFX } {#1}
432     }
433   }
434   ,x .code:n =
435   {
436     \tl_if_blank:nF {#1}
437     {
438       \prop_gput:Nnn \g_pdfmeta_outputintents_prop
439       { GTS_PDFX } {#1}
440     }
441   }
442   ,unknown .code:n =
443   {
444     \tl_if_blank:nF {#1}

```

```

445     {
446       \exp_args:NNo
447       \prop_gput:Nnn \g__pdfmeta_outputintents_prop
448       { \l_keys_key_str } {#1}
449     }
450   }
451 }

```

At first we setup our two default profiles. This is internal as the public interface is still undecided.

```

452 \pdfdict_new:n {l_pdfmeta/outputintent}
453 \pdfdict_put:nnn {l_pdfmeta/outputintent}
454 {Type}{/OutputIntent}
455 \prop_const_from_keyval:cn { c__pdfmeta_colorprofile_sRGB.icc}
456 {
457   ,OutputConditionIdentifier=IEC~sRGB
458   ,Info=IEC~61966~2.1~Default~RGB~colour~space~~~sRGB
459   ,RegistryName=http://www.iec.ch
460   ,N = 3
461 }
462 \prop_const_from_keyval:cn { c__pdfmeta_colorprofile_FOGRA39L_coated.icc}
463 {
464   ,OutputConditionIdentifier=FOGRA39L~Coated
465   ,Info={Offset~printing,~according~to~ISO~12647~2:2004/Amd~1,~OFCOM,~ %
466     paper~type~1~or~2~==~coated~art,~115~g/m2,~tone~value~increase~
467     curves~A~(CMY)~and~B~(K)}
468   ,RegistryName=http://www.fogra.org
469   ,N = 4
470 }

```

```

__pdfmeta_embed_colorprofile:n
__pdfmeta_write_outputintent:nn

```

The commands embed the profile, and write the dictionary and add it to the catalog. The first command should perhaps be moved to l3color as it needs such profiles too. We used named objects so that we can check if the profile is already there. This is not foolproof if paths are used.

```

471 \cs_new_protected:Npn \__pdfmeta_embed_colorprofile:n #1%#1 file name
472 {
473   \pdf_object_if_exist:nF { __color_icc_ #1 }
474   {
475     \pdf_object_new:n { __color_icc_ #1 }
476     \pdf_object_write:nne { __color_icc_ #1 } { fstream }
477     {
478       {/N\c_space_tl
479         \prop_item:cn{c__pdfmeta_colorprofile_#1}{N}
480       }
481       {#1}
482     }
483   }
484 }
485
486 \cs_new_protected:Npn \__pdfmeta_write_outputintent:nn #1 #2 %#1 file name, #2 subtype
487 {
488   \group_begin:
489   \pdfdict_put:nne {l_pdfmeta/outputintent}{S}{/\str_convert_pdfname:n{#2}}
490   \pdfdict_put:nne {l_pdfmeta/outputintent}

```

```

491     {DestOutputProfile}
492     {\pdf_object_ref:n{ __color_icc_ #1 }}
493 \clist_map_inline:nn { OutputConditionIdentifier, Info, RegistryName }
494 {
495     \prop_get:cnNT
496     { c__pdfmeta_colorprofile_#1}
497     { ##1 }
498     \l__pdfmeta_tmpa_tl
499     {
500         \pdf_string_from_unicode:nVN {utf8/string}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_st
501         \pdfdict_put:nne
502         {l_pdfmeta/outputintent}{##1}{\l__pdfmeta_tmpa_str}
503     }
504 }
505 \pdf_object_unnamed_write:ne {dict}{\pdfdict_use:n {l_pdfmeta/outputintent} }
506 \pdfmanagement_add:nne {Catalog}{OutputIntents}{\pdf_object_ref_last:}
507 \group_end:
508 }

```

(End of definition for `\__pdfmeta_embed_colorprofile:n` and `\__pdfmeta_write_outputintent:nn`.)

Now the verifying code. If no requirement is set we simply loop over the property

```

509
510 \AddToHook{begindocument/end}
511 {
512     \pdfmeta_standard_verify:nTF {outputintent_A}
513     {
514         \prop_map_inline:Nn \g__pdfmeta_outputintents_prop
515         {
516             \prop_if_exist:cTF {c__pdfmeta_colorprofile_#2}
517             {
518                 \__pdfmeta_embed_colorprofile:n
519                 {#2}
520                 \__pdfmeta_write_outputintent:nn
521                 {#2}
522                 {#1}
523             }
524             {
525                 \msg_warning:nnn{pdfmeta}{colorprofile-undefined}{#2}
526             }
527         }
528     }

```

If an output intent is required for pdf/A we need to ensure, that the key of default subtype has a value, as default we take sRGB.icc. Then we loop but take always the same profile.

```

529     {
530         \exp_args:NNe
531         \prop_if_in:NnF
532         \g__pdfmeta_outputintents_prop
533         { \pdfmeta_standard_item:n { outputintent_A } }
534         {
535             \exp_args:NNe
536             \prop_gput:Nnn
537             \g__pdfmeta_outputintents_prop

```

```

538         { \pdfmeta_standard_item:n { outputintent_A } }
539         { sRGB.icc }
540     }
541     \exp_args:NNe
542     \prop_get:NnN
543     \g__pdfmeta_outputintents_prop
544     { \pdfmeta_standard_item:n { outputintent_A } }
545     \l__pdfmeta_tmpb_tl
546     \prop_if_exist:cTF {c__pdfmeta_colorprofile_\l__pdfmeta_tmpb_tl}
547     {
548         \exp_args:NV \__pdfmeta_embed_colorprofile:n \l__pdfmeta_tmpb_tl
549         \prop_map_inline:Nn \g__pdfmeta_outputintents_prop
550         {
551             \exp_args:NV
552             \__pdfmeta_write_outputintent:nn
553             \l__pdfmeta_tmpb_tl
554             { #1 }
555         }
556     }
557     {
558         \msg_warning:nne{pdfmeta}{colorprofile-undefined}{\l__pdfmeta_tmpb_tl}
559     }
560 }
561 }

```

### 3.2 Regression test

This is simply a copy of the backend function.

```

562 \cs_new_protected:Npn \pdfmeta_set_regression_data:
563 { \__pdf_backend_set_regression_data: }

```

## 4 XMP-Metadata implementation

`\g__pdfmeta_xmp_bool` This boolean decides if the metadata are included

```

564 \bool_new:N\g__pdfmeta_xmp_bool
565 \bool_gset_true:N \g__pdfmeta_xmp_bool

```

*(End of definition for \g\_\_pdfmeta\_xmp\_bool.)*

Preset the two fields to avoid problems with standards.

```

566 \hook_gput_code:nnn{pdfmanagement/add}{pdfmanagement}
567 {
568     \pdfmanagement_add:nne {Info}{Producer}{(\c_sys_engine_exec_str-\c_sys_engine_version_str)}
569     \pdfmanagement_add:nne {Info}{Creator}{(LaTeX)}
570 }

```

### 4.1 New document keys

```

571 \keys_define:nn { document / metadata }
572 {
573     _pdfstandard / X-4 .code:n =
574     {
575         \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-4}

```

```

576     \__pdfmeta_xmp_add_pdfxid:
577   },
578   _pdfstandard / X-4p .code:n =
579   {
580     \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-4p}
581     \__pdfmeta_xmp_add_pdfxid:
582   },
583   _pdfstandard / X-5g .code:n =
584   {
585     \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5g}
586     \__pdfmeta_xmp_add_pdfxid:
587   },
588   _pdfstandard / X-5n .code:n =
589   {
590     \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5n}
591     \__pdfmeta_xmp_add_pdfxid:
592   },
593   _pdfstandard / X-5pg .code:n =
594   {
595     \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5pg}
596     \__pdfmeta_xmp_add_pdfxid:
597   },
598   _pdfstandard / X-6 .code:n =
599   {
600     \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6p}
601     \__pdfmeta_xmp_add_pdfxid:
602   },
603   _pdfstandard / X-6n .code:n =
604   {
605     \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6n}
606     \__pdfmeta_xmp_add_pdfxid:
607   },
608   _pdfstandard / X-6p .code:n =
609   {
610     \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6p}
611     \__pdfmeta_xmp_add_pdfxid:
612   },
613   _pdfstandard / UA-1 .code:n =
614   {
615     \AddToDocumentProperties [document]{pdfstandard-UA}{{1}}{}
616     \AddToHook{begindocument/before}
617     {
618       \prop_gput:Nnn \g__pdfmeta_standard_prop {omit_CID}{}
619       \pdf_version_compare:NnF < {2.0}
620       {
621         \msg_warning:nneee
622         {pdf}{wrong-pdfversion}
623         {\pdf_version:}{high}{UA-1}
624       }
625     }
626   },

```

currently it is not possible to merge requirements - these need some thoughts as every standard has some common keys like the name or the yes. We therefore add some



requirements manually.

```
627 _pdfstandard / UA-2 .code:n =
628 {
629   \AddToDocumentProperties [document]{pdfstandard-UA}{2}{2024}
630   \AddToHook{begindocument/before}
631   {\prop_gput:Nnn \g__pdfmeta_standard_prop {Trailer_no_Info}{}}
632   \AddToHook{begindocument/before}
633   {
634     \__pdfmeta_xmp_wtpdf_accessibility_declaration:
635     \__pdfmeta_xmp_wtpdf_reuse_declaration:
636     \pdf_version_compare:NnT < {2.0}
637     {
638       \msg_warning:nneee
639       {pdf}{wrong-pdfversion}
640       {\pdf_version:}{low}{UA-2}
641     }
642   }
643 },
644 xmp .choice:,
645 xmp / true .code:n = { \bool_gset_true:N \g__pdfmeta_xmp_bool },
646 xmp / false .code:n = { \bool_gset_false:N \g__pdfmeta_xmp_bool},
647 xmp .default:n = true,
```

These keys allow to disable or force the wtpdf declarations. Currently the content can not be changed and once they have been disabled there are gone. This will perhaps change.

```
648 xmp / wtpdf .code:n =
649 {
650   \keys_set:nn {__pdfmeta/xmp}{#1}
651 },
652 }
653 \keys_define:nn {__pdfmeta/xmp}
654 {
655   reuse .choice:,
656   reuse / true .code:n = \__pdfmeta_xmp_wtpdf_reuse_declaration:,
657   reuse / false .code:n =
658   {
659     \cs_set_eq:NN \__pdfmeta_xmp_wtpdf_reuse_declaration: \prg_do_nothing:
660   },
661   accessibility .choice:,
662   accessibility / true .code:n = \__pdfmeta_xmp_wtpdf_accessibility_declaration:,
663   accessibility /false .code:n =
664   {
665     \cs_set_eq:NN \__pdfmeta_xmp_wtpdf_accessibility_declaration: \prg_do_nothing:
666   },
667 }
```

XMP debugging option

```
668 \bool_new:N \g__pdfmeta_xmp_export_bool
669 \str_new:N \g__pdfmeta_xmp_export_str
670
671 \keys_define:nn { document / metadata }
672 {
673   ,debug / xmp-export .choice:
674   ,debug / xmp-export / true .code:n=
675   {
```

```

676     \bool_gset_true:N \g__pdfmeta_xmp_export_bool
677     \str_gset_eq:NN \g__pdfmeta_xmp_export_str \c_sys_jobname_str
678   }
679   ,debug / xmp-export / false .code:n =
680   {
681     \bool_gset_false:N \g__pdfmeta_xmp_export_bool
682   }
683   ,debug / xmp-export /unknown .code:n =
684   {
685     \bool_gset_true:N \g__pdfmeta_xmp_export_bool
686     \str_gset:Nn \g__pdfmeta_xmp_export_str { #1 }
687   }
688   ,debug / xmp-export .default:n = true
689 }

```

## 4.2 Messages

```

690 \msg_new:nnn{pdfmeta}{xmp-defined}{The~XMP~#1~'#2'~is~already~declared}
691 \msg_new:nnn{pdfmeta}{xmp-undefined}{The~XMP~#1~'#2'~is~undefined}
692 \msg_new:nnn{pdfmeta}{colorprofile-undefined}{The~colorprofile~'#1'~is~unknown}

```

## 4.3 Some helper commands

### 4.3.1 Generate a BOM

`\__pdfmeta_xmp_generate_bom:`

```

693 \bool_lazy_or:nnTF
694 { \sys_if_engine_luatex_p: }
695 { \sys_if_engine_xetex_p: }
696 {
697   \cs_new:Npn \__pdfmeta_xmp_generate_bom:
698     { \char_generate:nn {"FEFF"}{12} }
699 }
700 {
701   \cs_new:Npn \__pdfmeta_xmp_generate_bom:
702     {
703       \char_generate:nn {"EF"}{12}
704       \char_generate:nn {"BB"}{12}
705       \char_generate:nn {"BF"}{12}
706     }
707 }

```

*(End of definition for \\_\_pdfmeta\_xmp\_generate\_bom:.)*

### 4.3.2 Indentation

We provide a command which indents the xml based on a counter, and one which accepts a fix number. The counter can be increased and decreased.

`\l__pdfmeta_xmp_indent_int`

```

708 \int_new:N \l__pdfmeta_xmp_indent_int

```

*(End of definition for \l\_\_pdfmeta\_xmp\_indent\_int.)*

```

    \_pdfmeta_xmp_indent:
    \_pdfmeta_xmp_indent:n
    \_pdfmeta_xmp_incr_indent:
    \_pdfmeta_xmp_decr_indent:
709 \cs_new:Npn \_pdfmeta_xmp_indent:
710 {
711   \iow_newline:
712   \prg_replicate:nn {\_pdfmeta_xmp_indent_int}{\c_space_tl}
713 }
714
715 \cs_new:Npn \_pdfmeta_xmp_indent:n #1
716 {
717   \iow_newline:
718   \prg_replicate:nn {#1}{\c_space_tl}
719 }
720
721 \cs_new_protected:Npn \_pdfmeta_xmp_incr_indent:
722 {
723   \int_incr:N \_pdfmeta_xmp_indent_int
724 }
725
726 \cs_new_protected:Npn \_pdfmeta_xmp_decr_indent:
727 {
728   \int_decr:N \_pdfmeta_xmp_indent_int
729 }

```

*(End of definition for \\_pdfmeta\_xmp\_indent: and others.)*

### 4.3.3 Date and time handling

If the date is given in PDF format we have to split it to create the XMP format. We use a precompiled regex for this. To some extent the regex can also handle incomplete dates.

```

\_pdfmeta_xmp_date_regex
730 \regex_new:N \_pdfmeta_xmp_date_regex
731 \regex_set:Nn \_pdfmeta_xmp_date_regex
732 {D:(\d{4})(\d{2})(\d{2})(\d{2})?(\d{2})?(\d{2})?([Z+|-])?(?:\d{2}\')?(?:\d{2}\')?}

```

*(End of definition for \\_pdfmeta\_xmp\_date\_regex.)*

\\_pdfmeta\_xmp\_date\_split:nN This command takes a date in PDF format, splits it with the regex and stores the captures in a sequence.

```

733 \cs_new_protected:Npn \_pdfmeta_xmp_date_split:nN #1 #2 % #1 date, #2 seq
734 {
735   \regex_split:NnN \_pdfmeta_xmp_date_regex {#1} #2
736 }
737 \cs_generate_variant:Nn \_pdfmeta_xmp_date_split:nN {VN,eN}

```

*(End of definition for \\_pdfmeta\_xmp\_date\_split:nN.)*

\\_pdfmeta\_xmp\_print\_date:N This prints the date stored in a sequence as created by the previous command.

```

738 \cs_new:Npn \_pdfmeta_xmp_print_date:N #1 % seq
739 {
740   \tl_if_blank:eTF { \seq_item:Nn #1 {1} }
741   {
742     \seq_item:Nn #1 {2} %year

```

```

743     -
744     \seq_item:Nn #1 {3} %month
745     -
746     \seq_item:Nn #1 {4} % day
747     \tl_if_blank:eF
748     { \seq_item:Nn #1 {5} }
749     { T \seq_item:Nn #1 {5} } %hour
750     \tl_if_blank:eF
751     { \seq_item:Nn #1 {6} }
752     { : \seq_item:Nn #1 {6} } %minutes
753     \tl_if_blank:eF
754     { \seq_item:Nn #1 {7} }
755     { : \seq_item:Nn #1 {7} } %seconds
756     \seq_item:Nn #1 {8} %Z,+,-
757     \seq_item:Nn #1 {9}
758     \tl_if_blank:eF
759     { \seq_item:Nn #1 {10} }
760     { : \seq_item:Nn #1 {10} }
761   }
762   {
763     \seq_item:Nn #1 {1}
764   }
765 }

```

*(End of definition for \\_\_pdfmeta\_xmp\_print\_date:N.)*

`\l__pdfmeta_xmp_currentdate_tl` The tl var contains the date of the log-file in PDF format, the seq the result split with  
`\l__pdfmeta_xmp_currentdate_seq` the regex.

```

766 \tl_new:N \l__pdfmeta_xmp_currentdate_tl
767 \seq_new:N \l__pdfmeta_xmp_currentdate_seq

```

*(End of definition for \l\_\_pdfmeta\_xmp\_currentdate\_tl and \l\_\_pdfmeta\_xmp\_currentdate\_seq.)*

`\__pdfmeta_xmp_date_get:nNN` This checks a document property and if empty uses the current date.

```

768 \cs_new_protected:Npn \__pdfmeta_xmp_date_get:nNN #1 #2 #3
769   %#1 property, #2 tl var with PDF date, #3 seq for split date
770   {
771     \tl_set:Ne #2 { \GetDocumentProperties{#1} }
772     \tl_if_blank:VTF #2
773     {
774       \seq_set_eq:NN #3 \l__pdfmeta_xmp_currentdate_seq
775       \tl_set_eq:NN #2 \l__pdfmeta_xmp_currentdate_tl
776     }
777     {
778       \__pdfmeta_xmp_date_split:VN #2 #3
779     }
780   }

```

*(End of definition for \\_\_pdfmeta\_xmp\_date\_get:nNN.)*

#### 4.3.4 UUID

We need a command to generate an uuid

`\_pdfmeta_xmp_create_uid:nN`

```
781 \cs_new_protected:Npn \_pdfmeta_xmp_create_uid:nN #1 #2
782 {
783   \str_set:Ne#2 {\str_lowercase:f{\tex_mdffivesum:D{#1}}}
784   \str_set:Ne#2
785     {
786       \str_range:Nnn #2{1}{8}
787       -\str_range:Nnn#2{9}{12}
788       -4\str_range:Nnn#2{13}{15}
789       -8\str_range:Nnn#2{16}{18}
790       -\str_range:Nnn#2{19}{30}
791     }
792 }
```

*(End of definition for \\_pdfmeta\_xmp\_create\_uid:nN.)*

### 4.3.5 Purifying and escaping of strings

`\_pdfmeta_xmp_sanitize:nN`

We have to sanitize the user input. For this we pass it through `\text_purify` and then replace a few special chars.

```
793 \cs_new_protected:Npn \_pdfmeta_xmp_sanitize:nN #1 #2
794 %#1 input string, #2 str with the output
795 {
796   \group_begin:
797   \text_declare_purify_equivalent:Nn \& {\tl_to_str:N & }
798   \text_declare_purify_equivalent:Nn \texttilde {\c_tilde_str}
799   \tl_set:Ne \l__pdfmeta_tmpa_tl { \text_purify:n {#1} }
800   \str_gset:Ne \g__pdfmeta_tmpa_str { \tl_to_str:N \l__pdfmeta_tmpa_tl }
801   \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {\&}{&}
802   \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {<}{&lt;}
803   \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {>}{&gt;}
804   \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {"}{&quot;}
805   \group_end:
806   \str_set_eq:NN #2 \g__pdfmeta_tmpa_str
807 }
808
809 \cs_generate_variant:Nn\_pdfmeta_xmp_sanitize:nN {VN}
```

*(End of definition for \\_pdfmeta\_xmp\_sanitize:nN.)*

## 4.4 Language handling

The language of the metadata is used in various attributes, so we store it in command.

`\l__pdfmeta_xmp_doclang_tl`

`\l__pdfmeta_xmp_metalang_tl`

```
810 \tl_new:N \l__pdfmeta_xmp_doclang_tl
811 \tl_new:N \l__pdfmeta_xmp_metalang_tl
```

*(End of definition for \l\_\_pdfmeta\_xmp\_doclang\_tl and \l\_\_pdfmeta\_xmp\_metalang\_tl.)*

The language is retrieved at the start of the packet. We assume that `lang` is always set and so don't use the x-default value of `hyperxmp`.

`\l__pdfmeta_xmp_lang_regex`

```
812 \regex_new:N\l__pdfmeta_xmp_lang_regex
813 \regex_set:Nn\l__pdfmeta_xmp_lang_regex {\A\[[A-Za-z\-\_]+\]\(.*)}
```

(End of definition for `\l__pdfmeta_xmp_lang_regex`.)

```
814 \cs_new_protected:Npn \__pdfmeta_xmp_lang_get:nNN #1 #2 #3
815 % #1 text, #2 tl var for lang match (or default), #3 tl var for text
816 {
817   \regex_extract_once:NnN \l__pdfmeta_xmp_lang_regex {#1}\l__pdfmeta_tmpa_seq
818   \seq_if_empty:NTF \l__pdfmeta_tmpa_seq
819     {
820       \tl_set:Nn #2 \l__pdfmeta_xmp_metalang_tl
821       \tl_set:Nn #3 {#1}
822     }
823     {
824       \tl_set:Ne #2 {\seq_item:Nn\l__pdfmeta_tmpa_seq{2}}
825       \tl_set:Ne #3 {\seq_item:Nn\l__pdfmeta_tmpa_seq{3}}
826     }
827 }
828 \cs_generate_variant:Nn \__pdfmeta_xmp_lang_get:nNN {eNN,VNN}
```

## 4.5 Filling the packet

This tl var that holds the whole packet

`\g__pdfmeta_xmp_packet_tl`

```
829 \tl_new:N \g__pdfmeta_xmp_packet_tl
(End of definition for \g__pdfmeta_xmp_packet_tl.)
```

### 4.5.1 Helper commands to add lines and lists

`\__pdfmeta_xmp_add_packet_chunk:n`

This is the most basic command. It is meant to produce a line and will use the current indent.

```
830 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_chunk:n #1
831 {
832   \tl_gput_right:Ne\g__pdfmeta_xmp_packet_tl
833   {
834     \__pdfmeta_xmp_indent: \exp_not:n{#1}
835   }
836 }
837 \cs_generate_variant:Nn \__pdfmeta_xmp_add_packet_chunk:n {e}
```

(End of definition for `\__pdfmeta_xmp_add_packet_chunk:n`.)

`\__pdfmeta_xmp_add_packet_chunk:nN`

This is the most basic command. It is meant to produce a line and will use the current indent.

```
838 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_chunk:nN #1 #2
839 {
840   \tl_put_right:Ne#2
841   {
842     \__pdfmeta_xmp_indent: \exp_not:n{#1}
843   }
844 }
845 \cs_generate_variant:Nn \__pdfmeta_xmp_add_packet_chunk:nN {eN}
```

(End of definition for `\__pdfmeta_xmp_add_packet_chunk:nN`.)

`\_pdfmeta_xmp_add_packet_open:nn` This commands opens a xml structure and increases the indent.

```

846 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_open:nn #1 #2 %#1 prefix #2 name
847 {
848   \_pdfmeta_xmp_add_packet_chunk:n {<#1:#2>}
849   \_pdfmeta_xmp_incr_indent:
850 }
851 \cs_generate_variant:Nn \_pdfmeta_xmp_add_packet_open:nn {ne}

```

*(End of definition for \\_pdfmeta\_xmp\_add\_packet\_open:nn.)*

`\_pdfmeta_xmp_add_packet_open_attr:nnn` This commands opens a xml structure too but allows also to give an attribute.

```

852 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_open_attr:nnn #1 #2 #3
853   %#1 prefix #2 name #3 attr
854 {
855   \_pdfmeta_xmp_add_packet_chunk:n {<#1:#2~#3>}
856   \_pdfmeta_xmp_incr_indent:
857 }
858 \cs_generate_variant:Nn \_pdfmeta_xmp_add_packet_open_attr:nnn {nne}

```

*(End of definition for \\_pdfmeta\_xmp\_add\_packet\_open\_attr:nnn.)*

`\_pdfmeta_xmp_add_packet_close:nn` This closes a structure and decreases the indent.

```

859 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_close:nn #1 #2 %#1 prefix #2:name
860 {
861   \_pdfmeta_xmp_decr_indent:
862   \_pdfmeta_xmp_add_packet_chunk:n {</#1:#2>}
863 }

```

*(End of definition for \\_pdfmeta\_xmp\_add\_packet\_close:nn.)*

`\_pdfmeta_xmp_add_packet_line:nnn` This will produce a full line with open and closing xml. The content is sanitized. We test if there is content to be able to suppress data which has not be set.

```

864 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_line:nnn #1 #2 #3
865   %#1 prefix #2 name #3 content
866 {
867   \tl_if_blank:nF {#3}
868   {
869     \_pdfmeta_xmp_sanitize:nN {#3}\l_pdfmeta_tmpa_str
870     \_pdfmeta_xmp_add_packet_chunk:e {<#1:#2>\l_pdfmeta_tmpa_str</#1:#2>}
871   }
872 }
873 \cs_generate_variant:Nn \_pdfmeta_xmp_add_packet_line:nnn {nne,nnV,nee}

```

*(End of definition for \\_pdfmeta\_xmp\_add\_packet\_line:nnn.)*

`\_pdfmeta_xmp_add_packet_line:nnnN` This will produce a full line with open and closing xml and store it in the given tl-var. This allows to prebuild blocks and then to test if there are empty. The content is sanitized. We test if there is content to be able to suppress data which has not be set.

```

874 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_line:nnnN #1 #2 #3 #4
875   %#1 prefix #2 name #3 content #4 tl_var to prebuilt.
876 {
877   \tl_if_blank:nF {#3}
878   {
879     \_pdfmeta_xmp_sanitize:nN {#3}\l_pdfmeta_tmpa_str

```

```

880     \_pdfmeta_xmp_add_packet_chunk:eN {<#1:#2>\l__pdfmeta_tmpa_str</#1:#2>} #4
881     }
882   }
883 \cs_generate_variant:Nn \_pdfmeta_xmp_add_packet_line:nnnN {nneN}

```

*(End of definition for \\_pdfmeta\_xmp\_add\_packet\_line:nnnN.)*

\\_pdfmeta\_xmp\_add\_packet\_line\_attr:nnnn

A similar command with attribute

```

884 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_line_attr:nnnn #1 #2 #3 #4
885   %#1 prefix #2 name #3 attribute #4 content
886   {
887     \tl_if_blank:nF {#4}
888     {
889       \_pdfmeta_xmp_sanitize:nN {#4}\l__pdfmeta_tmpa_str
890       \_pdfmeta_xmp_add_packet_chunk:e {<#1:#2-#3>\l__pdfmeta_tmpa_str</#1:#2>}
891     }
892   }
893 \cs_generate_variant:Nn \_pdfmeta_xmp_add_packet_line_attr:nnnn {nnee,nneV}

```

*(End of definition for \\_pdfmeta\_xmp\_add\_packet\_line\_attr:nnnn.)*

\\_pdfmeta\_xmp\_add\_packet\_line\_default:nnnn

```

894 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_line_default:nnnn #1 #2 #3 #4
895   % #1 prefix #2 name #3 default #4 content
896   {
897     \tl_if_blank:nTF { #4 }
898     {
899       \tl_set:Nn \l__pdfmeta_tmpa_tl {#3}
900     }
901     {
902       \tl_set:Nn \l__pdfmeta_tmpa_tl {#4}
903     }
904     \_pdfmeta_xmp_add_packet_line:nnV {#1}{#2}\l__pdfmeta_tmpa_tl
905   }
906 \cs_generate_variant:Nn \_pdfmeta_xmp_add_packet_line_default:nnnn {nnee}

```

*(End of definition for \\_pdfmeta\_xmp\_add\_packet\_line\_default:nnnn.)*

Some data are stored as unordered (Bag) or ordered lists (Seq) or (Alt). The first variant are for simple text without language support:

```

907 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_list_simple:nnnn #1 #2 #3 #4
908   %#1 prefix, #2 name, #3 type (Seq/Bag/Alt) #4 a clist
909   {
910     \clist_if_empty:nF { #4 }
911     {
912       \_pdfmeta_xmp_add_packet_open:nn {#1}{#2}
913       \_pdfmeta_xmp_add_packet_open:nn {rdf}{#3}
914       \clist_map_inline:nn {#4}
915       {
916         \_pdfmeta_xmp_add_packet_line:nnn
917         {rdf}{li}{##1}
918       }
919       \_pdfmeta_xmp_add_packet_close:nn{rdf}{#3}
920       \_pdfmeta_xmp_add_packet_close:nn {#1}{#2}
921     }

```



```

922     }
923 \cs_generate_variant:Nn \_pdfmeta_xmp_add_packet_list_simple:nmmm {nnnV,nnne}

```

Here we check also for the language.

```

924 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_list:nmmm #1 #2 #3 #4
925   % #1 prefix, #2 name, #3 type (Seq/Bag/Alt) #4 a clist
926   {
927     \clist_if_empty:nF { #4 }
928     {
929       \_pdfmeta_xmp_add_packet_open:nn {#1}{#2}
930       \_pdfmeta_xmp_add_packet_open:nn {rdf}{#3}
931       \clist_map_inline:nn {#4}
932       {
933         \_pdfmeta_xmp_lang_get:nNN {##1}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl

```

change 2024-02-22. There should be if possible a x-default entry as some viewers need that. So if the language is equal to the main language we use that. This assumes that the user hasn't marked every entry as some other language!

```

934         \tl_if_eq:eeTF{\l__pdfmeta_tmpa_tl}{\l__pdfmeta_xmp_metalang_tl}
935         {
936           \_pdfmeta_xmp_add_packet_line_attr:nneV
937           {rdf}{li}{xml:lang="x-default" }\l__pdfmeta_tmpb_tl
938         }
939         {
940           \_pdfmeta_xmp_add_packet_line_attr:nneV
941           {rdf}{li}{xml:lang="\l__pdfmeta_tmpa_tl" }\l__pdfmeta_tmpb_tl
942         }
943       }
944       \_pdfmeta_xmp_add_packet_close:nn{rdf}{#3}
945       \_pdfmeta_xmp_add_packet_close:nn {#1}{#2}
946     }
947   }
948 \cs_generate_variant:Nn \_pdfmeta_xmp_add_packet_list:nmmm {nnne}

```

#### 4.5.2 Building the main packet

`\_pdfmeta_xmp_build_packet:` This is the main command to build the packet. As data has to be set and collected first, it will be expanded rather late in the document.

```

949 \cs_new_protected:Npn \_pdfmeta_xmp_build_packet:
950   {

```

Get the main languages

```

951   \tl_set:Ne \l__pdfmeta_xmp_doclang_tl {\GetDocumentProperties{document/lang}}
952   \tl_set:Ne \l__pdfmeta_xmp_metalang_tl {\GetDocumentProperties{hyperref/pdfmetalang}}
953   \tl_if_blank:VT \l__pdfmeta_xmp_metalang_tl
954   { \cs_set_eq:NN \l__pdfmeta_xmp_metalang_tl\l__pdfmeta_xmp_doclang_tl}

```

we preprocess a number of data to be able to suppress them and their schema if there are unused. Currently only done for iptc

```

955   \_pdfmeta_xmp_build_iptc_data:N \l__pdfmeta_xmp_iptc_data_tl
956   \tl_if_empty:NT \l__pdfmeta_xmp_iptc_data_tl
957   {
958     \seq_remove_all:Nn \l__pdfmeta_xmp_schema_seq { Iptc4xmpCore }
959   }

```

The start of the package. No need to try to juggle with catcode, this is fix text

```

960 \_pdfmeta_xmp_add_packet_chunk:e
961 {<?xpacket~begin="\_pdfmeta_xmp_generate_bom:"~id="W5MOMpCehiHzreSzNTczkc9d"?>}
962 \_pdfmeta_xmp_add_packet_open:nn{x}{xmpmeta~xmlns:x="adobe:ns:meta/"}
963 \_pdfmeta_xmp_add_packet_open:ne{rdf}
964 {RDF~xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\c_hash_str"}

```

The rdf namespaces

```

965 \_pdfmeta_xmp_add_packet_open_attr:nne
966 {rdf}{Description}{rdf:about="" \g__pdfmeta_xmp_xmlns_tl}

```

The extensions

```

967 \_pdfmeta_xmp_add_packet_open:nn{pdfaExtension}{schemas}
968 \_pdfmeta_xmp_add_packet_open:nn {rdf}{Bag}
969 \seq_map_inline:Nn \l__pdfmeta_xmp_schema_seq
970 {
971 \tl_use:c { g__pdfmeta_xmp_schema_##1_tl }
972 }
973 \_pdfmeta_xmp_add_packet_close:nn {rdf}{Bag}
974 \_pdfmeta_xmp_add_packet_close:nn {pdfaExtension}{schemas}

```

Now starts the part with the data.

```

975 % data
976 \_pdfmeta_xmp_build_pdf:
977 \_pdfmeta_xmp_build_xmpRights:
978 \_pdfmeta_xmp_build_standards: %pdfaid,pdfxid,pdfuaid
979 \_pdfmeta_xmp_build_pdfd:
980 \_pdfmeta_xmp_build_dc:
981 \_pdfmeta_xmp_build_photoshop:
982 \_pdfmeta_xmp_build_xmp:
983 \_pdfmeta_xmp_build_xmpMM:
984 \_pdfmeta_xmp_build_prism:
985 \_pdfmeta_xmp_build_iptc:
986 \_pdfmeta_xmp_build_user: %user additions
987 % end
988 \_pdfmeta_xmp_add_packet_close:nn {rdf}{Description}
989 \_pdfmeta_xmp_add_packet_close:nn {rdf}{RDF}
990 \_pdfmeta_xmp_add_packet_close:nn {x}{xmpmeta}
991 \int_set:Nn \l__pdfmeta_xmp_indent_int{20}
992 \prg_replicate:nn{10}{\_pdfmeta_xmp_add_packet_chunk:n {}}
993 \int_zero:N \l__pdfmeta_xmp_indent_int
994 \_pdfmeta_xmp_add_packet_chunk:n {<?xpacket~end="w"?>}
995 }

```

*(End of definition for \\_pdfmeta\_xmp\_build\_packet:.)*

## 4.6 Building the chunks: rdf namespaces

This is the list of external names spaces. They are rather simple, and we store them directly into a string. Special chars should be escaped properly, see e.g. `\c_hash_str` for the hash.

```

\g__pdfmeta_xmp_xmlns_tl
\g__pdfmeta_xmp_xmlns_prop

```

The string will hold the prepared chunk, the prop stores the name spaces so that one can check on the user level for duplicates.

```

996 \str_new:N \g__pdfmeta_xmp_xmlns_tl
997 \prop_new:N \g__pdfmeta_xmp_xmlns_prop

```

(End of definition for `\g_pdfmeta_xmp_xmlns_tl` and `\g_pdfmeta_xmp_xmlns_prop`.)

`\__pdfmeta_xmp_xmlns_new:nn`

```
998 \cs_new_protected:Npn \__pdfmeta_xmp_xmlns_new:nn #1 #2
999   {
1000     \prop_gput:Nnn \g_pdfmeta_xmp_xmlns_prop {#1}{#2}
1001     \tl_gput_right:Ne \g_pdfmeta_xmp_xmlns_tl
1002     {
1003       \__pdfmeta_xmp_indent:n{4} xmlns:\exp_not:n{#1="#2"}
1004     }
1005   }
```

(End of definition for `\__pdfmeta_xmp_xmlns_new:nn`.)

Now we fill the data. The list is more or less the same as in hyperxmp. The `pdfxid` entry is only added if an X standard is used, see issue #50 and the schema below.

```
1006 \__pdfmeta_xmp_xmlns_new:nn {pdf} {http://ns.adobe.com/pdf/1.3/}
1007 \__pdfmeta_xmp_xmlns_new:nn {xmpRights}{http://ns.adobe.com/xap/1.0/rights/}
1008 \__pdfmeta_xmp_xmlns_new:nn {dc} {http://purl.org/dc/elements/1.1/}
1009 \__pdfmeta_xmp_xmlns_new:nn {photoshop}{http://ns.adobe.com/photoshop/1.0/}
1010 \__pdfmeta_xmp_xmlns_new:nn {xmp} {http://ns.adobe.com/xap/1.0/}
1011 \__pdfmeta_xmp_xmlns_new:nn {xmpMM} {http://ns.adobe.com/xap/1.0/mm/}
1012 \__pdfmeta_xmp_xmlns_new:nn {stEvt}
1013   {http://ns.adobe.com/xap/1.0/sType/ResourceEvent\c_hash_str}
1014 \__pdfmeta_xmp_xmlns_new:nn {pdfaid} {http://www.aiim.org/pdfa/ns/id/}
1015 \__pdfmeta_xmp_xmlns_new:nn {pdfuaid} {http://www.aiim.org/pdfua/ns/id/}
1016 \__pdfmeta_xmp_xmlns_new:nn {pdfx} {http://ns.adobe.com/pdfx/1.3/}
1017 %\__pdfmeta_xmp_xmlns_new:nn {pdfxid} {http://www.npes.org/pdfx/ns/id/}
1018 \__pdfmeta_xmp_xmlns_new:nn {prism} {http://prismstandard.org/namespaces/basic/3.0/}
1019 %\__pdfmeta_xmp_xmlns_new:nn {jav} {http://www.niso.org/schemas/jav/1.0/}
1020 %\__pdfmeta_xmp_xmlns_new:nn {xmpTPg} {http://ns.adobe.com/xap/1.0/t/pg/}
1021 \__pdfmeta_xmp_xmlns_new:nn {stFnt} {http://ns.adobe.com/xap/1.0/sType/Font\c_hash_str}
1022 \__pdfmeta_xmp_xmlns_new:nn {Iptc4xmpCore}{http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
1023 \__pdfmeta_xmp_xmlns_new:nn {pdfaExtension}{http://www.aiim.org/pdfa/ns/extension/}
1024 \__pdfmeta_xmp_xmlns_new:nn {pdfaSchema}{http://www.aiim.org/pdfa/ns/schema\c_hash_str}
1025 \__pdfmeta_xmp_xmlns_new:nn {pdfaProperty}{http://www.aiim.org/pdfa/ns/property\c_hash_str}
1026 \__pdfmeta_xmp_xmlns_new:nn {pdfaType} {http://www.aiim.org/pdfa/ns/type\c_hash_str}
1027 \__pdfmeta_xmp_xmlns_new:nn {pdfaField}{http://www.aiim.org/pdfa/ns/field\c_hash_str}
```

## 4.7 Building the chunks: Extensions

In this part local name spaces or additional names in a name space can be declared. A “schema” declaration consist of the declaration of the name, uri and prefix which then surrounds a bunch of property declarations. The current code doesn’t support all syntax options but sticks to what is used in hyperxmp and pdfx. If needed it can be extended later.

`\l_pdfmeta_xmp_schema_seq` This variable will hold the list of prefix so that we can loop to produce the final XML

```
1028 \seq_new:N \l_pdfmeta_xmp_schema_seq
```

(End of definition for `\l_pdfmeta_xmp_schema_seq`.)

\\_pdfmeta\_xmp\_schema\_new:nnn

With this command a new schema can be declared. The main tl contains the XML wrapper code, it then includes the list of properties which are created with the next command.

```
1029 \cs_new_protected:Npn \_pdfmeta_xmp_schema_new:nnn #1 #2 #3
1030   %#1 name #2 prefix, #3 text
1031   {
1032     \tl_if_exist:cTF { g__pdfmeta_xmp_schema_#2_tl }
1033     {
1034       \msg_warning:nnnn{pdfmeta}{xmp-defined}{schema}{#2}
1035     }
1036     {
1037       \seq_put_right:Nn \l__pdfmeta_xmp_schema_seq { #2 }
1038       \tl_new:c { g__pdfmeta_xmp_schema_#2_tl }
1039       \tl_new:c { g__pdfmeta_xmp_schema_#2_properties_tl }
1040       \tl_gput_right:cn { g__pdfmeta_xmp_schema_#2_tl }
1041       {
1042         \_pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1043         \_pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{schema}{#1}
1044         \_pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{prefix}{#2}
1045         \_pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{namespaceURI}{#3}
1046         \_pdfmeta_xmp_add_packet_open:nn {pdfaSchema}{property}
1047         \_pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1048         \tl_use:c { g__pdfmeta_xmp_schema_#2_properties_tl }
1049         \_pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1050         \_pdfmeta_xmp_add_packet_close:nn {pdfaSchema}{property}
1051         \cs_if_exist_use:c {\_pdfmeta_xmp_schema_#2_additions:}
1052         \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1053       }
1054     }
1055   }
```

(End of definition for \\_pdfmeta\_xmp\_schema\_new:nnn.)

\\_pdfmeta\_xmp\_property\_new:nnnnn

This adds a property to a schema.

```
1056 \prop_new:N\g__pdfmeta_xmp_schema_property_prop
1057 \cs_new_protected:Npn \_pdfmeta_xmp_property_new:nnnnn #1 #2 #3 #4 #5 %
1058   %#1 schema #2 name, #3 type, #4 category #5 description
1059   {
1060     \tl_if_exist:cTF { g__pdfmeta_xmp_schema_#1_properties_tl }
1061     {
1062       \prop_get:NeNF \g__pdfmeta_xmp_schema_property_prop {#1:#2}\l__pdfmeta_tmpa_tl
1063       {
1064         \prop_gput:Nee \g__pdfmeta_xmp_schema_property_prop {#1:#2}{#3}
1065         \tl_gput_right:cn { g__pdfmeta_xmp_schema_#1_properties_tl }
1066         {
1067           \_pdfmeta_xmp_add_packet_open:nn {rdf}{li-rdf:parseType="Resource"}
1068           \_pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{name}{#2}
1069           \_pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{valueType}{#3}
1070           \_pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{category}{#4}
1071           \_pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{description}{#5}
1072           \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1073         }
1074       }
1075     }
```

```

1076     {
1077     \msg_warning:nnnn{pdfmeta}{xmp-undefined}{schema}{#1}
1078     }
1079 }

```

(End of definition for \\_\_pdfmeta\_xmp\_property\_new:nnnn.)

\\_\_pdfmeta\_xmp\_add\_packet\_field:nnn This adds a field to a schema.

```

1080 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_field:nnn #1 #2 #3 %
1081 %#1 name #2 valuetype #3 description
1082 {
1083   \__pdfmeta_xmp_add_packet_open_attr:nnn {rdf}{li}{rdf:parseType="Resource"}
1084   \__pdfmeta_xmp_add_packet_line:nnn {pdfaField}{name}{#1}
1085   \__pdfmeta_xmp_add_packet_line:nnn {pdfaField}{valueType}{#2}
1086   \__pdfmeta_xmp_add_packet_line:nnn {pdfaField}{description}{#3}
1087   \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1088 }

```

(End of definition for \\_\_pdfmeta\_xmp\_add\_packet\_field:nnn.)

#### 4.7.1 The extension data

The list of extension has been reviewed and compared with the list of namespaces which can be used in pdf/A-1<sup>7</sup>

[1] [https://www.pdfa.org/wp-content/uploads/2011/08/tn0008\\_predefined\\_xmp\\_properties\\_in\\_pdfa-1\\_2008-03-20.pdf](https://www.pdfa.org/wp-content/uploads/2011/08/tn0008_predefined_xmp_properties_in_pdfa-1_2008-03-20.pdf) and the content of the namespaces as listed here [2] <https://developer.adobe.com/xmp/docs/XMPNamespaces/pdf/>

**pdf** property: Trapped. We ignore it, it seems to validate without it.

**xmpMM** properties DocumentID, InstanceID, VersionID, Renditionclass declared by hyperxmp. Properties InstanceID and OriginalDocumentID declared by pdfx (pdfx.xmp) With the exception of OriginalDocumentID all are already allowed and predefined.

```

1089   \__pdfmeta_xmp_schema_new:nnn
1090     {XMP~Media~Management~Schema}
1091     {xmpMM}
1092     {http://ns.adobe.com/xap/1.0/mm/}
1093   \__pdfmeta_xmp_property_new:nnnnn
1094     {xmpMM}
1095     {OriginalDocumentID}
1096     {URI}
1097     {internal}
1098     {The~common~identifier~for~all~versions~and~renditions~of~a~document.}

```

**pdfaid** properties part and conformance are declared by hyperxmp, but no here as already in <http://www.aiim.org/pdfa/ns/id/>. But we declare year so that it can be used also with older A-standards.

<sup>7</sup>While A-1 builds on PDF 1.4 and so it probably no longer relevant, it is not quite clear if one can remove this for A-2 and newer, so we stay on the safe side.

pdfaid-(schema)

```
1099     \_pdfmeta_xmp_schema_new:nnn
1100         {PDF/A-Identification-Schema}
1101         {pdfaid}
1102         {http://www.aiim.org/pdfa/ns/id/}
1103     \_pdfmeta_xmp_property_new:nnnnn
1104         {pdfaid}
1105         {year}
1106         {Integer}
1107         {internal}
1108         {Year-of~standard}
1109     \_pdfmeta_xmp_property_new:nnnnn
1110         {pdfaid}
1111         {rev}
1112         {Integer}
1113         {internal}
1114         {Revision~year~of~standard}
```

*(End of definition for pdfaid-(schema).)*

**pdfuaid** here we need (?) to declare the property “part” and “rev”.

pdfuaid-(schema)

```
1115     \_pdfmeta_xmp_schema_new:nnn
1116         {PDF/UA-Universal-Accessibility-Schema}
1117         {pdfuaid}
1118         {http://www.aiim.org/pdfua/ns/id/}
1119     \_pdfmeta_xmp_property_new:nnnnn
1120         {pdfuaid}
1121         {part}
1122         {Integer}
1123         {internal}
1124         {Part-of~ISO~14289~standard}
1125     \_pdfmeta_xmp_property_new:nnnnn
1126         {pdfuaid}
1127         {rev}
1128         {Integer}
1129         {internal}
1130         {Revision~of~ISO~14289~standard}
```

*(End of definition for pdfuaid-(schema).)*

**pdfx** According to [1] not an allowed schema, but it seems to validate and allow to set the pdf/X version, hyperxmp declares here the properties `GTS_PDFXVersion` and `GTS_PDFXConformance`. Ignored as only relevant for older pdf/X version not supported by the pdfmanagement.

**pdfxid** we set this so that we can add the pdf/X version for pdf/X-4 and higher. This is only set if a pdf/X standard is used, see issue #50

pdfxid-(schema)

```

1131 \cs_new_protected:Npn \__pdfmeta_xmp_add_pdfxid:
1132 {
1133   \__pdfmeta_xmp_xmlns_new:nn {pdfxid} {http://www.npes.org/pdfx/ns/id/}
1134   \__pdfmeta_xmp_schema_new:nnn
1135     {PDF/X-ID~Schema}
1136     {pdfxid}
1137     {http://www.npes.org/pdfx/ns/id/}
1138   \__pdfmeta_xmp_property_new:nnnnn
1139     {pdfxid}
1140     {GTS_PDFXVersion}
1141     {Text}
1142     {internal}
1143     {ID-of-PDF/X-standard}
1144 }

```

*(End of definition for pdfxid-(schema).)*

### prism~(schema)

```

1145 \__pdfmeta_xmp_schema_new:nnn
1146   {PRISM~Basic~Metadata}
1147   {prism}
1148   {http://prismstandard.org/namespaces/basic/3.0/}
1149 \__pdfmeta_xmp_property_new:nnnnn
1150   {prism}
1151   {complianceProfile}
1152   {Text}
1153   {internal}
1154   {PRISM-specification-compliance-profile-to-which-this-document-adheres}
1155 \__pdfmeta_xmp_property_new:nnnnn
1156   {prism}
1157   {publicationName}
1158   {Text}
1159   {external}
1160   {Publication-name}
1161 \__pdfmeta_xmp_property_new:nnnnn
1162   {prism}
1163   {aggregationType}
1164   {Text}
1165   {external}
1166   {Publication-type}
1167 \__pdfmeta_xmp_property_new:nnnnn
1168   {prism}
1169   {bookEdition}
1170   {Text}
1171   {external}
1172   {Edition-of-the-book-in-which-the-document-was-published}
1173 \__pdfmeta_xmp_property_new:nnnnn
1174   {prism}
1175   {volume}
1176   {Text}
1177   {external}
1178   {Publication-volume-number}
1179 \__pdfmeta_xmp_property_new:nnnnn
1180   {prism}

```

```

1181     {number}
1182     {Text}
1183     {external}
1184     {Publication~issue~number~within~a~volume}
1185 \_pdfmeta_xmp_property_new:nnnnn
1186     {prism}
1187     {pageRange}
1188     {Text}
1189     {external}
1190     {Page~range~for~the~document~within~the~print~version~of~its~publication}
1191 \_pdfmeta_xmp_property_new:nnnnn
1192     {prism}
1193     {issn}
1194     {Text}
1195     {external}
1196     {ISSN~for~the~printed~publication~in~which~the~document~was~published}
1197 \_pdfmeta_xmp_property_new:nnnnn
1198     {prism}
1199     {eIssn}
1200     {Text}
1201     {external}
1202     {ISSN~for~the~electronic~publication~in~which~the~document~was~published}
1203 \_pdfmeta_xmp_property_new:nnnnn
1204     {prism}
1205     {isbn}
1206     {Text}
1207     {external}
1208     {ISBN~for~the~publication~in~which~the~document~was~published}
1209 \_pdfmeta_xmp_property_new:nnnnn
1210     {prism}
1211     {doi}
1212     {Text}
1213     {external}
1214     {Digital~Object~Identifier~for~the~document}
1215 \_pdfmeta_xmp_property_new:nnnnn
1216     {prism}
1217     {url}
1218     {URL}
1219     {external}
1220     {URL~at~which~the~document~can~be~found}
1221 \_pdfmeta_xmp_property_new:nnnnn
1222     {prism}
1223     {byteCount}
1224     {Integer}
1225     {internal}
1226     {Approximate~file~size~in~octets}
1227 \_pdfmeta_xmp_property_new:nnnnn
1228     {prism}
1229     {pageCount}
1230     {Integer}
1231     {internal}
1232     {Number~of~pages~in~the~print~version~of~the~document}
1233 \_pdfmeta_xmp_property_new:nnnnn
1234     {prism}

```



```

1235     {subtitle}
1236     {Text}
1237     {external}
1238     {Document's~subtitle}

```

(End of definition for prism~(schema).)

#### iptc\_(schema)C

```

1239     \__pdfmeta_xmp_schema_new:nnn
1240     {IPTC~Core~Schema}
1241     {Iptc4xmpCore}
1242     {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
1243     \__pdfmeta_xmp_property_new:nnnnn
1244     {Iptc4xmpCore}
1245     {CreatorContactInfo}
1246     {ContactInfo}
1247     {external}
1248     {Document~creator's~contact~information}
1249     \cs_new_protected:cpn { __pdfmeta_xmp_schema_Iptc4xmpCore_additions: }
1250     {
1251         \__pdfmeta_xmp_add_packet_open:nn{pdfaSchema}{valueType}
1252         \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1253         \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1254         \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{ContactInfo}
1255         \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
1256         {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
1257         \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{Iptc4xmpCore}
1258         \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
1259         {Basic~set~of~information~to~get~in~contact~with~a~person}
1260         \__pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
1261         \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1262         \__pdfmeta_xmp_add_packet_field:nnn{CiAdrCity}{Text}
1263         {Contact~information~city}
1264         \__pdfmeta_xmp_add_packet_field:nnn{CiAdrCtry}{Text}
1265         {Contact~information~country}
1266         \__pdfmeta_xmp_add_packet_field:nnn{CiAdrExtadr}{Text}
1267         {Contact~information~address}
1268         \__pdfmeta_xmp_add_packet_field:nnn{CiAdrPcode}{Text}
1269         {Contact~information~local~postal~code}
1270         \__pdfmeta_xmp_add_packet_field:nnn{CiAdrRegion}{Text}
1271         {Contact~information~regional~information~such~as~state~or~province}
1272         \__pdfmeta_xmp_add_packet_field:nnn{CiEmailWork}{Text}
1273         {Contact~information~email~address(es)}
1274         \__pdfmeta_xmp_add_packet_field:nnn{CiTelWork}{Text}
1275         {Contact~information~telephone~number(s)}
1276         \__pdfmeta_xmp_add_packet_field:nnn{CiUrlWork}{Text}
1277         {Contact~information~Web~URL(s)}
1278         \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1279         \__pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
1280         \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1281         \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1282         \__pdfmeta_xmp_add_packet_close:nn{pdfaSchema}{valueType}
1283     }

```

(End of definition for iptc (schema).)

**jav** : currently ignored

**declarations** The PDF Declarations mechanism allows creation and editing software to declare, via a PDF Declaration, a PDF file to be in conformance with a 3rd party specification or profile that may not be related to PDF technology. Their specification is for example described in <https://pdfa.org/wp-content/uploads/2019/09/PDF-Declarations.pdf>.

If declarations are added to the XMP-metadata they need (for pdf/A compliance) a schema declaration. We do not add it by default but define here a command to enable it. (This can be done in the document preamble as xmp is built only at the end.)

```
1284 \cs_new_protected:Npn \__pdfmeta_xmp_schema_enable_pdfd:
1285 {
1286   \__pdfmeta_xmp_xmlns_new:nn {pdfd}{http://pdfa.org/declarations/}
1287   \__pdfmeta_xmp_schema_new:nnn
1288     {PDF~Declarations~Schema}
1289     {pdfd}
1290     {http://pdfa.org/declarations/}
1291   \__pdfmeta_xmp_property_new:nnnnn
1292     {pdfd}
1293     {declarations}
1294     {Bag~declaration}
1295     {external}
1296     {An~unordered~array~of~PDF~Declaration~entries,~where~each~PDF~Declaration~represent
```

the values are complicated so we use the additions: method to add them.

```
1297 \cs_new_protected:cpn { __pdfmeta_xmp_schema_pdfd_additions: }
1298 {
1299   \__pdfmeta_xmp_add_packet_open:nn{pdfaSchema}{valueType}
1300   \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1301   \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1302   \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{claim}
1303   \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
1304     {http://pdfa.org/declarations/}
1305   \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{pdfd}
1306   \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
1307     {A~structure~describing~properties~of~an~individual~claim.}
1308   \__pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
1309   \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1310   \__pdfmeta_xmp_add_packet_field:nnn{claimReport}{Text}
1311     {A~URL~to~a~report~containing~details~of~the~specific~conformance~claim}
1312   \__pdfmeta_xmp_add_packet_field:nnn{claimCredentials}{Text}
1313     {The~claimant's~credentials.}
1314   \__pdfmeta_xmp_add_packet_field:nnn{claimDate}{Text}
1315     {A~date~identifying~when~the~claim~was~made.}
1316   \__pdfmeta_xmp_add_packet_field:nnn{claimBy}{Text}
1317     {The~name~of~the~organization~and/or~individual~and/or~software~making}
1318   \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1319   \__pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
```

```

1320     \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1321     \_pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1322     \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{declaration}
1323     \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
1324         {http://pdfa.org/declarations/}
1325     \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{pdfd}
1326     \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
1327         {A~structure~describing~a~single~PDF~ Declaration~asserting~conformance~v
1328     \_pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
1329     \_pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1330     \_pdfmeta_xmp_add_packet_field:nnn{conformsTo}{Text}
1331         {A~property~containing~a~URI~specifying~the~standard~or~profile~by~the
1332     \_pdfmeta_xmp_add_packet_field:nnn{claimData}{Bag~claim}
1333         {An~unordered~array~of~claim~data,~where~each~claim~identifies~the~natu
1334     \_pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1335     \_pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
1336     \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1337     \_pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1338     \_pdfmeta_xmp_add_packet_close:nn{pdfaSchema}{valueType}
1339 }

```

the schema should be added only once so disable it after use:

```

1340     \cs_gset_eq:NN \_pdfmeta_xmp_schema_enable_pdfd: \prg_do_nothing:
1341 }

```

## 4.8 The actual user / document data

### 4.8.1 pdf

This builds pdf related the data with the (prefix “pdf”).

```

\_pdfmeta_xmp_build_pdf:
  Producer/pdfproducer 1342 \cs_new_protected:Npn \_pdfmeta_xmp_build_pdf:
    PDFversion         1343 {

```

At first the producer. If not given manually we build it from the exec string plus the version number

```

1344   \_pdfmeta_xmp_add_packet_line_default:nnee
1345     {pdf}{Producer}
1346     {\c_sys_engine_exec_str-\c_sys_engine_version_str}
1347     {\GetDocumentProperties{hyperref/pdfproducer}}

```

Now the PDF version

```

1348   \_pdfmeta_xmp_add_packet_line:nne{pdf}{PDFVersion}{\pdf_version:}
1349 }

```

*(End of definition for \\_pdfmeta\_xmp\_build\_pdf:, Producer/pdfproducer, and PDFversion.)*

## 4.8.2 xmp

This builds the data with the (prefix “xmp”).

```
\_pdfmeta_xmp_build_xmp:
  CreatorTool/pdfcreator 1350 \cs_new_protected:Npn \_pdfmeta_xmp_build_xmp:
    BaseUrl/baseurl 1351 {
```

The creator

```
1352 \_pdfmeta_xmp_add_packet_line_default:nnee
1353   {xmp}{CreatorTool}
1354   {LaTeX}
1355   { \GetDocumentProperties{hyperref/pdfcreator} }
```

The baseurl

```
1356 \_pdfmeta_xmp_add_packet_line_default:nnee
1357   {xmp}{BaseUrl}{}
1358   { \GetDocumentProperties{hyperref/baseurl} }
```

CreationDate

```
1359 \_pdfmeta_xmp_date_get:nNN
1360   {document/creationdate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1361 \_pdfmeta_xmp_add_packet_line:nne{xmp}{CreateDate}{\_pdfmeta_xmp_print_date:N\l__pdfme
1362 \pdfmanagement_add:nne{Info}{CreateDate}{(\l__pdfmeta_tmpa_tl)}
```

ModifyDate

```
1363 \_pdfmeta_xmp_date_get:nNN
1364   {document/moddate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1365 \_pdfmeta_xmp_add_packet_line:nne{xmp}{ModifyDate}{\_pdfmeta_xmp_print_date:N\l__pdfme
1366 \pdfmanagement_add:nne{Info}{ModDate}{(\l__pdfmeta_tmpa_tl)}
```

MetadataDate

```
1367 \_pdfmeta_xmp_date_get:nNN
1368   {hyperref/pdfmetadate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1369 \_pdfmeta_xmp_add_packet_line:nne{xmp}{MetadataDate}{\_pdfmeta_xmp_print_date:N\l__pdf
1370 }
```

*(End of definition for \\_pdfmeta\_xmp\_build\_xmp:, CreatorTool/pdfcreator, and BaseUrl/baseurl.)*

## 4.8.3 Standards

The metadata for standards are taken from the pdfstandard key of \DocumentMetadata. The values for A-standards are taken from the property, X and UA are currently taken from the document container, this should be changed when merging of standards are possible.

```
\_pdfmeta_xmp_build_standards:
1371 \cs_new_protected:Npn \_pdfmeta_xmp_build_standards:
1372   {
1373     \_pdfmeta_xmp_add_packet_line:nne {pdfaid}{part}{\_pdfmeta_standard_item:n{level}}
1374     \_pdfmeta_xmp_add_packet_line:nne
1375       {pdfaid}{conformance}{\_pdfmeta_standard_item:n{conformance}}
1376     \int_compare:nNnTF {0\_pdfmeta_standard_item:n{level}}<{4}
1377       {\_pdfmeta_xmp_add_packet_line:nne {pdfaid}{year} {\_pdfmeta_standard_item:n{year}}}
1378       {\_pdfmeta_xmp_add_packet_line:nne {pdfaid}{rev} {\_pdfmeta_standard_item:n{year}}}
1379     \_pdfmeta_xmp_add_packet_line:nne
```

```

1380     {pdfxid}{GTS_PDFXVersion}{\GetDocumentProperties{document/pdfstandard-X}}
1381 \pdfmanagement_get_documentproperties:nNT {document/pdfstandard-UA}\l__pdfmeta_tmpa_tl
1382 {
1383     \__pdfmeta_xmp_add_packet_line:nne
1384     {pdfuaid}{part}{\exp_last_unbraced:No\use_i:nn \l__pdfmeta_tmpa_tl}
1385     \__pdfmeta_xmp_add_packet_line:nne
1386     {pdfuaid}{rev}{\exp_last_unbraced:No\use_ii:nn \l__pdfmeta_tmpa_tl}
1387 }
1388 }

```

(End of definition for \\_\_pdfmeta\_xmp\_build\_standards:.)

## 4.9 Declarations

See <https://pdfa.org/wp-content/uploads/2019/09/PDF-Declarations.pdf>

\g\_\_pdfmeta\_xmp\_pdfd\_data\_prop This holds the data for declarations.

```

1389 \prop_new:N \g__pdfmeta_xmp_pdfd_data_prop

```

(End of definition for \g\_\_pdfmeta\_xmp\_pdfd\_data\_prop.)

the main building command used in the xmp generation

\\_\_pdfmeta\_xmp\_build\_pdfd:

```

1390 \cs_new_protected:Npn \__pdfmeta_xmp_build_pdfd:
1391 {
1392     \prop_if_empty:NF \g__pdfmeta_xmp_pdfd_data_prop
1393     {
1394         \__pdfmeta_xmp_add_packet_open:nn{pdfd}{declarations}
1395         \__pdfmeta_xmp_add_packet_open:nn{rdf}{Bag}
1396         \prop_map_inline:Nn \g__pdfmeta_xmp_pdfd_data_prop
1397         {
1398             \__pdfmeta_xmp_build_pdfd_claim:nn{##1}{##2}
1399         }
1400         \__pdfmeta_xmp_add_packet_close:nn{rdf}{Bag}
1401         \__pdfmeta_xmp_add_packet_close:nn{pdfd}{declarations}
1402     }
1403 }

```

(End of definition for \\_\_pdfmeta\_xmp\_build\_pdfd:.)

\\_\_pdfmeta\_xmp\_build\_pdfd\_claim:nn This build the xml for one claim. If there is no claimData only the conformsTo is output.

```

1404 \cs_new_protected:Npn \__pdfmeta_xmp_build_pdfd_claim:nn #1#2
1405 {
1406     \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1407     \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{conformsTo}{#1}
1408     \tl_if_empty:NF {#2}
1409     {
1410         \__pdfmeta_xmp_add_packet_open:nn{pdfd}{claimData}
1411         \__pdfmeta_xmp_add_packet_open:nn{rdf}{Bag}
1412         #2
1413         \__pdfmeta_xmp_add_packet_close:nn{rdf}{Bag}
1414         \__pdfmeta_xmp_add_packet_close:nn{pdfd}{claimData}
1415     }
1416     \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1417 }

```

(End of definition for \\_\_pdfmeta\_xmp\_build\_pdfd\_claim:nn.)

## 4.10 Photoshop

\\_pdfmeta\_xmp\_build\_photoshop:

```
1418 \cs_new_protected:Npn \_pdfmeta_xmp_build_photoshop:
1419 {
pdfauthor/photshop:AuthorsPosition
1420   \_pdfmeta_xmp_add_packet_line:nne{photoshop}{AuthorsPosition}
1421   { \GetDocumentProperties{hyperref/pdfauthor} }
pdfcaptionwriter/photshop:CaptionWriter
1422   \_pdfmeta_xmp_add_packet_line:nne{photoshop}{CaptionWriter}
1423   { \GetDocumentProperties{hyperref/pdfcaptionwriter} }
1424 }
```

*(End of definition for \\_pdfmeta\_xmp\_build\_photoshop:.)*

## 4.11 XMP Media Management

\\_pdfmeta\_xmp\_build\_xmpMM:

```
1425 \cs_new_protected:Npn \_pdfmeta_xmp_build_xmpMM:
1426 {
pdfdocumentid / xmpMM:DocumentID
1427   \str_set:Ne\l__pdfmeta_tmpa_str {\GetDocumentProperties{hyperref/pdfdocumentid}}
1428   \str_if_empty:NT \l__pdfmeta_tmpa_str
1429   {
1430     \_pdfmeta_xmp_create_uuid:nN
1431     {\jobname\GetDocumentProperties{hyperref/pdfdocumentid}}
1432     \l__pdfmeta_tmpa_str
1433   }
1434   \_pdfmeta_xmp_add_packet_line:nnV{xmpMM}{DocumentID}
1435   \l__pdfmeta_tmpa_str
pdfinstanceid / xmpMM:InstanceID
1436   \str_set:Ne\l__pdfmeta_tmpa_str {\GetDocumentProperties{hyperref/pdfinstanceid}}
1437   \str_if_empty:NT \l__pdfmeta_tmpa_str
1438   {
1439     \_pdfmeta_xmp_create_uuid:nN
1440     {\jobname\l__pdfmeta_xmp_currentdate_tl}
1441     \l__pdfmeta_tmpa_str
1442   }
1443   \_pdfmeta_xmp_add_packet_line:nnV{xmpMM}{InstanceID}
1444   \l__pdfmeta_tmpa_str
pdfversionid/xmpMM:VersionID
1445   \_pdfmeta_xmp_add_packet_line:nne{xmpMM}{VersionID}
1446   { \GetDocumentProperties{hyperref/pdfversionid} }
pdfrendition/xmpMM:RenditionClass
1447   \_pdfmeta_xmp_add_packet_line:nne{xmpMM}{RenditionClass}
1448   { \GetDocumentProperties{hyperref/pdfrendition} }
1449 }
```

*(End of definition for \\_pdfmeta\_xmp\_build\_xmpMM:.)*

## 4.12 Rest of dublin Core data

```

__pdfmeta_xmp_build_dc:
  dc:creator/pdfauthor 1450 \cs_new_protected:Npn __pdfmeta_xmp_build_dc:
  dc:subject/pdfkeywords 1451 {
    dc:type/pdfdtype pdfauthor/dc:creator
    dc:publisher/pdfpublisher 1452 \__pdfmeta_xmp_add_packet_list:nne {dc}{creator}{Seq}
    dc:description/pdfsubject 1453 { \GetDocumentProperties{hyperref/pdfauthor} }
    dc:language/lang/pdflang 1454 \int_compare:nNnT {0\pdfmeta_standard_item:n{level}}={1}
dc:identifier/pdfidentifier 1455 { \pdfmanagement_remove:nn{Info}{Author} }
photoshop:AuthorsPosition/pdfauthortitle pdftitle/dc:title. This is rather complex as we want to support a list with different
photoshop:CaptionWriter/pdfcaptionwriter languages.
  1456 \__pdfmeta_xmp_add_packet_list:nne {dc}{title}{Alt}
  1457 { \GetDocumentProperties{hyperref/pdftitle} }
pdfkeywords/dc:subject
  1458 \__pdfmeta_xmp_add_packet_list:nne {dc}{subject}{Bag}
  1459 { \GetDocumentProperties{hyperref/pdfkeywords} }
  1460 \int_compare:nNnT {0\pdfmeta_standard_item:n{level}}={1}
  1461 { \pdfmanagement_remove:nn{Info}{Keywords} }
pdfdtype/dc:type
  1462 \pdfmanagement_get_documentproperties:nNTF { hyperref/pdfdtype } \l__pdfmeta_tmpa_tl
  1463 {
  1464 \__pdfmeta_xmp_add_packet_list_simple:nnnV {dc}{type}{Bag}\l__pdfmeta_tmpa_tl
  1465 }
  1466 {
  1467 \__pdfmeta_xmp_add_packet_list_simple:nnnn {dc}{type}{Bag}{Text}
  1468 }
pdfpublisher/dc:publisher
  1469 \__pdfmeta_xmp_add_packet_list:nne {dc}{publisher}{Bag}
  1470 { \GetDocumentProperties{hyperref/pdfpublisher} }
pdfsubject/dc:description
  1471 \__pdfmeta_xmp_add_packet_list:nne
  1472 {dc}{description}{Alt}
  1473 { \GetDocumentProperties{hyperref/pdfsubject} }
lang/pdflang/dc:language
  1474 \__pdfmeta_xmp_add_packet_list_simple:nnnV
  1475 {dc}{language}{Bag}\l__pdfmeta_xmp_doclang_tl
pdfidentifier/dc:identifier
  1476 \__pdfmeta_xmp_add_packet_line:nne{dc}{identifier}
  1477 { \GetDocumentProperties{hyperref/pdfidentifier} }
pdfdate/dc:date
  1478 \__pdfmeta_xmp_date_get:nNN {hyperref/pdfdate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
  1479 \__pdfmeta_xmp_add_packet_list_simple:nne
  1480 {dc}{date}{Seq}{\__pdfmeta_xmp_print_date:N\l__pdfmeta_tmpa_seq}
The file format
  1481 \__pdfmeta_xmp_add_packet_line:nnn{dc}{format}{application/pdf}

```

The source

```
1482 \__pdfmeta_xmp_add_packet_line_default:nnee
1483 {dc}{source}
1484 { \c_sys_jobname_str.tex }
1485 { \GetDocumentProperties{hyperref/pdfsource} }
1486 \__pdfmeta_xmp_add_packet_list:nne{dc}{rights}{Alt}
1487 {\GetDocumentProperties{hyperref/pdfcopyright}}
1488 }
```

(End of definition for \\_\_pdfmeta\_xmp\_build\_dc: and others.)

### 4.13 xmpRights

\\_\_pdfmeta\_xmp\_build\_xmpRights:

```
1489 \cs_new_protected:Npn \__pdfmeta_xmp_build_xmpRights:
1490 {
1491 \__pdfmeta_xmp_add_packet_line:nne
1492 {xmpRights}
1493 {WebStatement}
1494 {\GetDocumentProperties{hyperref/pdflicenseurl}}
1495 \__pdfmeta_xmp_add_packet_line:nne
1496 {xmpRights}
1497 {Marked}
1498 {
1499 \str_case:en {\GetDocumentProperties{document/copyright}}
1500 {
1501 {true}{True}
1502 {false}{False}
1503 }
1504 }
1505 }
```

(End of definition for \\_\_pdfmeta\_xmp\_build\_xmpRights:.)

### 4.14 IPTC

We want the block and also the resources only if they are actually used. So we pack them first in a local variable

\l\_\_pdfmeta\_xmp\_iptc\_data\_tl

```
1506 \tl_new:N\l__pdfmeta_xmp_iptc_data_tl
```

(End of definition for \l\_\_pdfmeta\_xmp\_iptc\_data\_tl.)

\\_\_pdfmeta\_xmp\_build\_iptc\_data:N

```
1507 \cs_new_protected:Npn \__pdfmeta_xmp_build_iptc_data:N #1
1508 {
1509 \tl_clear:N #1
1510 \__pdfmeta_xmp_incr_indent:\__pdfmeta_xmp_incr_indent:\__pdfmeta_xmp_incr_indent:\__pdf
1511 \__pdfmeta_xmp_add_packet_line:nneN
1512 {Iptc4xmpCore}{CiAdrExtadr}
1513 {\GetDocumentProperties{hyperref/pdfcontactaddress}}
1514 #1
```



```

1515 \_pdfmeta_xmp_add_packet_line:nneN
1516 {Iptc4xmpCore}{CiAdrCity}
1517 {\GetDocumentProperties{hyperref/pdfcontactcity}}
1518 #1
1519 \_pdfmeta_xmp_add_packet_line:nneN
1520 {Iptc4xmpCore}{CiAdrPcode}
1521 {\GetDocumentProperties{hyperref/pdfcontactpostcode}}
1522 #1
1523 \_pdfmeta_xmp_add_packet_line:nneN
1524 {Iptc4xmpCore}{CiAdrCtry}
1525 {\GetDocumentProperties{hyperref/pdfcontactcountry}}
1526 #1
1527 \_pdfmeta_xmp_add_packet_line:nneN
1528 {Iptc4xmpCore}{CiTelWork}
1529 {\GetDocumentProperties{hyperref/pdfcontactphone}}
1530 #1
1531 \_pdfmeta_xmp_add_packet_line:nneN
1532 {Iptc4xmpCore}{CiEmailWork}
1533 {\GetDocumentProperties{hyperref/pdfcontactemail}}
1534 #1
1535 \_pdfmeta_xmp_add_packet_line:nneN
1536 {Iptc4xmpCore}{CiUrlWork}
1537 {\GetDocumentProperties{hyperref/pdfcontacturl}}
1538 #1
1539 \_pdfmeta_xmp_decr_indent:\_pdfmeta_xmp_decr_indent:\_pdfmeta_xmp_decr_indent:\_pdf
1540 }

```

(End of definition for \\_pdfmeta\_xmp\_build\_iptc:N.)

\\_pdfmeta\_xmp\_build\_iptc:

```

1541 \cs_new_protected:Npn \_pdfmeta_xmp_build_iptc:
1542 {
1543   \tl_if_empty:NF\l__pdfmeta_xmp_iptc_data_tl
1544   {
1545     \_pdfmeta_xmp_add_packet_open_attr:nnn
1546     {Iptc4xmpCore}{CreatorContactInfo}{rdf:parseType="Resource"}
1547     \tl_gput_right:Ne\g__pdfmeta_xmp_packet_tl { \l__pdfmeta_xmp_iptc_data_tl }
1548     \_pdfmeta_xmp_add_packet_close:nn
1549     {Iptc4xmpCore}{CreatorContactInfo}
1550   }
1551 }

```

(End of definition for \\_pdfmeta\_xmp\_build\_iptc:.)

## 4.15 Prism

\\_pdfmeta\_xmp\_build\_prism:  
 complianceProfile  
 prism:subtitle/pdfsubtitle

```

1552 \cs_new_protected:Npn \_pdfmeta_xmp_build_prism:
1553 {

```

The compliance profile is a fix value taken from hyperxmp

```

1554 \_pdfmeta_xmp_add_packet_line:nnn
1555 {prism}{complianceProfile}
1556 {three}

```

the next two values can take an optional language argument. First subtitle

```
1557 \__pdfmeta_xmp_lang_get:eNN
1558 {\GetDocumentProperties{hyperref/pdfsubtitle}}
1559 \l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl
1560 \__pdfmeta_xmp_add_packet_line_attr:nneV
1561 {prism}{subtitle}
1562 {xml:lang="\l__pdfmeta_tmpa_tl"}
1563 \l__pdfmeta_tmpb_tl
```

Then publicationName

```
1564 \__pdfmeta_xmp_lang_get:eNN
1565 {\GetDocumentProperties{hyperref/pdfpublication}}
1566 \l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl
1567 \__pdfmeta_xmp_add_packet_line_attr:nneV
1568 {prism}{publicationName}
1569 {xml:lang="\l__pdfmeta_tmpa_tl"}
1570 \l__pdfmeta_tmpb_tl
```

Now the rest

```
1571 \__pdfmeta_xmp_add_packet_line:nne
1572 {prism}{bookEdition}
1573 {\GetDocumentProperties{hyperref/pdfbookedition}}
1574 \__pdfmeta_xmp_add_packet_line:nne
1575 {prism}{aggregationType}
1576 {\GetDocumentProperties{hyperref/pdfpubtype}}
1577 \__pdfmeta_xmp_add_packet_line:nne
1578 {prism}{volume}
1579 {\GetDocumentProperties{hyperref/pdfvolumenum}}
1580 \__pdfmeta_xmp_add_packet_line:nne
1581 {prism}{number}
1582 {\GetDocumentProperties{hyperref/pdfissuenum}}
1583 \__pdfmeta_xmp_add_packet_line:nne
1584 {prism}{pageRange}
1585 {\GetDocumentProperties{hyperref/pdfpagerange}}
1586 \__pdfmeta_xmp_add_packet_line:nne
1587 {prism}{issn}
1588 {\GetDocumentProperties{hyperref/pdfissn}}
1589 \__pdfmeta_xmp_add_packet_line:nne
1590 {prism}{eIssn}
1591 {\GetDocumentProperties{hyperref/pdfeissn}}
1592 \__pdfmeta_xmp_add_packet_line:nne
1593 {prism}{doi}
1594 {\GetDocumentProperties{hyperref/pdfdoi}}
1595 \__pdfmeta_xmp_add_packet_line:nne
1596 {prism}{url}
1597 {\GetDocumentProperties{hyperref/pdfurl}}
```

The page count is take from the previous run or from pdfnumpages.

```
1598 \tl_set:Nc \l__pdfmeta_tmpa_tl { \GetDocumentProperties{hyperref/pdfnumpages} }
1599 \__pdfmeta_xmp_add_packet_line:nne
1600 {prism}{pageCount}
1601 {\tl_if_blank:VTF \l__pdfmeta_tmpa_tl {\PreviousTotalPages}{\l__pdfmeta_tmpa_tl}}
1602 }
```

*(End of definition for \\_\_pdfmeta\_xmp\_build\_prism:, complianceProfile, and prism:subtitle/pdfsubtitle.)*

### 4.15.1 User additions

`\g_pdfmeta_xmp_user_packet_str`

```
1603 \tl_new:N \g__pdfmeta_xmp_user_packet_tl
```

*(End of definition for \g\_\_pdfmeta\_xmp\_user\_packet\_str.)*

`\__pdfmeta_xmp_build_user:`

```
1604 \cs_new_protected:Npn \__pdfmeta_xmp_build_user:
1605 {
1606   \int_zero:N \l__pdfmeta_xmp_indent_int
1607   \g__pdfmeta_xmp_user_packet_tl
1608   \int_set:Nn \l__pdfmeta_xmp_indent_int {3}
1609 }
```

*(End of definition for \\_\_pdfmeta\_xmp\_build\_user:.)*

## 4.16 Activating the metadata

We don't try to get the byte count. So we can put everything in the `shipout/lastpage` hook

```
1610 \hook_new:n { pdfmeta/xmp }
1611 \AddToHook{shipout/lastpage}
1612 {
1613   \bool_if:NT\g__pdfmeta_xmp_bool
1614   {
1615     \str_if_exist:NTF\c_sys_timestamp_str
1616     {
1617       \tl_set_eq:NN \l__pdfmeta_xmp_currentdate_tl \c_sys_timestamp_str
1618     }
1619     {
1620       \file_get_timestamp:nN{\jobname.log}\l__pdfmeta_xmp_currentdate_tl
1621     }
1622     \__pdfmeta_xmp_date_split:VN\l__pdfmeta_xmp_currentdate_tl\l__pdfmeta_xmp_currentdate_tl
1623     \hook_use:n { pdfmeta/xmp }
1624     \__pdfmeta_xmp_build_packet:
1625     \exp_args:No
1626     \__pdf_backend_metadata_stream:n {\g__pdfmeta_xmp_packet_tl}
1627     \pdfmanagement_add:nne {Catalog} {Metadata}{\pdf_object_ref_last:}
1628     \bool_if:NT \g__pdfmeta_xmp_export_bool
1629     {
1630       \iow_open:Nn\g_tmpa_iow{\g__pdfmeta_xmp_export_str.xmpi}
1631       \exp_args:NNo\iow_now:Nn\g_tmpa_iow{\g__pdfmeta_xmp_packet_tl}
1632       \iow_close:N\g_tmpa_iow
1633     }
1634   }
1635 }
```

## 4.17 User commands

`\pdfmeta_xmp_add:n`

```
1636 \cs_new_protected:Npn \pdfmeta_xmp_add:n #1
1637 {
1638   \tl_gput_right:Nn \g__pdfmeta_xmp_user_packet_tl
```

```

1639     {
1640     \__pdfmeta_xmp_add_packet_chunk:n { #1 }
1641     }
1642 }

```

(End of definition for \pdfmeta\_xmp\_add:n. This function is documented on page 9.)

**\pdfmeta\_xmp\_xmlns\_new:nn**

```

1643 \cs_new_protected:Npn \pdfmeta_xmp_xmlns_new:nn #1 #2
1644 {
1645   \prop_if_in:NnTF \g__pdfmeta_xmp_xmlns_prop {#1}
1646   {\msg_warning:nnnn{pdfmeta}{xmp-defined}{xmlns-namespace}{#1}}
1647   {\__pdfmeta_xmp_xmlns_new:nn {#1}{#2}}
1648 }

```

(End of definition for \pdfmeta\_xmp\_xmlns\_new:nn. This function is documented on page 9.)

**\pdfmeta\_xmp\_schema\_new:nnn**

```

1649 \cs_set_eq:NN \pdfmeta_xmp_schema_new:nnn \__pdfmeta_xmp_schema_new:nnn

```

(End of definition for \pdfmeta\_xmp\_schema\_new:nnn. This function is documented on page 10.)

**\pdfmeta\_xmp\_property\_new:nnnn**

```

1650 \cs_set_eq:NN \pdfmeta_xmp_property_new:nnnn \__pdfmeta_xmp_property_new:nnnn

```

(End of definition for \pdfmeta\_xmp\_property\_new:nnnn. This function is documented on page 10.)

**\pdfmeta\_xmp\_add\_declaration:n**

**\pdfmeta\_xmp\_add\_declaration:e**

```

1651 \cs_new_protected:Npn \pdfmeta_xmp_add_declaration:n #1 %conformsTo uri
1652 {
1653   \__pdfmeta_xmp_schema_enable_pdfd:
1654   \prop_gput:Nnn\g__pdfmeta_xmp_pdfd_data_prop{#1}{}
1655 }
1656 \cs_generate_variant:Nn \pdfmeta_xmp_add_declaration:n {e}

```

(End of definition for \pdfmeta\_xmp\_add\_declaration:n. This function is documented on page 9.)

**\pdfmeta\_xmp\_add\_declaration:nnnn**

**\pdfmeta\_xmp\_add\_declaration:ennnn**

```

1657 \cs_new_protected:Npn \pdfmeta_xmp_add_declaration:nnnn #1#2#3#4#5
1658 %#1=conformsTo uri, #2 claimBy, #3 claimDate #4 claimCredentials #4 claimReport
1659 {
1660   \__pdfmeta_xmp_schema_enable_pdfd:
1661   \tl_set:Nn \l__pdfmeta_tmpa_tl
1662   {
1663     \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1664     \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimBy}{#2}
1665     \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimDate}{#3}
1666     \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimCredentials}{#4}
1667     \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimReport}{#5}
1668     \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1669   }
1670   \prop_get:NnNT \g__pdfmeta_xmp_pdfd_data_prop {#1}\l__pdfmeta_tmpb_tl
1671   {
1672     \tl_concat:NNN \l__pdfmeta_tmpa_tl \l__pdfmeta_tmpa_tl \l__pdfmeta_tmpb_tl
1673   }

```

```

1674 \prop_gput:Nno\g__pdfmeta_xmp_pdfd_data_prop{#1}
1675   {
1676     \l__pdfmeta_tmpa_tl
1677   }
1678 }
1679 \cs_generate_variant:Nn\pdfmeta_xmp_add_declaration:nnnnn {e,eee}

```

(End of definition for `\pdfmeta_xmp_add_declaration:nnnnn`. This function is documented on page 9.)

## 4.18 Default declarations

The two declarations will be required quite often with ua-2, so we provide some interface.

```

\__pdfmeta_xmp_wtpdf_reuse_declaration:
pdfmeta_xmp_wtpdf_accessibility_declaration:
1680 \cs_new:Npn \__pdfmeta_xmp_iso_today:
1681   {
1682     \int_use:N\c_sys_year_int-
1683     \int_compare:nNnT {\c_sys_month_int} < {10}{0} \int_use:N\c_sys_month_int -
1684     \int_compare:nNnT {\c_sys_day_int} < {10}{0} \int_use:N\c_sys_day_int
1685   }
1686 \cs_new_protected:Npn \__pdfmeta_xmp_wtpdf_reuse_declaration:
1687   {
1688     \pdfmeta_xmp_add_declaration:eeenn
1689     {http://pdfa.org/declarations/wtpdf\c_hash_str reuse1.0}
1690     {LaTeX~Project}
1691     {\__pdfmeta_xmp_iso_today:}{}{}
1692   }
1693 \cs_new_protected:Npn \__pdfmeta_xmp_wtpdf_accessibility_declaration:
1694   {
1695     \pdfmeta_xmp_add_declaration:ennnn
1696     {http://pdfa.org/declarations/wtpdf\c_hash_str accessibility1.0}
1697     {LaTeX~Project}
1698     {\__pdfmeta_xmp_iso_today:}{}{}
1699   }

```

(End of definition for `\__pdfmeta_xmp_wtpdf_reuse_declaration:` and `\__pdfmeta_xmp_wtpdf_accessibility_declaration:`.)

```
1700 </package>
```

# Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	
<code>\&amp;</code> .....	797
<code>\'</code> .....	732
<code>\+</code> .....	732
<code>\-</code> .....	732, 813
<code>\[</code> .....	813
<code>\</code> .....	12
<code>\]</code> .....	813
<b>A</b>	
<code>\A</code> .....	813
<code>\AddToDocumentProperties</code> ...	575, 580, 585, 590, 595, 600, 605, 610, 615, 629
<code>\AddToHook</code>	329, 354, 510, 616, 630, 632, 1611

<b>B</b>		<b>E</b>	
BaseUrl/baseurl	1350	exp commands:	
bitset commands:		\exp_args:NNe	530, 535, 541
\bitset_set_false:Nn	98, 99, 100	\exp_args:Nne	347
\bitset_set_true:Nn	97	\exp_args:Nnne	46
\bitset_to_arabic:N	101, 102, 103, 104, 105	\exp_args:NNo	446, 1631
bool commands:		\exp_args:No	1625
\bool_gset_false:N	646, 681	\exp_args:NV	548, 551
\bool_gset_true:N	565, 645, 676, 685	\exp_last_unbraced:No	1384, 1386
\bool_if:NTF	335, 1613, 1628	\exp_not:n	834, 842, 1003
\bool_lazy_or:nnTF	358, 693	<b>F</b>	
\bool_new:N	564, 668	file commands:	
<b>C</b>		\file_get_timestamp:nN	1620
char commands:		<b>G</b>	
\char_generate:nn	698, 703, 704, 705	\GetDocumentProperties	771, 951, 952, 1347, 1355, 1358, 1380, 1421, 1423, 1427, 1431, 1436, 1446, 1448, 1453, 1457, 1459, 1470, 1473, 1477, 1485, 1487, 1494, 1499, 1513, 1517, 1521, 1525, 1529, 1533, 1537, 1558, 1565, 1573, 1576, 1579, 1582, 1585, 1588, 1591, 1594, 1597, 1598
clist commands:		group commands:	
\clist_if_empty:nTF	910, 927	\group_begin:	339, 488, 796
\clist_map_inline:nn	493, 914, 931	\group_end:	348, 507, 805
complianceProfile	1552	<b>H</b>	
CreatorTool/pdfcreator	1350	hook commands:	
cs commands:		\hook_gput_code:nnn	107, 566
\cs_generate_variant:Nn	737, 809, 828, 837, 845, 851, 858, 873, 883, 893, 906, 923, 948, 1656, 1679	\hook_new:n	1610
\cs_gset_eq:NN	1340	\hook_use:n	1623
\cs_if_exist:NTF	44	<b>I</b>	
\cs_if_exist_use:N	1051	int commands:	
\cs_new:Npn	22, 697, 701, 709, 715, 738, 1680	\int_compare:nNnTF	1376, 1454, 1460, 1683, 1684
\cs_new_protected:Npn	26, 61, 69, 77, 83, 89, 95, 471, 486, 562, 721, 726, 733, 768, 781, 793, 814, 830, 838, 846, 852, 859, 864, 874, 884, 894, 907, 924, 949, 998, 1029, 1057, 1080, 1131, 1249, 1284, 1297, 1342, 1350, 1371, 1390, 1404, 1418, 1425, 1450, 1489, 1507, 1541, 1552, 1604, 1636, 1643, 1651, 1657, 1686, 1693	\int_decr:N	728
\cs_set_eq:NN	659, 665, 954, 1649, 1650	\int_if_zero:nTF	374
<b>D</b>		\int_if_zero_p:n	359, 360
\d	732	\int_incr:N	723
dc commands:		\int_new:N	708
dc:description/pdfsubject	1450	\int_set:Nn	991, 1608
dc:identififier/pdfidentififier	1450	\int_use:N	1682, 1683, 1684
dc:language/lang/pdflang	1450	\int_zero:N	993, 1606
dc:Nreator/pdfauthor	1450	iow commands:	
dc:publisher/pdfpublisher	1450	\iow_close:N	1632
dc:subject/pdfkeywords	1450	\iow_newline:	366, 385, 392, 711, 717
dc:type/pdftype	1450	\iow_now:Nn	1631
\DocumentMetadata	2, 4	\iow_open:Nn	1630
		\g_tmpa_iow	1630, 1631, 1632
		iptc_(schema)	1239
		<b>J</b>	
		\jobname	1431, 1440, 1620

<b>K</b>	
kernel internal commands:	
\g__kernel_pdfmanagement_end_-	
run_code_tl	333
keys commands:	
\keys_define:nn	401, 408, 571, 653, 671
\l_keys_key_str	448
\keys_set:nn	405, 650
<b>M</b>	
msg commands:	
\msg_new:nnn	7, 8, 10, 690, 691, 692
\msg_warning:nnn	364, 383, 390, 525, 558
\msg_warning:nnnn	1034, 1077, 1646
\msg_warning:nnnnn	119, 129, 621, 638
<b>P</b>	
pdf commands:	
\pdf_object_if_exist:nTF	473
\pdf_object_new:n	475
\pdf_object_ref:n	492
\pdf_object_ref_last:	506, 1627
\pdf_object_unnamed_write:nn	505
\pdf_object_write:nnn	476
\pdf_string_from_unicode:nnN	500
\pdf_version:	
4, 118, 120, 128, 130, 623, 640, 1348	
\pdf_version_compare:NnTF	
63, 71, 619, 636	
pdf internal commands:	
\__pdf_backend_metadata_stream:n	1626
\__pdf_backend_Names_gpush:nn	347
\__pdf_backend_omit_charset:n	114
\__pdf_backend_omit_cidset:n	116
\__pdf_backend_omit_info:n	112
\__pdf_backend_set_regression_-	
data:	563
pdfaid~(schema)	1099
pdfannot commands:	
\pdfannot_dict_put:nnn	
101, 102, 103, 104, 105	
\l_pdfannot_F_bitset	97,
98, 99, 100, 101, 102, 103, 104, 105	
pdfdict commands:	
\pdfdict_if_empty:nTF	337
\pdfdict_new:n	452
\pdfdict_put:nnn	
340, 341, 453, 489, 490, 501	
\pdfdict_use:n	505
pdffile commands:	
\g_pdffile_embed_nonpdfa_int	360, 374
\g_pdffile_embed_pdfa_int	359
\pdffile_embed_stream:nnN	342
pdfmanagement commands:	
\pdfmanagement_add:nnn	
506, 568, 569, 1362, 1366, 1627	
\pdfmanagement_get_documentproperties:nNTF	
1381, 1462	
\pdfmanagement_remove:nn	1455, 1461
pdfmanagement internal commands:	
\g__pdfmanagement_active_bool	335
pdfmeta commands:	
\pdfmeta_set_regression_data:	5, 562
\pdfmeta_standard_get:nN	2, 26, 26
\pdfmeta_standard_item:n	
2, 22, 22, 122,	
124, 132, 134, 533, 538, 544, 1373,	
1375, 1376, 1377, 1378, 1454, 1460	
\pdfmeta_standard_verify:n	2, 30
\pdfmeta_standard_verify:nn	2, 40
\pdfmeta_standard_verify:nnN	2
\pdfmeta_standard_verify:nnTF	
2, 40, 117, 127	
\pdfmeta_standard_verify:nTF	
2, 30,	
109, 111, 113, 115, 331, 356, 372, 512	
\pdfmeta_standard_verify_p:n	2, 30
\pdfmeta_xmp_add:n	9, 1636, 1636
\pdfmeta_xmp_add_declaration:n	
9, 1651, 1651, 1656	
\pdfmeta_xmp_add_declaration:nnnnn	
9, 1657, 1657, 1679, 1688, 1695	
\pdfmeta_xmp_property_new:nnnnn	
10, 1650, 1650	
\pdfmeta_xmp_schema_new:nnn	
10, 1649, 1649	
\pdfmeta_xmp_xmlns_new:nn	
9, 1643, 1643	
pdfmeta internal commands:	
\__pdfmeta_embed_colorprofile:n	
471, 471, 518, 548	
\g__pdfmeta_outputintents_prop	
400, 414, 422,	
430, 438, 447, 514, 532, 537, 543, 549	
\g__pdfmeta_standard_pdf/A-1B_-	
prop	138
\g__pdfmeta_standard_pdf/A-2A_-	
prop	138
\g__pdfmeta_standard_pdf/A-2B_-	
prop	138
\g__pdfmeta_standard_pdf/A-2U_-	
prop	138
\g__pdfmeta_standard_pdf/A-3A_-	
prop	138
\g__pdfmeta_standard_pdf/A-3B_-	
prop	138

\g\_pdfmeta\_standard\_pdf/A-3U\_-prop ..... 138  
 \g\_pdfmeta\_standard\_pdf/A-4\_-prop ..... 138  
 \g\_pdfmeta\_standard\_pdf/A-4F\_-prop ..... 138  
 \g\_pdfmeta\_standard\_prop ... 21, 24, 28, 32, 42, 50, 362, 376, 618, 631  
 \\_pdfmeta\_standard\_verify\_-handler\_annot\_action\_A:nn . 83, 83  
 \\_pdfmeta\_standard\_verify\_-handler\_max\_pdf\_version:nn 68, 69  
 \\_pdfmeta\_standard\_verify\_-handler\_min\_pdf\_version:nn 60, 61  
 \\_pdfmeta\_standard\_verify\_-handler\_named\_actions:nn .. 76, 77  
 \\_pdfmeta\_standard\_verify\_-handler\_outputintent\_subtype:nn ..... 89, 89  
 \l\_pdfmeta\_tmpa\_seq ..... 15, 817, 818, 824, 825, 1360, 1361, 1364, 1365, 1368, 1369, 1478, 1480  
 \g\_pdfmeta\_tmpa\_str ..... 18, 800, 801, 802, 803, 804, 806  
 \l\_pdfmeta\_tmpa\_str ..... 15, 500, 502, 869, 870, 879, 880, 889, 890, 1427, 1428, 1432, 1435, 1436, 1437, 1441, 1444  
 \l\_pdfmeta\_tmpa\_tl ..... 15, 346, 347, 362, 367, 376, 378, 393, 498, 500, 799, 800, 899, 902, 904, 933, 934, 941, 1062, 1360, 1362, 1364, 1366, 1368, 1381, 1384, 1386, 1462, 1464, 1478, 1559, 1562, 1566, 1569, 1598, 1601, 1661, 1672, 1676  
 \l\_pdfmeta\_tmpb\_seq ..... 15  
 \l\_pdfmeta\_tmpb\_tl ..... 15, 545, 546, 548, 553, 558, 933, 937, 941, 1559, 1563, 1566, 1570, 1670, 1672  
 \\_pdfmeta\_verify\_pdfa\_annot\_-flags: ..... 95, 110  
 \\_pdfmeta\_write\_outputintent:nn ..... 471, 486, 520, 552  
 \\_pdfmeta\_xmp\_add\_packet\_-chunk:n .... 830, 830, 837, 848, 855, 862, 870, 890, 960, 992, 994, 1640  
 \\_pdfmeta\_xmp\_add\_packet\_-chunk:nN ..... 838, 838, 845, 880  
 \\_pdfmeta\_xmp\_add\_packet\_-close:nn 859, 859, 919, 920, 944, 945, 973, 974, 988, 989, 990, 1049, 1050, 1052, 1072, 1087, 1278, 1279, 1280, 1281, 1282, 1318, 1319, 1320, 1334, 1335, 1336, 1337, 1338, 1400, 1401, 1413, 1414, 1416, 1548, 1668  
 \\_pdfmeta\_xmp\_add\_packet\_-field:nnn 1080, 1080, 1262, 1264, 1266, 1268, 1270, 1272, 1274, 1276, 1310, 1312, 1314, 1316, 1330, 1332  
 \\_pdfmeta\_xmp\_add\_packet\_-line:nnn ... 864, 864, 873, 904, 916, 1043, 1044, 1045, 1068, 1069, 1070, 1071, 1084, 1085, 1086, 1254, 1255, 1257, 1258, 1302, 1303, 1305, 1306, 1322, 1323, 1325, 1326, 1348, 1361, 1365, 1369, 1373, 1374, 1377, 1378, 1379, 1383, 1385, 1407, 1420, 1422, 1434, 1443, 1445, 1447, 1476, 1481, 1491, 1495, 1554, 1571, 1574, 1577, 1580, 1583, 1586, 1589, 1592, 1595, 1599, 1664, 1665, 1666, 1667  
 \\_pdfmeta\_xmp\_add\_packet\_-line:nnnN . 874, 874, 883, 1511, 1515, 1519, 1523, 1527, 1531, 1535  
 \\_pdfmeta\_xmp\_add\_packet\_line\_-attr:nnnn ..... 884, 884, 893, 936, 940, 1560, 1567  
 \\_pdfmeta\_xmp\_add\_packet\_line\_-default:nnnn ..... 894, 894, 906, 1344, 1352, 1356, 1482  
 \\_pdfmeta\_xmp\_add\_packet\_-list:nnnn ..... 924, 948, 1452, 1456, 1458, 1469, 1471, 1486  
 \\_pdfmeta\_xmp\_add\_packet\_list\_-simple:nnnn ..... 907, 923, 1464, 1467, 1474, 1479  
 \\_pdfmeta\_xmp\_add\_packet\_-open:nn ..... 846, 846, 851, 912, 913, 929, 930, 962, 963, 967, 968, 1046, 1047, 1067, 1251, 1252, 1260, 1261, 1299, 1300, 1308, 1309, 1328, 1329, 1394, 1395, 1410, 1411  
 \\_pdfmeta\_xmp\_add\_packet\_open\_-attr:nnn ..... 852, 852, 858, 965, 1042, 1083, 1253, 1301, 1321, 1406, 1545, 1663  
 \\_pdfmeta\_xmp\_add\_pdfxid: . 576, 581, 586, 591, 596, 601, 606, 611, 1131  
 \g\_pdfmeta\_xmp\_bool ..... 564, 645, 646, 1613  
 \\_pdfmeta\_xmp\_build\_dc: ..... 980, 1450, 1450  
 \\_pdfmeta\_xmp\_build\_iptc: ..... 985, 1541, 1541  
 \\_pdfmeta\_xmp\_build\_iptc\_data:N ..... 955, 1507, 1507



\_pdfmeta_xmp_build_packet: . . .	\_pdfmeta_xmp_lang_get:nNN . . . . .
. . . . . 949, 949, 1624	. . . . . 814, 828, 933, 1557, 1564
\_pdfmeta_xmp_build_pdf: . . . . .	\l_pdfmeta_xmp_lang_regex . 812, 817
. . . . . 976, 1342, 1342	\l_pdfmeta_xmp_metalang_tl . . . . .
\_pdfmeta_xmp_build_pdfd: . . . . .	. . . . . 810, 820, 934, 952, 953, 954
. . . . . 979, 1390, 1390	\g_pdfmeta_xmp_packet_tl . . . . .
\_pdfmeta_xmp_build_pdfd_-	. . . . . 829, 832, 1547, 1626, 1631
claim:nn . . . . . 1398, 1404, 1404	\g_pdfmeta_xmp_pdfd_data_prop . .
\_pdfmeta_xmp_build_photoshop: .	. . 1389, 1392, 1396, 1654, 1670, 1674
. . . . . 981, 1418, 1418	\_pdfmeta_xmp_print_date:N . . . . .
\_pdfmeta_xmp_build_prism: . . . . .	. . . . . 738, 738, 1361, 1365, 1369, 1480
. . . . . 984, 1552, 1552	\_pdfmeta_xmp_property_new:nnnnn
\_pdfmeta_xmp_build_standards: .	. . . . . 1056, 1057,
. . . . . 978, 1371, 1371	1093, 1103, 1109, 1119, 1125, 1138,
\_pdfmeta_xmp_build_user: . . . . .	1149, 1155, 1161, 1167, 1173, 1179,
. . . . . 986, 1604, 1604	1185, 1191, 1197, 1203, 1209, 1215,
\_pdfmeta_xmp_build_xmp: . . . . .	1221, 1227, 1233, 1243, 1291, 1650
. . . . . 982, 1350, 1350	\_pdfmeta_xmp_sanitize:nN . . . . .
\_pdfmeta_xmp_build_xmpMM: . . . . .	. . . . . 793, 793, 809, 869, 879, 889
. . . . . 983, 1425, 1425	\_pdfmeta_xmp_schema_enable_-
\_pdfmeta_xmp_build_xmpRights: .	pdfd: . . . . . 1284, 1340, 1653, 1660
. . . . . 977, 1489, 1489	\_pdfmeta_xmp_schema_new:nnn . . .
\_pdfmeta_xmp_create_uuid:nN . . .	. . . . . 1029, 1029, 1089, 1099,
. . . . . 781, 781, 1430, 1439	1115, 1134, 1145, 1239, 1287, 1649
\l_pdfmeta_xmp_currentdate_seq .	\g_pdfmeta_xmp_schema_property_-
. . . . . 766, 774, 1622	prop . . . . . 1056, 1062, 1064
\l_pdfmeta_xmp_currentdate_tl . .	\l_pdfmeta_xmp_schema_seq . . . . .
. . . . . 766, 775, 1440, 1617, 1620, 1622	. . . . . 958, 969, 1028, 1037
\_pdfmeta_xmp_date_get:nNN . . . . .	\g_pdfmeta_xmp_user_packet_str 1603
. . . . . 768, 768, 1359, 1363, 1367, 1478	\g_pdfmeta_xmp_user_packet_tl . .
\l_pdfmeta_xmp_date_regex . 730, 735	. . . . . 1603, 1607, 1638
\_pdfmeta_xmp_date_split:nN . . . . .	\_pdfmeta_xmp_wtpdf_accessibility_-
. . . . . 733, 733, 737, 778, 1622	declaration: . . . . .
\_pdfmeta_xmp_decr_indent: . . . . .	. . . . . 634, 662, 665, 1680, 1693
. . . . . 709, 726, 861, 1539	\_pdfmeta_xmp_wtpdf_reuse_-
\l_pdfmeta_xmp_doclang_tl . . . . .	declaration: . . . . .
. . . . . 810, 951, 954, 1475	. . . . . 635, 656, 659, 1680, 1686
\g_pdfmeta_xmp_export_bool . . . . .	\_pdfmeta_xmp_xmlns_new:nn . . . . .
. . . . . 668, 676, 681, 685, 1628	. . . . . 998, 998,
\g_pdfmeta_xmp_export_str . . . . .	1006, 1007, 1008, 1009, 1010, 1011,
. . . . . 669, 677, 686, 1630	1012, 1014, 1015, 1016, 1017, 1018,
\_pdfmeta_xmp_generate_bom: . . . . .	1019, 1020, 1021, 1022, 1023, 1024,
. . . . . 693, 697, 701, 961	1025, 1026, 1027, 1133, 1286, 1647
\_pdfmeta_xmp_incr_indent: . . . . .	\g_pdfmeta_xmp_xmlns_prop . . . . .
. . . . . 709, 721, 849, 856, 1510	. . . . . 996, 1000, 1645
\_pdfmeta_xmp_indent: . . . . .	\g_pdfmeta_xmp_xmlns_tl . . . . .
. . . . . 709, 709, 834, 842	. . . . . 966, 996, 1001
\_pdfmeta_xmp_indent:n 709, 715, 1003	pdfmetatmpa internal commands:
\l_pdfmeta_xmp_indent_int . 708,	\g_pdfmetatmpa_str . . . . . 15
712, 723, 728, 991, 993, 1606, 1608	pdfuaid~(schema) . . . . . 1115
\l_pdfmeta_xmp_iptc_data_tl . . . . .	PDFversion . . . . . 1342
. . . . . 955, 956, 1506, 1543, 1547	pdfxid~(schema) . . . . . 1131
\_pdfmeta_xmp_iso_today: . . . . .	
. . . . . 1680, 1691, 1698	



<code>\tl_if_blank:nTF</code> . . . . .	412, 420, 428, 436, 444, 740, 747, 750, 753, 758, 772, 867, 877, 887, 897, 953, 1601	<code>\tl_set:Nn</code> . . . . .	771, 799, 820, 821, 824, 825, 899, 902, 951, 952, 1598, 1661
<code>\tl_if_empty:NTF</code> . . . . .	956, 1543	<code>\tl_set_eq:NN</code> . . . . .	775, 1617
<code>\tl_if_empty:nTF</code> . . . . .	1408	<code>\tl_to_str:N</code> . . . . .	797, 800
<code>\tl_if_eq:nnTF</code> . . . . .	91, 934	<code>\tl_use:N</code> . . . . .	971, 1048
<code>\tl_if_exist:NTF</code> . . . . .	1032, 1060	<b>U</b>	
<code>\tl_if_in:nnTF</code> . . . . .	79, 85	use commands:	
<code>\tl_new:N</code> . . . . .	15, 16, 766, 810, 811, 829, 1038, 1039, 1506, 1603	<code>\use:N</code> . . . . .	47
<code>\tl_put_right:Nn</code> . . . . .	840	<code>\use_i:nn</code> . . . . .	1384
		<code>\use_ii:nn</code> . . . . .	1386