

The `l3pdfmeta` module

PDF standards

LaTeX PDF management bundle

The LaTeX Project*

Version 0.96v, released 2025-08-05

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.¹

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

¹One 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 profile².

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
```

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

```

    X = FOGRA39L_coated.icc, % or x or longer GTS_PDFX
    ISO_PDFF1 = whatever.icc
  }
}

```

sRGB.icc and FOGRA39L_coated.icc (from the colorprofiles package) are predefined and will work directly³. 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_PDFF1 = 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^AT_EX 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.

³The `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`⁴. 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 code⁵, 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

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

⁵with 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^AT_EX 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`⁶. 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}
```

⁶Extracting 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}{2025-08-05}{0.96v}
4 {PDF-Standards---LaTeX PDF management 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`
`\pdfmeta_standard_verify:nTF`

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

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

```

(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`

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

(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`

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

(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:

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

At begin document this should be checked:

```
106 \hook_gput_code:nnn {begindocument} {pdf}
107 {
108   \pdfmeta_standard_verify:nF { annot_flags }
109   { \__pdfmeta_verify_pdfa_annot_flags: }
110   \pdfmeta_standard_verify:nF { Trailer_no_Info }
111   { \__pdf_backend_omit_info:n {1} }
```

```

112 \pdfmeta_standard_verify:nF { no_CharSet }
113 { \_pdf_backend_omit_charset:n {1} }
114 \pdfmeta_standard_verify:nF { omit_CID }
115 { \_pdf_backend_omit_cidset:n {1} }
116 \pdfmeta_standard_verify:nnF { min_pdf_version }
117 { \pdf_version: }
118 { \msg_warning:nneee {pdf}{wrong-pdfversion}
119   {\pdf_version:}{low}
120   {
121     \pdfmeta_standard_item:n{type}
122     -
123     \pdfmeta_standard_item:n{level}
124   }
125 }
126 \pdfmeta_standard_verify:nnF { max_pdf_version }
127 { \pdf_version: }
128 { \msg_warning:nneee {pdf}{wrong-pdfversion}
129   {\pdf_version:}{high}
130   {
131     \pdfmeta_standard_item:n{type}
132     -
133     \pdfmeta_standard_item:n{level}
134   }
135 }
136 }

```

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
137 \prop_new:c { g__pdfmeta_standard_pdf/A-1B_prop }
138 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/A-1B_prop }
139 {
140   ,name           = pdf/A-1B
141   ,type           = A
142   ,level          = 1
143   ,conformance    = B
144   ,year           = 2005
145   ,min_pdf_version = 1.4      %minimum
146   ,max_pdf_version = 1.4      %minimum
147   ,no_encryption  =
148   ,no_external_content = % no F, FFilter, or FDecodeParms in stream dicts
149   ,no_embed_content = % no EF key in filespec, no /Type/EmbeddedFiles
150   ,max_string_size = 65535
151   ,max_array_size = 8191
152   ,max_dict_size  = 4095
153   ,max_obj_num    = 8388607
154   ,max_nest_qQ    = 28
155   ,named_actions  = {NextPage, PrevPage, FirstPage, LastPage}
156   ,annot_flags    =
157   %booleans. Only the existence of the key matter.

```

```

158 %If the entry is added it means a requirements is there
159 %(in most cases "don't use ...")
160 %
161 %=====
162 % Rule 6.1.13-1 CosDocument, isOptionalContentPresent == false
163 ,Catalog_no_OCProperties =
164 % Rule 6.9-4 The AS key shall not appear in any optional content configuration dictionary
165 % actually only starting with A-2 but doesn't harm here either
166 ,Catalog_OCProperties_no_AS=
167 %=====
168 % Rule 6.6.1-1: PDAction, S == "GoTo" || S == "GoToR" || S == "Thread"
169 % || S == "URI" || S == "Named" || S == "SubmitForm"
170 % means: no /S/Launch, /S/Sound, /S/Movie, /S/ResetForm, /S/ImportData,
171 % /S/JavaScript, /S/Hide
172 ,annot_action_A = {GoTo,GoToR,Thread,URI,Named,SubmitForm}
173 %=====
174 % Rule 6.6.2-1: PDAnnot, Subtype != "Widget" || AA_size == 0
175 % means: no AA dictionary
176 ,annot_widget_no_AA =
177 %=====
178 % Rule 6.9-2: PDAnnot, Subtype != "Widget" || (A_size == 0 && AA_size == 0)
179 % (looks like a tightening of the previous rule)
180 ,annot_widget_no_A_AA =
181 %=====
182 % Rule 6.9-1 PDAcroForm, NeedAppearances == null || NeedAppearances == false
183 ,form_no_NeedAppearances =
184 %=====
185 %Rule 6.9-3 PDFFormField, AA_size == 0
186 ,form_no_AA =
187 %=====
188 % to be continued https://docs.verapdf.org/validation/pdfa-part1/
189 % - Outputintent/colorprofiles requirements
190 % an outputintent should be loaded and is unique.
191 ,outputintent_A = {GTS_PDFA1}
192 % - no Alternates key in image dictionaries
193 % - no OPI, Ref, Subtype2 with PS key in xobjects
194 % - Interpolate = false in images
195 % - no TR, TR2 in ExtGstate
196 }
197
198 %A-2b =====
199 \prop_new:c { g__pdfmeta_standard_pdf/A-2B_prop }
200 \prop_gset_eq:cc
201 { g__pdfmeta_standard_pdf/A-2B_prop }
202 { g__pdfmeta_standard_pdf/A-1B_prop }
203 \prop_gput:cnn
204 { g__pdfmeta_standard_pdf/A-2B_prop }{name}{pdf/A-2B}
205 \prop_gput:cnn
206 { g__pdfmeta_standard_pdf/A-2B_prop }{year}{2011}
207 \prop_gput:cnn
208 { g__pdfmeta_standard_pdf/A-2B_prop }{level}{2}
209 % embedding files is allowed (with restrictions)
210 \prop_gremove:cn
211 { g__pdfmeta_standard_pdf/A-2B_prop }

```

```

212 { no_embed_content }
213 \prop_gput:cnn
214 { g__pdfmeta_standard_pdf/A-2B_prop }
215 { only_pdfa_embed_content }
216 {}
217 \prop_gput:cnn
218 { g__pdfmeta_standard_pdf/A-2B_prop }{max_pdf_version}{1.7}
219 \prop_gput:cnn
220 { g__pdfmeta_standard_pdf/A-2B_prop }{omit_CID}{}
221 % OCG layers are allowed (with restrictions)
222 \prop_gremove:cn
223 { g__pdfmeta_standard_pdf/A-2B_prop }
224 { Catalog_no_OCProperties }
225
226 %A-2u =====
227 \prop_new:c { g__pdfmeta_standard_pdf/A-2U_prop }
228 \prop_gset_eq:cc
229 { g__pdfmeta_standard_pdf/A-2U_prop }
230 { g__pdfmeta_standard_pdf/A-2B_prop }
231 \prop_gput:cnn
232 { g__pdfmeta_standard_pdf/A-2U_prop }{name}{pdf/A-2U}
233 \prop_gput:cnn
234 { g__pdfmeta_standard_pdf/A-2U_prop }{conformance}{U}
235 \prop_gput:cnn
236 { g__pdfmeta_standard_pdf/A-2U_prop }{unicode}{}
237
238 %A-2a =====
239 \prop_new:c { g__pdfmeta_standard_pdf/A-2A_prop }
240 \prop_gset_eq:cc
241 { g__pdfmeta_standard_pdf/A-2A_prop }
242 { g__pdfmeta_standard_pdf/A-2B_prop }
243 \prop_gput:cnn
244 { g__pdfmeta_standard_pdf/A-2A_prop }{name}{pdf/A-2A}
245 \prop_gput:cnn
246 { g__pdfmeta_standard_pdf/A-2A_prop }{conformance}{A}
247 \prop_gput:cnn
248 { g__pdfmeta_standard_pdf/A-2A_prop }{tagged}{}
249
250
251 %A-3b =====
252 \prop_new:c { g__pdfmeta_standard_pdf/A-3B_prop }
253 \prop_gset_eq:cc
254 { g__pdfmeta_standard_pdf/A-3B_prop }
255 { g__pdfmeta_standard_pdf/A-2B_prop }
256 \prop_gput:cnn
257 { g__pdfmeta_standard_pdf/A-3B_prop }{name}{pdf/A-3B}
258 \prop_gput:cnn
259 { g__pdfmeta_standard_pdf/A-3B_prop }{year}{2012}
260 \prop_gput:cnn
261 { g__pdfmeta_standard_pdf/A-3B_prop }{level}{3}
262 % embedding files is allowed
263 \prop_gremove:cn
264 { g__pdfmeta_standard_pdf/A-3B_prop }
265 { only_pdfa_embed_content }

```



```

266 %A-3u =====
267 \prop_new:c { g__pdfmeta_standard_pdf/A-3U_prop }
268 \prop_gset_eq:cc
269 { g__pdfmeta_standard_pdf/A-3U_prop }
270 { g__pdfmeta_standard_pdf/A-3B_prop }
271 \prop_gput:cnn
272 { g__pdfmeta_standard_pdf/A-3U_prop }{name}{pdf/A-3U}
273 \prop_gput:cnn
274 { g__pdfmeta_standard_pdf/A-3U_prop }{conformance}{U}
275 \prop_gput:cnn
276 { g__pdfmeta_standard_pdf/A-3U_prop }{unicode}{ }
277
278 %A-3a =====
279 \prop_new:c { g__pdfmeta_standard_pdf/A-3A_prop }
280 \prop_gset_eq:cc
281 { g__pdfmeta_standard_pdf/A-3A_prop }
282 { g__pdfmeta_standard_pdf/A-3B_prop }
283 \prop_gput:cnn
284 { g__pdfmeta_standard_pdf/A-3A_prop }{name}{pdf/A-3A}
285 \prop_gput:cnn
286 { g__pdfmeta_standard_pdf/A-3A_prop }{conformance}{A}
287 \prop_gput:cnn
288 { g__pdfmeta_standard_pdf/A-3A_prop }{tagged}{ }
289
290 %A-4 =====
291 \prop_new:c { g__pdfmeta_standard_pdf/A-4_prop }
292 \prop_gset_eq:cc
293 { g__pdfmeta_standard_pdf/A-4_prop }
294 { g__pdfmeta_standard_pdf/A-3U_prop }
295 \prop_gput:cnn
296 { g__pdfmeta_standard_pdf/A-4_prop }{name}{pdf/A-4}
297 \prop_gput:cnn
298 { g__pdfmeta_standard_pdf/A-4_prop }{level}{4}
299 \prop_gput:cnn
300 { g__pdfmeta_standard_pdf/A-4_prop }{min_pdf_version}{2.0}
301 \prop_gput:cnn
302 { g__pdfmeta_standard_pdf/A-4_prop }{year}{2020}
303 \prop_gput:cnn
304 { g__pdfmeta_standard_pdf/A-4_prop }{no_CharSet}{ }
305 \prop_gput:cnn
306 { g__pdfmeta_standard_pdf/A-4_prop }{Trailer_no_Info}{ }
307 \prop_gput:cnn
308 { g__pdfmeta_standard_pdf/A-4_prop }{only_pdfa_embed_content}{ }
309 \prop_gremove:cn
310 { g__pdfmeta_standard_pdf/A-4_prop }{conformance}
311 \prop_gremove:cn
312 { g__pdfmeta_standard_pdf/A-4_prop }{max_pdf_version}
313 \prop_gremove:cn
314 { g__pdfmeta_standard_pdf/A-4_prop }{Catalog_OCProperties_no_AS}
315 %A-4f =====
316 \prop_new:c { g__pdfmeta_standard_pdf/A-4F_prop }
317 \prop_gset_eq:cc
318 { g__pdfmeta_standard_pdf/A-4F_prop }
319 { g__pdfmeta_standard_pdf/A-4_prop }

```

```

320 \prop_gput:cnn
321 { g__pdfmeta_standard_pdf/A-4F_prop }{conformance}{F}
322 % containsEmbeddedFiles == true ISO 19005-4:2020, Clause: 6.9, Test number: 5
323 \prop_gput:cnn
324 { g__pdfmeta_standard_pdf/A-4F_prop }{Catalog_EmbeddedFiles}{}
325 % can contain any file
326 \prop_gremove:cn
327 { 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

```

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

```

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.

```

350 \AddToHook{pdfmeta/xmp}
351 {
352   \pdfmeta_standard_verify:nF{no_embed_content}
353   {
354     \bool_lazy_or:nnT
355     { ! \int_if_zero_p:n { \g_pdffile_embed_pdfa_int } }
356     { ! \int_if_zero_p:n { \g_pdffile_embed_nonpdfa_int } }
357     {
358       \prop_get:NnNT\g__pdfmeta_standard_prop { name } \l__pdfmeta_tmpa_tl
359       {
360         \msg_warning:nne { pdf } { validation-failure }
361         {

```

```

362         Embedded-files-detected.\iow_newline:
363         This-is-not-allowed-in-standard-\l__pdfmeta_tmpa_tl
364     }
365 }
366 }
367 }
368 \pdfmeta_standard_verify:nF {only_pdfa_embed_content}
369 {
370     \int_if_zero:nF { \g_pdffile_embed_nonpdfa_int }
371     {
372         \prop_get:NnNT\g__pdfmeta_standard_prop { name }\l__pdfmeta_tmpa_tl
373         {
374             \str_if_eq:VnTF {\l__pdfmeta_tmpa_tl} { pdf/A-4 }
375             {
376                 \prop_gset_eq:cc
377                 { g__pdfmeta_standard_prop }
378                 { g__pdfmeta_standard_pdf/A-4F_prop }
379                 \msg_warning:nne { pdf } { validation-failure }
380                 {
381                     Embedded-non-PDF-files-detected.\iow_newline:
382                     Switching-standard-from-PDF/A-4-to-PDF/A-4F
383                 }
384             }
385             {
386                 \msg_warning:nne { pdf } { validation-failure }
387                 {
388                     Embedded-non-PDF-files-detected.\iow_newline:
389                     This-is-not-allowed-in-standard-\l__pdfmeta_tmpa_tl
390                 }
391             }
392         }
393     }
394 }
395 }

```

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.

```
396 \prop_new:N \g__pdfmeta_outputintents_prop
```

(End of definition for \g__pdfmeta_outputintents_prop.)

Some keys to fill the property.

```

397 \keys_define:nn { document / metadata }
398 {
399   colorprofiles .code:n =
400   {
401     \keys_set:nn { document / metadata / colorprofiles }{#1}
402   }
403 }
404 \keys_define:nn { document / metadata / colorprofiles }
405 {
406   ,A .code:n =
407   {
408     \tl_if_blank:nF {#1}
409     {
410       \prop_gput:Nnn \g__pdfmeta_outputintents_prop
411       { GTS_PDFA1 } {#1}
412     }
413   }
414   ,a .code:n =
415   {
416     \tl_if_blank:nF {#1}
417     {
418       \prop_gput:Nnn \g__pdfmeta_outputintents_prop
419       { GTS_PDFA1 } {#1}
420     }
421   }
422   ,X .code:n =
423   {
424     \tl_if_blank:nF {#1}
425     {
426       \prop_gput:Nnn \g__pdfmeta_outputintents_prop
427       { GTS_PDFX } {#1}
428     }
429   }
430   ,x .code:n =
431   {

```

```

432     \tl_if_blank:nF {#1}
433     {
434         \prop_gput:Nnn \g__pdfmeta_outputintents_prop
435         { GTS_PDFX } {#1}
436     }
437 }
438 ,unknown .code:n =
439 {
440     \tl_if_blank:nF {#1}
441     {
442         \exp_args:NNo
443         \prop_gput:Nnn \g__pdfmeta_outputintents_prop
444         { \l_keys_key_str } {#1}
445     }
446 }
447 }

```

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

```

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

```

`_pdfmeta_embed_colorprofile:n` The commands embed the profile, and write the dictionary and add it to the catalog.
`_pdfmeta_write_outputintent:nn` 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.

```

467 \cs_new_protected:Npn \__pdfmeta_embed_colorprofile:n #1#1 file name
468 {
469     \pdf_object_if_exist:nF { __color_icc_ #1 }
470     {
471         \pdf_object_new:n { __color_icc_ #1 }
472         \pdf_object_write:mne { __color_icc_ #1 } { fstream }
473         {
474             {/N\c_space_tl
475                 \prop_item:cn{c__pdfmeta_colorprofile_#1}{N}
476             }

```

```

477         {#1}
478     }
479 }
480 }
481
482 \cs_new_protected:Npn \__pdfmeta_write_outputintent:nn #1 #2 %#1 file name, #2 subtype
483 {
484     \group_begin:
485     \pdfdict_put:nne {l_pdfmeta/outputintent}{S}{/\str_convert_pdfname:n{#2}}
486     \pdfdict_put:nne {l_pdfmeta/outputintent}
487         {DestOutputProfile}
488         {\pdf_object_ref:n{ __color_icc_ #1 }}
489     \clist_map_inline:nn { OutputConditionIdentifier, Info, RegistryName }
490     {
491         \prop_get:cnNT
492         { c__pdfmeta_colorprofile_#1}
493         { ##1 }
494         \l__pdfmeta_tmpa_tl
495         {
496             \pdf_string_from_unicode:nVN {utf8/string}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_st
497             \pdfdict_put:nne
498                 {l_pdfmeta/outputintent}{##1}{\l__pdfmeta_tmpa_str}
499         }
500     }
501     \pdf_object_unnamed_write:ne {dict}{\pdfdict_use:n {l_pdfmeta/outputintent} }
502     \pdfmanagement_add:nne {Catalog}{OutputIntents}{\pdf_object_ref_last:}
503     \group_end:
504 }

```

(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

```

505 \AddToHook{begindocument/end}
506 {
507     \pdfmeta_standard_verify:nTF {outputintent_A}
508     {
509         \prop_map_inline:Nn \g__pdfmeta_outputintents_prop
510         {
511             \prop_if_exist:cTF {c__pdfmeta_colorprofile_#2}
512             {
513                 \__pdfmeta_embed_colorprofile:n
514                 {#2}
515                 \__pdfmeta_write_outputintent:nn
516                 {#2}
517                 {#1}
518             }
519             {
520                 \msg_warning:nnn{pdfmeta}{colorprofile-undefined}{#2}
521             }
522         }
523     }

```

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.

```
524     {
525         \exp_args:NNe
526         \prop_if_in:NnF
527         \g__pdfmeta_outputintents_prop
528         { \pdfmeta_standard_item:n { outputintent_A } }
529         {
530             \exp_args:NNe
531             \prop_gput:Nnn
532             \g__pdfmeta_outputintents_prop
533             { \pdfmeta_standard_item:n { outputintent_A } }
534             { sRGB.icc }
535         }
536         \exp_args:NNe
537         \prop_get:NnN
538         \g__pdfmeta_outputintents_prop
539         { \pdfmeta_standard_item:n { outputintent_A } }
540         \l__pdfmeta_tmpb_tl
541         \prop_if_exist:cTF {c__pdfmeta_colorprofile_\l__pdfmeta_tmpb_tl}
542         {
543             \exp_args:NV \__pdfmeta_embed_colorprofile:n \l__pdfmeta_tmpb_tl
544             \prop_map_inline:Nn \g__pdfmeta_outputintents_prop
545             {
546                 \exp_args:NV
547                 \__pdfmeta_write_outputintent:nn
548                 \l__pdfmeta_tmpb_tl
549                 { #1 }
550             }
551         }
552         {
553             \msg_warning:nne{pdfmeta}{colorprofile-undefined}{\l__pdfmeta_tmpb_tl}
554         }
555     }
556 }
```

3.2 Regression test

This is simply a copy of the backend function.

```
557 \cs_new_protected:Npn \pdfmeta_set_regression_data:
558     { \__pdf_backend_set_regression_data: }
```

4 XMP-Metadata implementation

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

```
559 \bool_new:N\g__pdfmeta_xmp_bool
560 \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.

```
561 \hook_gput_code:nnn{pdfmanagement/add}{pdfmanagement}
562 {
563   \pdfmanagement_add:nne {Info}{Producer}{(\c_sys_engine_exec_str-\c_sys_engine_version_str}
564   \pdfmanagement_add:nne {Info}{Creator}{(LaTeX)}
565 }
```

4.1 New document keys

```
566 \keys_define:nn { document / metadata }
567 {
568   _pdfstandard .choices:nn =
569     {A-1B,A-2A,A-2B,A-2U,A-3A,A-3B,A-3U,A-4}
570     {
571       \prop_gset_eq:Nc \g__pdfmeta_standard_prop { g__pdfmeta_standard_pdf/#1 _prop }
572       \AddToDocumentProperties [document]{pdfstandard}{#1}
573     },
574   _pdfstandard / A-4F .code:n =
575     {
576       \prop_gset_eq:Nc \g__pdfmeta_standard_prop { g__pdfmeta_standard_pdf/A-
577 4F_prop }
578       \AddToDocumentProperties [document]{pdfstandard}{A-4F}
579     },
580   _pdfstandard / A-4E .code:n =
581     {
582       \prop_gset_eq:Nc \g__pdfmeta_standard_prop { g__pdfmeta_standard_pdf/A-
583 4E_prop }
584       \AddToDocumentProperties [document]{pdfstandard}{A-4E}
585     },
586   _pdfstandard / unknown .code:n =
587     {
588       \msg_error:nnn{pdf}{unknown-standard}{#1}
589     },
590   _pdfstandard / X-4 .code:n =
591     {
592       \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-4}
593       \__pdfmeta_xmp_add_pdfxid:
594     },
595   _pdfstandard / X-4p .code:n =
596     {
597       \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-4p}
598       \__pdfmeta_xmp_add_pdfxid:
599     },
600   _pdfstandard / X-5g .code:n =
601     {
602       \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5g}
603       \__pdfmeta_xmp_add_pdfxid:
604     },
605   _pdfstandard / X-5n .code:n =
606     {
607       \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5n}
608       \__pdfmeta_xmp_add_pdfxid:
609     },
610 }
```



```

608  _pdfstandard / X-5pg .code:n =
609  {
610    \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5pg}
611    \__pdfmeta_xmp_add_pdfxid:
612  },
613  _pdfstandard / X-6 .code:n =
614  {
615    \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6p}
616    \__pdfmeta_xmp_add_pdfxid:
617  },
618  _pdfstandard / X-6n .code:n =
619  {
620    \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6n}
621    \__pdfmeta_xmp_add_pdfxid:
622  },
623  _pdfstandard / X-6p .code:n =
624  {
625    \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6p}
626    \__pdfmeta_xmp_add_pdfxid:
627  },
628  _pdfstandard / UA-1 .code:n =
629  {
630    \AddToDocumentProperties [document]{pdfstandard-UA}{{1}}{}
631    \AddToHook{begindocument/before}
632    {
633      \prop_gput:Nnn \g__pdfmeta_standard_prop {omit_CID}{}
634      \pdf_version_compare:NnF < {2.0}
635      {
636        \msg_warning:nneee
637        {pdf}{wrong-pdfversion}
638        {\pdf_version:}{high}{UA-1}
639      }
640    }
641  },

```

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.

```

642  _pdfstandard / UA-2 .code:n =
643  {
644    \AddToDocumentProperties [document]{pdfstandard-UA}{{2}}{2024}}

```

2025-06-11 Trailer_no_Info is only a should not a shall in UA-2 so we do not force it.

```

645  %\AddToHook{begindocument/before}
646  %{\prop_gput:Nnn \g__pdfmeta_standard_prop {Trailer_no_Info}{}
647  \AddToHook{begindocument/before}
648  {
649    \__pdfmeta_xmp_wtpdf_accessibility_declaration:
650    \__pdfmeta_xmp_wtpdf_reuse_declaration:
651    \pdf_version_compare:NnT < {2.0}
652    {
653      \msg_warning:nneee
654      {pdf}{wrong-pdfversion}
655      {\pdf_version:}{low}{UA-2}

```

```

656     }
657   }
658 },
659 xmp .choice:,
660 xmp / true .code:n = { \bool_gset_true:N \g__pdfmeta_xmp_bool },
661 xmp / false .code:n = { \bool_gset_false:N \g__pdfmeta_xmp_bool},
662 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.

```

663 xmp / wtpdf .code:n =
664 {
665   \keys_set:nn {__pdfmeta/xmp}{#1}
666 },
667 }
668 \keys_define:nn {__pdfmeta/xmp}
669 {
670   reuse .choice:,
671   reuse / true .code:n = \__pdfmeta_xmp_wtpdf_reuse_declaration:,
672   reuse / false .code:n =
673   {
674     \cs_set_eq:NN \__pdfmeta_xmp_wtpdf_reuse_declaration: \prg_do_nothing:
675   },
676   accessibility .choice:,
677   accessibility / true .code:n = \__pdfmeta_xmp_wtpdf_accessibility_declaration:,
678   accessibility /false .code:n =
679   {
680     \cs_set_eq:NN \__pdfmeta_xmp_wtpdf_accessibility_declaration: \prg_do_nothing:
681   },
682 }

```

XMP debugging option

```

683 \bool_new:N \g__pdfmeta_xmp_export_bool
684 \str_new:N \g__pdfmeta_xmp_export_str
685
686 \keys_define:nn { document / metadata }
687 {
688   ,debug / xmp-export .choice:
689   ,debug / xmp-export / true .code:n=
690   {
691     \bool_gset_true:N \g__pdfmeta_xmp_export_bool
692     \str_gset_eq:NN \g__pdfmeta_xmp_export_str \c_sys_jobname_str
693   }
694   ,debug / xmp-export / false .code:n =
695   {
696     \bool_gset_false:N \g__pdfmeta_xmp_export_bool
697   }
698   ,debug / xmp-export /unknown .code:n =
699   {
700     \bool_gset_true:N \g__pdfmeta_xmp_export_bool
701     \str_gset:Nn \g__pdfmeta_xmp_export_str { #1 }
702   }
703   ,debug / xmp-export .default:n = true
704 }

```

4.2 Messages

```
705 \msg_new:nnn{pdfmeta}{xmp-defined}{The~XMP~#1~`#2`~is~already~declared}
706 \msg_new:nnn{pdfmeta}{xmp-undefined}{The~XMP~#1~`#2`~is~undefined}
707 \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:`

```
708 \bool_lazy_or:nnTF
709 { \sys_if_engine luatex_p: }
710 { \sys_if_engine xetex_p: }
711 {
712   \cs_new:Npn \_pdfmeta_xmp_generate_bom:
713     { \char_generate:nn {"FEFF"}{12} }
714 }
715 {
716   \cs_new:Npn \_pdfmeta_xmp_generate_bom:
717     {
718       \char_generate:nn {"EF"}{12}
719       \char_generate:nn {"BB"}{12}
720       \char_generate:nn {"BF"}{12}
721     }
722 }
```

(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`

```
723 \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:`

```
724 \cs_new:Npn \_pdfmeta_xmp_indent:
725   {
726     \iow_newline:
727     \prg_replicate:nn {\l_pdfmeta_xmp_indent_int}{\c_space_tl}
728   }
729
730 \cs_new:Npn \_pdfmeta_xmp_indent:n #1
731   {
732     \iow_newline:
733     \prg_replicate:nn {#1}{\c_space_tl}
734   }
735
736 \cs_new_protected:Npn \_pdfmeta_xmp_incr_indent:
```

```

737 {
738   \int_incr:N \l__pdfmeta_xmp_indent_int
739 }
740
741 \cs_new_protected:Npn \__pdfmeta_xmp_decr_indent:
742 {
743   \int_decr:N \l__pdfmeta_xmp_indent_int
744 }

```

(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.

\l__pdfmeta_xmp_date_regex

```

745 \regex_new:N \l__pdfmeta_xmp_date_regex
746 \regex_set:Nn \l__pdfmeta_xmp_date_regex
747 {D:(\d{4})(\d{2})(\d{2})(\d{2})?(\d{2})?(\d{2})?([Z\+|-])?(?:\d{2}\')?(?:\d{2}\')?}

```

(End of definition for \l__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.

```

748 \cs_new_protected:Npn \__pdfmeta_xmp_date_split:nN #1 #2 %#1 date, #2 seq
749 {
750   \regex_split:NnN \l__pdfmeta_xmp_date_regex {#1} #2
751 }
752 \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.

```

753 \cs_new:Npn \__pdfmeta_xmp_print_date:N #1 % seq
754 {
755   \tl_if_blank:eTF { \seq_item:Nn #1 {1} }
756   {
757     \seq_item:Nn #1 {2} %year
758     -
759     \seq_item:Nn #1 {3} %month
760     -
761     \seq_item:Nn #1 {4} % day
762     \tl_if_blank:eF
763     { \seq_item:Nn #1 {5} }
764     { T \seq_item:Nn #1 {5} } %hour
765     \tl_if_blank:eF
766     { \seq_item:Nn #1 {6} }
767     { : \seq_item:Nn #1 {6} } %minutes
768     \tl_if_blank:eF
769     { \seq_item:Nn #1 {7} }

```

```

770     { : \seq_item:Nn #1 {7} } %seconds
771     \seq_item:Nn #1 {8} %Z,+,-
772     \seq_item:Nn #1 {9}
773     \tl_if_blank:eF
774     { \seq_item:Nn #1 {10} }
775     { : \seq_item:Nn #1 {10} }
776   }
777   {
778     \seq_item:Nn #1 {1}
779   }
780 }

```

(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.

```

781 \tl_new:N \l__pdfmeta_xmp_currentdate_tl
782 \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.

```

783 \cs_new_protected:Npn \__pdfmeta_xmp_date_get:nNN #1 #2 #3
784   %#1 property, #2 tl var with PDF date, #3 seq for split date
785   {
786     \tl_set:Ne #2 { \GetDocumentProperties{#1} }
787     \tl_if_blank:VTF #2
788     {
789       \seq_set_eq:NN #3 \l__pdfmeta_xmp_currentdate_seq
790       \tl_set_eq:NN #2 \l__pdfmeta_xmp_currentdate_tl
791     }
792     {
793       \__pdfmeta_xmp_date_split:VN #2 #3
794     }
795   }

```

(End of definition for `__pdfmeta_xmp_date_get:nNN`.)

4.3.4 UUID

We need a command to generate an uuid

`__pdfmeta_xmp_create_uuid:nN`

```

796 \cs_new_protected:Npn \__pdfmeta_xmp_create_uuid:nN #1 #2
797   {
798     \str_set:Ne#2 {\str_lowercase:f{\tex_mdffivesum:D{#1}}}
799     \str_set:Ne#2
800     { uuid:
801       \str_range:Nnn #2{1}{8}
802       -\str_range:Nnn#2{9}{12}
803       -4\str_range:Nnn#2{13}{15}
804       -8\str_range:Nnn#2{16}{18}

```

```

805     -\str_range:Nnn#2{19}{30}
806   }
807 }

```

(End of definition for __pdfmeta_xmp_create_uuid: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.

```

808 \cs_new_protected:Npn \__pdfmeta_xmp_sanitize:nN #1 #2
809   % #1 input string, #2 str with the output
810   {
811     \group_begin:
812     \text_declare_purify_equivalent:Nn \& {\tl_to_str:N & }
813     \text_declare_purify_equivalent:Nn \texttilde {\c_tilde_str}
814     \tl_set:Nc \l__pdfmeta_tmpa_tl { \text_purify:n {#1} }
815     \str_gset:Nc \g__pdfmeta_tmpa_str { \tl_to_str:N \l__pdfmeta_tmpa_tl }
816     \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {&}{&amp;}
817     \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {<}{&lt;}
818     \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {>}{&gt;}
819     \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {"}{&quot;}
820     \group_end:
821     \str_set_eq:NN #2 \g__pdfmeta_tmpa_str
822   }
823
824 \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

```

```

825 \tl_new:N \l__pdfmeta_xmp_doclang_tl
826 \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

```

```

827 \regex_new:N\l__pdfmeta_xmp_lang_regex
828 \regex_set:Nn\l__pdfmeta_xmp_lang_regex {\A\([A-Za-z\-\+)](.*)}

```

(End of definition for \l__pdfmeta_xmp_lang_regex.)

```

829 \cs_new_protected:Npn \__pdfmeta_xmp_lang_get:nNN #1 #2 #3
830 % #1 text, #2 tl var for lang match (or default), #3 tl var for text
831 {
832   \regex_extract_once:NnN \l__pdfmeta_xmp_lang_regex {#1}\l__pdfmeta_tmpa_seq
833   \seq_if_empty:NTF \l__pdfmeta_tmpa_seq
834     {
835       \tl_set:Nn #2 \l__pdfmeta_xmp_metalang_tl
836       \tl_set:Nn #3 {#1}
837     }
838     {
839       \tl_set:Ne #2 {\seq_item:Nn\l__pdfmeta_tmpa_seq{2}}
840       \tl_set:Ne #3 {\seq_item:Nn\l__pdfmeta_tmpa_seq{3}}
841     }
842   }
843 \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
```

```
844 \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.

```

845 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_chunk:n #1
846 {
847   \tl_gput_right:Ne\g__pdfmeta_xmp_packet_tl
848     {
849       \__pdfmeta_xmp_indent: \exp_not:n{#1}
850     }
851 }
852 \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.

```

853 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_chunk:nN #1 #2
854 {
855   \tl_put_right:Ne#2
856     {
857       \__pdfmeta_xmp_indent: \exp_not:n{#1}
858     }
859 }
860 \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.

```
861 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_open:nn #1 #2 %\#1 prefix #2 name
862   {
863     \_pdfmeta_xmp_add_packet_chunk:n {<#1:#2>}
864     \_pdfmeta_xmp_incr_indent:
865   }
866 \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.

```
867 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_open_attr:nnn #1 #2 #3
868   %\#1 prefix #2 name #3 attr
869   {
870     \_pdfmeta_xmp_add_packet_chunk:n {<#1:#2~#3>}
871     \_pdfmeta_xmp_incr_indent:
872   }
873 \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.

```
874 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_close:nn #1 #2 %\#1 prefix #2:name
875   {
876     \_pdfmeta_xmp_decr_indent:
877     \_pdfmeta_xmp_add_packet_chunk:n {</#1:#2>}
878   }
```

(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.

```
879 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_line:nnn #1 #2 #3
880   %\#1 prefix #2 name #3 content
881   {
882     \tl_if_blank:nF {#3}
883     {
884       \_pdfmeta_xmp_sanitizate:nN {#3}\l_pdfmeta_tmpa_str
885       \_pdfmeta_xmp_add_packet_chunk:e {<#1:#2>\l_pdfmeta_tmpa_str</#1:#2>}
886     }
887   }
888 \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.

```
889 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_line:nnnN #1 #2 #3 #4
890   %\#1 prefix #2 name #3 content #4 tl_var to prebuilt.
```



```

891 {
892   \tl_if_blank:nF {#3}
893   {
894     \__pdfmeta_xmp_sanitizе:nN {#3}\l__pdfmeta_tmpa_str
895     \__pdfmeta_xmp_add_packet_chunk:eN {<#1:#2>\l__pdfmeta_tmpa_str</#1:#2>} #4
896   }
897 }
898 \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

```

899 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_line_attr:nnnn #1 #2 #3 #4
900   % #1 prefix #2 name #3 attribute #4 content
901   {
902     \tl_if_blank:nF {#4}
903     {
904       \__pdfmeta_xmp_sanitizе:nN {#4}\l__pdfmeta_tmpa_str
905       \__pdfmeta_xmp_add_packet_chunk:e {<#1:#2-#3>\l__pdfmeta_tmpa_str</#1:#2>}
906     }
907   }
908 \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

```

909 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_line_default:nnnn #1 #2 #3 #4
910   % #1 prefix #2 name #3 default #4 content
911   {
912     \tl_if_blank:nTF { #4 }
913     {
914       \tl_set:Nn \l__pdfmeta_tmpa_tl {#3}
915     }
916     {
917       \tl_set:Nn \l__pdfmeta_tmpa_tl {#4}
918     }
919     \__pdfmeta_xmp_add_packet_line:nnV {#1}{#2}\l__pdfmeta_tmpa_tl
920   }
921 \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:

```

922 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_list_simple:nnnn #1 #2 #3 #4
923   % #1 prefix, #2 name, #3 type (Seq/Bag/Alt) #4 a clist
924   {
925     \clist_if_empty:nF { #4 }
926     {
927       \__pdfmeta_xmp_add_packet_open:nn {#1}{#2}
928       \__pdfmeta_xmp_add_packet_open:nn {rdf}{#3}
929       \clist_map_inline:nn {#4}

```

```

930         {
931             \_pdfmeta_xmp_add_packet_line:nnn
932             {rdf}{li}{##1}
933         }
934         \_pdfmeta_xmp_add_packet_close:nn{rdf}{#3}
935         \_pdfmeta_xmp_add_packet_close:nn {#1}{#2}
936     }
937 }
938 \cs_generate_variant:Nn \_pdfmeta_xmp_add_packet_list_simple:nnnn {nnnV,nnne}

```

Here we check also for the language.

```

939 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_list:nnnn #1 #2 #3 #4
940 %#1 prefix, #2 name, #3 type (Seq/Bag/Alt) #4 a clist
941 {
942     \clist_if_empty:nF { #4 }
943     {
944         \_pdfmeta_xmp_add_packet_open:nn {#1}{#2}
945         \_pdfmeta_xmp_add_packet_open:nn {rdf}{#3}
946         \clist_map_inline:nn {#4}
947         {
948             \_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! x-default has to be the first entry, see issue #92, so we have to go through the list twice.

```

949         \tl_if_eq:eeT{\l__pdfmeta_tmpa_tl}{\l__pdfmeta_xmp_metalang_tl}
950         {
951             \_pdfmeta_xmp_add_packet_line_attr:nneV
952             {rdf}{li}{xml:lang="x-default" }\l__pdfmeta_tmpb_tl
953             \_pdfmeta_xmp_add_packet_line_attr:nneV
954             {rdf}{li}{xml:lang="\l__pdfmeta_tmpa_tl" }\l__pdfmeta_tmpb_tl
955         }
956     }
957     \clist_map_inline:nn {#4}
958     {
959         \_pdfmeta_xmp_lang_get:nNN {##1}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl
960         \tl_if_eq:eeF{\l__pdfmeta_tmpa_tl}{\l__pdfmeta_xmp_metalang_tl}
961         {
962             \_pdfmeta_xmp_add_packet_line_attr:nneV
963             {rdf}{li}{xml:lang="\l__pdfmeta_tmpa_tl" }\l__pdfmeta_tmpb_tl
964         }
965     }
966     \_pdfmeta_xmp_add_packet_close:nn{rdf}{#3}
967     \_pdfmeta_xmp_add_packet_close:nn {#1}{#2}
968 }
969 }
970 \cs_generate_variant:Nn \_pdfmeta_xmp_add_packet_list:nnnn {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.

```
971 \cs_new_protected:Npn \_pdfmeta_xmp_build_packet:  
972 {
```

Get the main languages

```
973 \tl_set:Ne \l__pdfmeta_xmp_doclang_tl {\GetDocumentProperties{document/lang}}  
974 \tl_set:Ne \l__pdfmeta_xmp_metalang_tl {\GetDocumentProperties{hyperref/pdfmetalang}}  
975 \tl_if_blank:VT \l__pdfmeta_xmp_metalang_tl  
976 { \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

```
977 \_pdfmeta_xmp_build_iptc_data:N \l__pdfmeta_xmp_iptc_data_tl  
978 \tl_if_empty:NT \l__pdfmeta_xmp_iptc_data_tl  
979 {  
980 \seq_remove_all:Nn \l__pdfmeta_xmp_schema_seq { Iptc4xmpCore }  
981 }
```

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

```
982 \_pdfmeta_xmp_add_packet_chunk:e  
983 {<?xpacket~begin="\_pdfmeta_xmp_generate_bom:"-id="W5M0MpCehiHzreSzNTczkc9d"?>}  
984 \_pdfmeta_xmp_add_packet_open:nn{x}{xmpmeta~xmlns:x="adobe:ns:meta/"}  
985 \_pdfmeta_xmp_add_packet_open:ne{rdf}  
986 {RDF~xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\c_hash_str"}
```

The rdf namespaces

```
987 \_pdfmeta_xmp_add_packet_open_attr:nne  
988 {rdf}{Description}{rdf:about="" \g__pdfmeta_xmp_xmlns_tl}
```

The extensions

```
989 \_pdfmeta_xmp_add_packet_open:nn{pdfaExtension}{schemas}  
990 \_pdfmeta_xmp_add_packet_open:nn {rdf}{Bag}  
991 \seq_map_inline:Nn \l__pdfmeta_xmp_schema_seq  
992 {  
993 \tl_use:c { g__pdfmeta_xmp_schema_##1_tl }  
994 }  
995 \_pdfmeta_xmp_add_packet_close:nn {rdf}{Bag}  
996 \_pdfmeta_xmp_add_packet_close:nn {pdfaExtension}{schemas}
```

Now starts the part with the data.

```
997 % data  
998 \_pdfmeta_xmp_build_pdf:  
999 \_pdfmeta_xmp_build_xmpRights:  
1000 \_pdfmeta_xmp_build_standards: %pdfaid, pdfxid, pdfuaid
```

```

1001     \_pdfmeta_xmp_build_pdfd:
1002     \_pdfmeta_xmp_build_dc:
1003     \_pdfmeta_xmp_build_photoshop:
1004     \_pdfmeta_xmp_build_xmp:
1005     \_pdfmeta_xmp_build_xmpMM:
1006     \_pdfmeta_xmp_build_prism:
1007     \_pdfmeta_xmp_build_iptc:
1008     \_pdfmeta_xmp_build_user: %user additions
1009 % end
1010     \_pdfmeta_xmp_add_packet_close:nn {rdf}{Description}
1011     \_pdfmeta_xmp_add_packet_close:nn {rdf}{RDF}
1012     \_pdfmeta_xmp_add_packet_close:nn {x}{xmpmeta}
1013     \int_set:Nn \l__pdfmeta_xmp_indent_int{20}
1014     \prg_replicate:nn{10}{\_pdfmeta_xmp_add_packet_chunk:n {}}
1015     \int_zero:N \l__pdfmeta_xmp_indent_int
1016     \_pdfmeta_xmp_add_packet_chunk:n {<?xpacket~end="w"?>}
1017 }

```

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

```

1018 \str_new:N \g__pdfmeta_xmp_xmlns_tl
1019 \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`

```

1020 \cs_new_protected:Npn \_pdfmeta_xmp_xmlns_new:nn #1 #2
1021 {
1022     \prop_gput:Nnn \g__pdfmeta_xmp_xmlns_prop {#1}{#2}
1023     \tl_gput_right:Ne \g__pdfmeta_xmp_xmlns_tl
1024     {
1025         \_pdfmeta_xmp_indent:n{4} xmlns:\exp_not:n{#1="#2"}
1026     }
1027 }

```

(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.

```

1028 \_pdfmeta_xmp_xmlns_new:nn {pdf}      {http://ns.adobe.com/pdf/1.3/}
1029 \_pdfmeta_xmp_xmlns_new:nn {xmpRights}{http://ns.adobe.com/xap/1.0/rights/}
1030 \_pdfmeta_xmp_xmlns_new:nn {dc}      {http://purl.org/dc/elements/1.1/}
1031 \_pdfmeta_xmp_xmlns_new:nn {photoshop}{http://ns.adobe.com/photoshop/1.0/}
1032 \_pdfmeta_xmp_xmlns_new:nn {xmp}     {http://ns.adobe.com/xap/1.0/}

```

```

1033 \_pdfmeta_xmp_xmlns_new:nn {xmpMM}      {http://ns.adobe.com/xap/1.0/mm/}
1034 \_pdfmeta_xmp_xmlns_new:nn {stEvt}
1035      {http://ns.adobe.com/xap/1.0/sType/ResourceEvent\c_hash_str}
1036 \_pdfmeta_xmp_xmlns_new:nn {pdfaid}     {http://www.aiim.org/pdfa/ns/id/}
1037 \_pdfmeta_xmp_xmlns_new:nn {pdfuaid}    {http://www.aiim.org/pdfua/ns/id/}
1038 \_pdfmeta_xmp_xmlns_new:nn {pdfx}      {http://ns.adobe.com/pdfx/1.3/}
1039 %\_pdfmeta_xmp_xmlns_new:nn {pdfxid}    {http://www.npes.org/pdfx/ns/id/}
1040 \_pdfmeta_xmp_xmlns_new:nn {prism}      {http://prismstandard.org/namespaces/basic/3.0/}
1041 %\_pdfmeta_xmp_xmlns_new:nn {jav}      {http://www.niso.org/schemas/jav/1.0/}
1042 %\_pdfmeta_xmp_xmlns_new:nn {xmpTPg}   {http://ns.adobe.com/xap/1.0/t/pg/}
1043 \_pdfmeta_xmp_xmlns_new:nn {stFnt}     {http://ns.adobe.com/xap/1.0/sType/Font\c_hash_str}
1044 \_pdfmeta_xmp_xmlns_new:nn {Iptc4xmpCore}{http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
1045 \_pdfmeta_xmp_xmlns_new:nn {pdfaExtension}{http://www.aiim.org/pdfa/ns/extension/}
1046 \_pdfmeta_xmp_xmlns_new:nn {pdfaSchema}{http://www.aiim.org/pdfa/ns/schema\c_hash_str}
1047 \_pdfmeta_xmp_xmlns_new:nn {pdfaProperty}{http://www.aiim.org/pdfa/ns/property\c_hash_str}
1048 \_pdfmeta_xmp_xmlns_new:nn {pdfaType}  {http://www.aiim.org/pdfa/ns/type\c_hash_str}
1049 \_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

```
1050 \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.

```

1051 \cs_new_protected:Npn \_pdfmeta_xmp_schema_new:nnn #1 #2 #3
1052   %#1 name #2 prefix, #3 text
1053   {
1054     \tl_if_exist:cTF { g__pdfmeta_xmp_schema_#2_tl }
1055     {
1056       \msg_warning:nnnn{pdfmeta}{xmp-defined}{schema}{#2}
1057     }
1058     {
1059       \seq_put_right:Nn \l__pdfmeta_xmp_schema_seq { #2 }
1060       \tl_new:c { g__pdfmeta_xmp_schema_#2_tl }
1061       \tl_new:c { g__pdfmeta_xmp_schema_#2_properties_tl }
1062       \tl_gput_right:cn { g__pdfmeta_xmp_schema_#2_tl }
1063       {
1064         \_pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1065         \_pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{schema}{#1}
1066         \_pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{prefix}{#2}
1067         \_pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{namespaceURI}{#3}
1068         \_pdfmeta_xmp_add_packet_open:nn {pdfaSchema}{property}

```

```

1069         \_pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1070         \tl_use:c { g__pdfmeta_xmp_schema_#2_properties_tl }
1071         \_pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1072         \_pdfmeta_xmp_add_packet_close:nn {pdfaSchema}{property}
1073         \cs_if_exist_use:c {__pdfmeta_xmp_schema_#2_additions:}
1074         \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1075     }
1076 }
1077 }

```

(End of definition for _pdfmeta_xmp_schema_new:nnn.)

_pdfmeta_xmp_property_new:nnnnn This adds a property to a schema.

```

1078 \prop_new:N\g__pdfmeta_xmp_schema_property_prop
1079 \cs_new_protected:Npn \_pdfmeta_xmp_property_new:nnnnn #1 #2 #3 #4 #5 %
1080     %#1 schema #2 name, #3 type, #4 category #5 description
1081     {
1082     \tl_if_exist:cTF { g__pdfmeta_xmp_schema_#1_properties_tl }
1083     {
1084     \prop_get:NeNF \g__pdfmeta_xmp_schema_property_prop {#1:#2}\l__pdfmeta_tmpa_tl
1085     {
1086     \prop_gput:Nee \g__pdfmeta_xmp_schema_property_prop {#1:#2}{#3}
1087     \tl_gput_right:cn { g__pdfmeta_xmp_schema_#1_properties_tl }
1088     {
1089     \_pdfmeta_xmp_add_packet_open:nn {rdf}{li-rdf:parseType="Resource"}
1090     \_pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{name}{#2}
1091     \_pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{valueType}{#3}
1092     \_pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{category}{#4}
1093     \_pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{description}{#5}
1094     \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1095     }
1096     }
1097     }
1098     {
1099     \msg_warning:nnnn{pdfmeta}{xmp-undefined}{schema}{#1}
1100     }
1101 }

```

(End of definition for _pdfmeta_xmp_property_new:nnnnn.)

_pdfmeta_xmp_add_packet_field:nnn This adds a field to a schema.

```

1102 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_field:nnn #1 #2 #3 %
1103     %#1 name #2 valuetype #3 description
1104     {
1105     \_pdfmeta_xmp_add_packet_open_attr:nnn {rdf}{li}{rdf:parseType="Resource"}
1106     \_pdfmeta_xmp_add_packet_line:nnn {pdfaField}{name}{#1}
1107     \_pdfmeta_xmp_add_packet_line:nnn {pdfaField}{valueType}{#2}
1108     \_pdfmeta_xmp_add_packet_line:nnn {pdfaField}{description}{#3}
1109     \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1110     }

```

(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-1⁷

[1] https://www.pdfe.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.

```
1111 \_pdfmeta_xmp_schema_new:nnn
1112   {XMP~Media~Management~Schema}
1113   {xmpMM}
1114   {http://ns.adobe.com/xap/1.0/mm/}
1115 \_pdfmeta_xmp_property_new:nnnnn
1116   {xmpMM}
1117   {OriginalDocumentID}
1118   {URI}
1119   {internal}
1120   {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.

pdfaid-(schema)

```
1121 \_pdfmeta_xmp_schema_new:nnn
1122   {PDF/A~Identification~Schema}
1123   {pdfaid}
1124   {http://www.aiim.org/pdfa/ns/id/}
1125 \_pdfmeta_xmp_property_new:nnnnn
1126   {pdfaid}
1127   {year}
1128   {Integer}
1129   {internal}
1130   {Year~of~standard}
1131 \_pdfmeta_xmp_property_new:nnnnn
1132   {pdfaid}
1133   {rev}
1134   {Integer}
1135   {internal}
1136   {Revision~year~of~standard}
```

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

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

⁷While 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.

pdfuaid-(schema)

```
1137 \_pdfmeta_xmp_schema_new:nnn
1138   {PDF/UA~Universal~Accessibility~Schema}
1139   {pdfuaid}
1140   {http://www.aiim.org/pdfua/ns/id/}
1141 \_pdfmeta_xmp_property_new:nnnnn
1142   {pdfuaid}
1143   {part}
1144   {Integer}
1145   {internal}
1146   {Part~of~ISO~14289~standard}
1147 \_pdfmeta_xmp_property_new:nnnnn
1148   {pdfuaid}
1149   {rev}
1150   {Integer}
1151   {internal}
1152   {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)

```
1153 \cs_new_protected:Npn \_pdfmeta_xmp_add_pdfxid:
1154 {
1155   \_pdfmeta_xmp_xmlns_new:nn {pdfxid}   {http://www.npes.org/pdfx/ns/id/}
1156   \_pdfmeta_xmp_schema_new:nnn
1157     {PDF/X~ID~Schema}
1158     {pdfxid}
1159     {http://www.npes.org/pdfx/ns/id/}
1160   \_pdfmeta_xmp_property_new:nnnnn
1161     {pdfxid}
1162     {GTS_PDFXVersion}
1163     {Text}
1164     {internal}
1165     {ID~of~PDF/X~standard}
1166 }
```

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

prism-(~~schema~~**prism**)

```
1167 \_pdfmeta_xmp_schema_new:nnn
1168   {PRISM~Basic~Metadata}
1169   {prism}
1170   {http://prismstandard.org/namespaces/basic/3.0/}
```



```

1171 \_pdfmeta_xmp_property_new:nnnnn
1172 {prism}
1173 {complianceProfile}
1174 {Text}
1175 {internal}
1176 {PRISM~specification~compliance~profile~to~which~this~document~adheres}
1177 \_pdfmeta_xmp_property_new:nnnnn
1178 {prism}
1179 {publicationName}
1180 {Text}
1181 {external}
1182 {Publication~name}
1183 \_pdfmeta_xmp_property_new:nnnnn
1184 {prism}
1185 {aggregationType}
1186 {Text}
1187 {external}
1188 {Publication~type}
1189 \_pdfmeta_xmp_property_new:nnnnn
1190 {prism}
1191 {bookEdition}
1192 {Text}
1193 {external}
1194 {Edition~of~the~book~in~which~the~document~was~published}
1195 \_pdfmeta_xmp_property_new:nnnnn
1196 {prism}
1197 {volume}
1198 {Text}
1199 {external}
1200 {Publication~volume~number}
1201 \_pdfmeta_xmp_property_new:nnnnn
1202 {prism}
1203 {number}
1204 {Text}
1205 {external}
1206 {Publication~issue~number~within~a~volume}
1207 \_pdfmeta_xmp_property_new:nnnnn
1208 {prism}
1209 {pageRange}
1210 {Text}
1211 {external}
1212 {Page~range~for~the~document~within~the~print~version~of~its~publication}
1213 \_pdfmeta_xmp_property_new:nnnnn
1214 {prism}
1215 {issn}
1216 {Text}
1217 {external}
1218 {ISSN~for~the~printed~publication~in~which~the~document~was~published}
1219 \_pdfmeta_xmp_property_new:nnnnn
1220 {prism}
1221 {eIssn}
1222 {Text}
1223 {external}
1224 {ISSN~for~the~electronic~publication~in~which~the~document~was~published}

```

```

1225 \_pdfmeta_xmp_property_new:nnnnn
1226 {prism}
1227 {isbn}
1228 {Text}
1229 {external}
1230 {ISBN~for~the~publication~in~which~the~document~was~published}
1231 \_pdfmeta_xmp_property_new:nnnnn
1232 {prism}
1233 {doi}
1234 {Text}
1235 {external}
1236 {Digital~Object~Identifier~for~the~document}
1237 \_pdfmeta_xmp_property_new:nnnnn
1238 {prism}
1239 {url}
1240 {URL}
1241 {external}
1242 {URL~at~which~the~document~can~be~found}
1243 \_pdfmeta_xmp_property_new:nnnnn
1244 {prism}
1245 {byteCount}
1246 {Integer}
1247 {internal}
1248 {Approximate~file~size~in~octets}
1249 \_pdfmeta_xmp_property_new:nnnnn
1250 {prism}
1251 {pageCount}
1252 {Integer}
1253 {internal}
1254 {Number~of~pages~in~the~print~version~of~the~document}
1255 \_pdfmeta_xmp_property_new:nnnnn
1256 {prism}
1257 {subtitle}
1258 {Text}
1259 {external}
1260 {Document's~subtitle}

```

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

iptc (schema)

```

1261 \_pdfmeta_xmp_schema_new:nnn
1262 {IPTC~Core~Schema}
1263 {Iptc4xmpCore}
1264 {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
1265 \_pdfmeta_xmp_property_new:nnnnn
1266 {Iptc4xmpCore}
1267 {CreatorContactInfo}
1268 {ContactInfo}
1269 {external}
1270 {Document~creator's~contact~information}
1271 \cs_new_protected:cpn { __pdfmeta_xmp_schema_Iptc4xmpCore_additions: }
1272 {
1273   \_pdfmeta_xmp_add_packet_open:nn{pdfaSchema}{valueType}
1274   \_pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}

```

```

1275     \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1276     \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{ContactInfo}
1277     \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
1278         {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
1279     \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{Iptc4xmpCore}
1280     \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
1281         {Basic~set~of~information~to~get~in~contact~with~a~person}
1282     \__pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
1283     \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1284     \__pdfmeta_xmp_add_packet_field:nnn{CiAdrCity}{Text}
1285         {Contact~information~city}
1286     \__pdfmeta_xmp_add_packet_field:nnn{CiAdrCtry}{Text}
1287         {Contact~information~country}
1288     \__pdfmeta_xmp_add_packet_field:nnn{CiAdrExtadr}{Text}
1289         {Contact~information~address}
1290     \__pdfmeta_xmp_add_packet_field:nnn{CiAdrPcode}{Text}
1291         {Contact~information~local~postal~code}
1292     \__pdfmeta_xmp_add_packet_field:nnn{CiAdrRegion}{Text}
1293         {Contact~information~regional~information~such~as~state~or~province}
1294     \__pdfmeta_xmp_add_packet_field:nnn{CiEmailWork}{Text}
1295         {Contact~information~email~address(es)}
1296     \__pdfmeta_xmp_add_packet_field:nnn{CiTelWork}{Text}
1297         {Contact~information~telephone~number(s)}
1298     \__pdfmeta_xmp_add_packet_field:nnn{CiUrlWork}{Text}
1299         {Contact~information~Web~URL(s)}
1300     \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1301     \__pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
1302     \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1303     \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1304     \__pdfmeta_xmp_add_packet_close:nn{pdfaSchema}{valueType}
1305 }

```

(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.)

```

1306 \cs_new_protected:Npn \__pdfmeta_xmp_schema_enable_pdfd:
1307 {
1308     \__pdfmeta_xmp_xmlns_new:nn {pdfd}{http://pdfa.org/declarations/}
1309     \__pdfmeta_xmp_schema_new:nnn
1310         {PDF~Declarations~Schema}
1311         {pdfd}
1312         {http://pdfa.org/declarations/}
1313     \__pdfmeta_xmp_property_new:nnnnn

```

```

1314 {pdfd}
1315 {declarations}
1316 {Bag-declaration}
1317 {external}
1318 {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.

```

1319 \cs_new_protected:cpn { __pdfmeta_xmp_schema_pdfd_additions: }
1320 {
1321   \__pdfmeta_xmp_add_packet_open:nn{pdfaSchema}{valueType}
1322   \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1323   \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1324   \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{claim}
1325   \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
1326     {http://pdfa.org/declarations/}
1327   \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{pdfd}
1328   \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
1329     {A-structure-describing-properties-of-an-individual claim.}
1330   \__pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
1331   \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1332   \__pdfmeta_xmp_add_packet_field:nnn{claimReport}{Text}
1333     {A-URL-to-a-report-containing-details-of-the-specific-conformance-claim}
1334   \__pdfmeta_xmp_add_packet_field:nnn{claimCredentials}{Text}
1335     {The-claimant's-credentials.}
1336   \__pdfmeta_xmp_add_packet_field:nnn{claimDate}{Text}
1337     {A-date-identifying-when-the-claim-was-made.}
1338   \__pdfmeta_xmp_add_packet_field:nnn{claimBy}{Text}
1339     {The-name-of-the-organization-and/or-individual-and/or-software-making-}
1340   \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1341   \__pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
1342   \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1343   \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1344   \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{declaration}
1345   \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
1346     {http://pdfa.org/declarations/}
1347   \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{pdfd}
1348   \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
1349     {A-structure-describing-a-single-PDF-Declaration-asserting-conformance-w
1350 identified-standard-or-profile.}
1351   \__pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
1352   \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1353   \__pdfmeta_xmp_add_packet_field:nnn{conformsTo}{Text}
1354     {A-property-containing-a-URI-specifying-the-standard-or-profile-by-the}
1355   \__pdfmeta_xmp_add_packet_field:nnn{claimData}{Bag-claim}
1356     {An-unordered-array-of-claim-data,~where~each-claim-identifies-the-natu}
1357   \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1358   \__pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
1359   \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1360   \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1361   \__pdfmeta_xmp_add_packet_close:nn{pdfaSchema}{valueType}
1362 }

```

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

```
1362 \cs_gset_eq:NN \__pdfmeta_xmp_schema_enable_pdfd: \prg_do_nothing:
1363 }
```

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
  PDFversion
1364 \cs_new_protected:Npn \__pdfmeta_xmp_build_pdf:
1365 {
```

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

```
1366 \__pdfmeta_xmp_add_packet_line_default:nnee
1367 {pdf}{Producer}
1368 {\c_sys_engine_exec_str-\c_sys_engine_version_str}
1369 {\GetDocumentProperties{hyperref/pdfproducer}}
```

Now the PDF version

```
1370 \__pdfmeta_xmp_add_packet_line:nne{pdf}{PDFVersion}{\pdf_version:}
1371 }
```

(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
  BaseUrl/baseurl
1372 \cs_new_protected:Npn \__pdfmeta_xmp_build_xmp:
1373 {
```

The creator

```
1374 \__pdfmeta_xmp_add_packet_line_default:nnee
1375 {xmp}{CreatorTool}
1376 {LaTeX}
1377 { \GetDocumentProperties{hyperref/pdfcreator} }
```

The baseurl

```
1378 \__pdfmeta_xmp_add_packet_line_default:nnee
1379 {xmp}{BaseUrl}{}
1380 { \GetDocumentProperties{hyperref/baseurl} }
```

CreationDate

```
1381     \_pdfmeta_xmp_date_get:nNN
1382         {document/creationdate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1383     \_pdfmeta_xmp_add_packet_line:nne{xmp}{CreateDate}{\_pdfmeta_xmp_print_date:N\l__pdfme
1384     \pdfmanagement_add:nne{Info}{CreateDate}{(\l__pdfmeta_tmpa_tl)}
```

ModifyDate

```
1385     \_pdfmeta_xmp_date_get:nNN
1386         {document/moddate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1387     \_pdfmeta_xmp_add_packet_line:nne{xmp}{ModifyDate}{\_pdfmeta_xmp_print_date:N\l__pdfme
1388     \pdfmanagement_add:nne{Info}{ModDate}{(\l__pdfmeta_tmpa_tl)}
```

MetadataDate

```
1389     \_pdfmeta_xmp_date_get:nNN
1390         {hyperref/pdfmetadate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1391     \_pdfmeta_xmp_add_packet_line:nne{xmp}{MetadataDate}{\_pdfmeta_xmp_print_date:N\l__pdfme
1392     }
```

(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:`

```
1393     \cs_new_protected:Npn \_pdfmeta_xmp_build_standards:
1394     {
1395         \_pdfmeta_xmp_add_packet_line:nne {pdfaid}{part}{\_pdfmeta_standard_item:n{level}}
1396         \_pdfmeta_xmp_add_packet_line:nne
1397             {pdfaid}{conformance}{\_pdfmeta_standard_item:n{conformance}}
1398         \int_compare:nNnTF {0\pdfmeta_standard_item:n{level}}<{4}
1399             {\_pdfmeta_xmp_add_packet_line:nne {pdfaid}{year} {\pdfmeta_standard_item:n{year}}}
1400             {\_pdfmeta_xmp_add_packet_line:nne {pdfaid}{rev} {\pdfmeta_standard_item:n{year}}}
1401         \_pdfmeta_xmp_add_packet_line:nne
1402             {pdfxid}{GTS_PDFXVersion}{\GetDocumentProperties{document/pdfstandard-X}}
1403         \pdfmanagement_get_documentproperties:nNT {document/pdfstandard-UA}\l__pdfmeta_tmpa_tl
1404         {
1405             \_pdfmeta_xmp_add_packet_line:nne
1406                 {pdfuaid}{part}{\exp_last_unbraced:No\use_i:nn \l__pdfmeta_tmpa_tl}
1407             \_pdfmeta_xmp_add_packet_line:nne
1408                 {pdfuaid}{rev}{\exp_last_unbraced:No\use_ii:nn \l__pdfmeta_tmpa_tl}
1409         }
1410     }
```

(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.

```
1411 \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:`

```
1412 \cs_new_protected:Npn \__pdfmeta_xmp_build_pdfd:
1413 {
1414   \prop_if_empty:NF\g__pdfmeta_xmp_pdfd_data_prop
1415   {
1416     \__pdfmeta_xmp_add_packet_open:nn{pdfd}{declarations}
1417     \__pdfmeta_xmp_add_packet_open:nn{rdf}{Bag}
1418     \prop_map_inline:Nn \g__pdfmeta_xmp_pdfd_data_prop
1419     {
1420       \__pdfmeta_xmp_build_pdfd_claim:nn{##1}{##2}
1421     }
1422     \__pdfmeta_xmp_add_packet_close:nn{rdf}{Bag}
1423     \__pdfmeta_xmp_add_packet_close:nn{pdfd}{declarations}
1424   }
1425 }
```

(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.

```
1426 \cs_new_protected:Npn \__pdfmeta_xmp_build_pdfd_claim:nn #1#2
1427 {
1428   \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1429   \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{conformsTo}{#1}
1430   \tl_if_empty:NF {#2}
1431   {
1432     \__pdfmeta_xmp_add_packet_open:nn{pdfd}{claimData}
1433     \__pdfmeta_xmp_add_packet_open:nn{rdf}{Bag}
1434     #2
1435     \__pdfmeta_xmp_add_packet_close:nn{rdf}{Bag}
1436     \__pdfmeta_xmp_add_packet_close:nn{pdfd}{claimData}
1437   }
1438   \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1439 }
```

(End of definition for __pdfmeta_xmp_build_pdfd_claim:nn.)

4.10 Photoshop

`__pdfmeta_xmp_build_photoshop:`

```
1440 \cs_new_protected:Npn \__pdfmeta_xmp_build_photoshop:
1441 {
```

pdfauthor/photshop:AuthorsPosition

```
1442   \__pdfmeta_xmp_add_packet_line:nne{photoshop}{AuthorsPosition}  
1443   { \GetDocumentProperties{hyperref/pdfauthor} }
```

pdfcaptionwriter/photshop:CaptionWriter

```
1444   \__pdfmeta_xmp_add_packet_line:nne{photoshop}{CaptionWriter}  
1445   { \GetDocumentProperties{hyperref/pdfcaptionwriter} }
```

```
1446 }
```

(End of definition for __pdfmeta_xmp_build_photshop:.)

4.11 XMP Media Management

__pdfmeta_xmp_build_xmpMM:

```
1447 \cs_new_protected:Npn \__pdfmeta_xmp_build_xmpMM:  
1448 {
```

pdfdocumentid / xmpMM:DocumentID

```
1449   \str_set:N\l__pdfmeta_tmpa_str {\GetDocumentProperties{hyperref/pdfdocumentid}}  
1450   \str_if_empty:NT \l__pdfmeta_tmpa_str  
1451   {  
1452     \__pdfmeta_xmp_create_uuid:nN  
1453     {\jobname\GetDocumentProperties{hyperref/pdftitle}}  
1454     \l__pdfmeta_tmpa_str  
1455   }  
1456   \__pdfmeta_xmp_add_packet_line:nnV{xmpMM}{DocumentID}  
1457   \l__pdfmeta_tmpa_str
```

pdfinstanceid / xmpMM:InstanceID

```
1458   \str_set:N\l__pdfmeta_tmpa_str {\GetDocumentProperties{hyperref/pdfinstanceid}}  
1459   \str_if_empty:NT \l__pdfmeta_tmpa_str  
1460   {  
1461     \__pdfmeta_xmp_create_uuid:nN  
1462     {\jobname\l__pdfmeta_xmp_currentdate_tl}  
1463     \l__pdfmeta_tmpa_str  
1464   }  
1465   \__pdfmeta_xmp_add_packet_line:nnV{xmpMM}{InstanceID}  
1466   \l__pdfmeta_tmpa_str
```

pdfversionid/xmpMM:VersionID

```
1467   \__pdfmeta_xmp_add_packet_line:nne{xmpMM}{VersionID}  
1468   { \GetDocumentProperties{hyperref/pdfversionid} }
```

pdfrendition/xmpMM:RenditionClass

```
1469   \__pdfmeta_xmp_add_packet_line:nne{xmpMM}{RenditionClass}  
1470   { \GetDocumentProperties{hyperref/pdfrendition} }
```

```
1471 }
```

(End of definition for __pdfmeta_xmp_build_xmpMM:.)

4.12 Rest of dublin Core data

```

\__pdfmeta_xmp_build_dc:
  dc:creator/pdfauthor
  dc:subject/pdfkeywords 1472 \cs_new_protected:Npn \__pdfmeta_xmp_build_dc:
    dc:type/pdftype 1473 {
pdfauthor/dc:creator
  1474 \__pdfmeta_xmp_add_packet_list_simple:nne {dc}{creator}{Seq}
  1475 { \GetDocumentProperties{hyperref/pdfauthor} }
  1476 \int_compare:nNnT {0\pdfmeta_standard_item:n{level}}={1}
  1477 { \pdfmanagement_remove:nn{Info}{Author} }

```

pdftitle/dc:title. This is rather complex as we want to support a list with different languages.

```

1478 \__pdfmeta_xmp_add_packet_list:nne {dc}{title}{Alt}
1479 { \GetDocumentProperties{hyperref/pdftitle} }

```

pdfkeywords/dc:subject

```

1480 \__pdfmeta_xmp_add_packet_list_simple:nne {dc}{subject}{Bag}
1481 { \GetDocumentProperties{hyperref/pdfkeywords} }
1482 \int_compare:nNnT {0\pdfmeta_standard_item:n{level}}={1}
1483 { \pdfmanagement_remove:nn{Info}{Keywords} }

```

pdftype/dc:type

```

1484 \pdfmanagement_get_documentproperties:nNTF { hyperref/pdftype } \l__pdfmeta_tmpa_tl
1485 {
1486   \__pdfmeta_xmp_add_packet_list_simple:nnnV {dc}{type}{Bag}\l__pdfmeta_tmpa_tl
1487 }
1488 {
1489   \__pdfmeta_xmp_add_packet_list_simple:nnnn {dc}{type}{Bag}{Text}
1490 }

```

pdfpublisher/dc:publisher

```

1491 \__pdfmeta_xmp_add_packet_list_simple:nne {dc}{publisher}{Bag}
1492 { \GetDocumentProperties{hyperref/pdfpublisher} }

```

pdfsubject/dc:description

```

1493 \__pdfmeta_xmp_add_packet_list:nne
1494 {dc}{description}{Alt}
1495 { \GetDocumentProperties{hyperref/pdfsubject} }

```

lang/pdflang/dc:language

```

1496 \__pdfmeta_xmp_add_packet_list_simple:nnnV
1497 {dc}{language}{Bag}\l__pdfmeta_xmp_doclang_tl

```

pdfidentifier/dc:identifier

```
1498 \__pdfmeta_xmp_add_packet_line:nne{dc}{identifier}  
1499 { \GetDocumentProperties{hyperref/pdfidentifier} }
```

pdfdate/dc:date

```
1500 \__pdfmeta_xmp_date_get:nNN {hyperref/pdfdate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq  
1501 \__pdfmeta_xmp_add_packet_list_simple:nne  
1502 {dc}{date}{Seq}{\__pdfmeta_xmp_print_date:N\l__pdfmeta_tmpa_seq}
```

The file format

```
1503 \__pdfmeta_xmp_add_packet_line:nnn{dc}{format}{application/pdf}
```

The source

```
1504 \__pdfmeta_xmp_add_packet_line_default:nnee  
1505 {dc}{source}  
1506 { \c_sys_jobname_str.tex }  
1507 { \GetDocumentProperties{hyperref/pdfsourc} }  
  
1508 \__pdfmeta_xmp_add_packet_list:nne{dc}{rights}{Alt}  
1509 {\GetDocumentProperties{hyperref/pdfcopyright}}  
  
1510 }
```

(End of definition for __pdfmeta_xmp_build_dc: and others.)

4.13 xmpRights

__pdfmeta_xmp_build_xmpRights:

```
1511 \cs_new_protected:Npn \__pdfmeta_xmp_build_xmpRights:  
1512 {  
1513 \__pdfmeta_xmp_add_packet_line:nne  
1514 {xmpRights}  
1515 {WebStatement}  
1516 {\GetDocumentProperties{hyperref/pdflicenseurl}}  
1517 \__pdfmeta_xmp_add_packet_line:nne  
1518 {xmpRights}  
1519 {Marked}  
1520 {  
1521 \str_case:en {\GetDocumentProperties{document/copyright}}  
1522 {  
1523 {true}{True}  
1524 {false}{False}  
1525 }  
1526 }  
1527 }
```

(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
```

```
1528 \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
```

```
1529 \cs_new_protected:Npn \__pdfmeta_xmp_build_iptc_data:N #1
```

```
1530 {
```

```
1531   \tl_clear:N #1
```

```
1532   \__pdfmeta_xmp_incr_indent:\__pdfmeta_xmp_incr_indent:\__pdfmeta_xmp_incr_indent:\__pdf
```

```
1533   \__pdfmeta_xmp_add_packet_line:nneN
```

```
1534     {Iptc4xmpCore}{CiAdrExtadr}
```

```
1535     {\GetDocumentProperties{hyperref/pdfcontactaddress}}
```

```
1536     #1
```

```
1537   \__pdfmeta_xmp_add_packet_line:nneN
```

```
1538     {Iptc4xmpCore}{CiAdrCity}
```

```
1539     {\GetDocumentProperties{hyperref/pdfcontactcity}}
```

```
1540     #1
```

```
1541   \__pdfmeta_xmp_add_packet_line:nneN
```

```
1542     {Iptc4xmpCore}{CiAdrPcode}
```

```
1543     {\GetDocumentProperties{hyperref/pdfcontactpostcode}}
```

```
1544     #1
```

```
1545   \__pdfmeta_xmp_add_packet_line:nneN
```

```
1546     {Iptc4xmpCore}{CiAdrCtry}
```

```
1547     {\GetDocumentProperties{hyperref/pdfcontactcountry}}
```

```
1548     #1
```

```
1549   \__pdfmeta_xmp_add_packet_line:nneN
```

```
1550     {Iptc4xmpCore}{CiTelWork}
```

```
1551     {\GetDocumentProperties{hyperref/pdfcontactphone}}
```

```
1552     #1
```

```
1553   \__pdfmeta_xmp_add_packet_line:nneN
```

```
1554     {Iptc4xmpCore}{CiEmailWork}
```

```
1555     {\GetDocumentProperties{hyperref/pdfcontactemail}}
```

```
1556     #1
```

```
1557   \__pdfmeta_xmp_add_packet_line:nneN
```

```
1558     {Iptc4xmpCore}{CiUrlWork}
```

```
1559     {\GetDocumentProperties{hyperref/pdfcontacturl}}
```

```
1560     #1
```

```
1561   \__pdfmeta_xmp_decr_indent:\__pdfmeta_xmp_decr_indent:\__pdfmeta_xmp_decr_indent:\__pdf
```

```
1562 }
```

(End of definition for __pdfmeta_xmp_build_iptc_data:N.)

```
\__pdfmeta_xmp_build_iptc:
```

```
1563 \cs_new_protected:Npn \__pdfmeta_xmp_build_iptc:
```

```
1564 {
```

```
1565   \tl_if_empty:NF\l__pdfmeta_xmp_iptc_data_tl
```

```

1566     {
1567         \__pdfmeta_xmp_add_packet_open_attr:nnn
1568         {Iptc4xmpCore}{CreatorContactInfo}{rdf:parseType="Resource"}
1569         \tl_gput_right:Ne\g__pdfmeta_xmp_packet_tl { \l__pdfmeta_xmp_iptc_data_tl }
1570         \__pdfmeta_xmp_add_packet_close:nn
1571         {Iptc4xmpCore}{CreatorContactInfo}
1572     }
1573 }

```

(End of definition for __pdfmeta_xmp_build_iptc:.)

4.15 Prism

```

\__pdfmeta_xmp_build_prism:
  complianceProfile
  prism:subtitle/pdfsubtitle
1574 \cs_new_protected:Npn \__pdfmeta_xmp_build_prism:
1575   {

```

The compliance profile is a fix value taken from hyperxmp

```

1576     \__pdfmeta_xmp_add_packet_line:nnn
1577     {prism}{complianceProfile}
1578     {three}

```

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

```

1579     \__pdfmeta_xmp_lang_get:eNN
1580     {\GetDocumentProperties{hyperref/pdfsubtitle}}
1581     \l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl
1582     \__pdfmeta_xmp_add_packet_line_attr:nneV
1583     {prism}{subtitle}
1584     {xml:lang="\l__pdfmeta_tmpa_tl"}
1585     \l__pdfmeta_tmpb_tl

```

Then publicationName

```

1586     \__pdfmeta_xmp_lang_get:eNN
1587     {\GetDocumentProperties{hyperref/pdfpublication}}
1588     \l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl
1589     \__pdfmeta_xmp_add_packet_line_attr:nneV
1590     {prism}{publicationName}
1591     {xml:lang="\l__pdfmeta_tmpa_tl"}
1592     \l__pdfmeta_tmpb_tl

```

Now the rest

```

1593     \__pdfmeta_xmp_add_packet_line:nne
1594     {prism}{bookEdition}
1595     {\GetDocumentProperties{hyperref/pdfbookedition}}
1596     \__pdfmeta_xmp_add_packet_line:nne
1597     {prism}{aggregationType}
1598     {\GetDocumentProperties{hyperref/pdfpubtype}}
1599     \__pdfmeta_xmp_add_packet_line:nne
1600     {prism}{volume}

```

```

1601     {\GetDocumentProperties{hyperref/pdfvolumenum}}
1602 \__pdfmeta_xmp_add_packet_line:nne
1603     {prism}{number}
1604     {\GetDocumentProperties{hyperref/pdfissuenum}}
1605 \__pdfmeta_xmp_add_packet_line:nne
1606     {prism}{pageRange}
1607     {\GetDocumentProperties{hyperref/pdfpagerange}}
1608 \__pdfmeta_xmp_add_packet_line:nne
1609     {prism}{issn}
1610     {\GetDocumentProperties{hyperref/pdfissn}}
1611 \__pdfmeta_xmp_add_packet_line:nne
1612     {prism}{eIssn}
1613     {\GetDocumentProperties{hyperref/pdfeissn}}
1614 \__pdfmeta_xmp_add_packet_line:nne
1615     {prism}{doi}
1616     {\GetDocumentProperties{hyperref/pdfdoi}}
1617 \__pdfmeta_xmp_add_packet_line:nne
1618     {prism}{url}
1619     {\GetDocumentProperties{hyperref/pdfurl}}

```

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

```

1620     \tl_set:Nc \l__pdfmeta_tmpa_tl { \GetDocumentProperties{hyperref/pdfnumpages} }
1621     \__pdfmeta_xmp_add_packet_line:nne
1622     {prism}{pageCount}
1623     {\tl_if_blank:VTF \l__pdfmeta_tmpa_tl {\PreviousTotalPages}{\l__pdfmeta_tmpa_tl}}
1624 }

```

(End of definition for __pdfmeta_xmp_build_prism:, complianceProfile, and prism:subtitle/pdfsubtitle.)

4.15.1 User additions

\g_pdfmeta_xmp_user_packet_str

```

1625 \tl_new:N \g_pdfmeta_xmp_user_packet_tl

```

(End of definition for \g_pdfmeta_xmp_user_packet_str.)

__pdfmeta_xmp_build_user:

```

1626 \cs_new_protected:Npn \__pdfmeta_xmp_build_user:
1627 {
1628     \int_zero:N \l__pdfmeta_xmp_indent_int
1629     \g_pdfmeta_xmp_user_packet_tl
1630     \int_set:Nn \l__pdfmeta_xmp_indent_int {3}
1631 }

```

(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

```
1632 \hook_new:n { pdfmeta/xmp }
1633 \AddToHook{shipout/lastpage}
1634 {
1635   \bool_if:NT\g__pdfmeta_xmp_bool
1636   {
1637     \str_if_exist:NTF\c_sys_timestamp_str
1638     {
1639       \tl_set_eq:NN \l__pdfmeta_xmp_currentdate_tl \c_sys_timestamp_str
1640     }
1641     {
1642       \file_get_timestamp:nN{\jobname.log}\l__pdfmeta_xmp_currentdate_tl
1643     }
1644     \__pdfmeta_xmp_date_split:VN\l__pdfmeta_xmp_currentdate_tl\l__pdfmeta_xmp_currentdate_tl
1645     \hook_use:n { pdfmeta/xmp }
1646     \__pdfmeta_xmp_build_packet:
1647     \exp_args:No
1648     \__pdf_backend_metadata_stream:n {\g__pdfmeta_xmp_packet_tl}
1649     \pdfmanagement_add:nne {Catalog} {Metadata}{\pdf_object_ref_last:}
1650     \bool_if:NT \g__pdfmeta_xmp_export_bool
1651     {
1652       \iow_open:Nn\g_tmpa_iow{\g__pdfmeta_xmp_export_str.xmpi}
1653       \exp_args:NNo\iow_now:Nn\g_tmpa_iow{\g__pdfmeta_xmp_packet_tl}
1654       \iow_close:N\g_tmpa_iow
1655     }
1656   }
1657 }
```

4.17 User commands

`\pdfmeta_xmp_add:n`

```
1658 \cs_new_protected:Npn \pdfmeta_xmp_add:n #1
1659 {
1660   \tl_gput_right:Nn \g__pdfmeta_xmp_user_packet_tl
1661   {
1662     \__pdfmeta_xmp_add_packet_chunk:n { #1 }
1663   }
1664 }
```

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

`\pdfmeta_xmp_xmlns_new:nn`

```
1665 \cs_new_protected:Npn \pdfmeta_xmp_xmlns_new:nn #1 #2
1666 {
1667   \prop_if_in:NnTF \g__pdfmeta_xmp_xmlns_prop {#1}
1668   {\msg_warning:nnnn{pdfmeta}{xmp-defined}{xmlns-namespace}{#1}}
1669   {\__pdfmeta_xmp_xmlns_new:nn {#1}{#2}}
1670 }
```

(End of definition for \pdfmeta_xmp_xmlns_new:nn. This function is documented on page 9.)

\pdfmeta_xmp_schema_new:nnn

```
1671 \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:nnnnn

```
1672 \cs_set_eq:NN \pdfmeta_xmp_property_new:nnnnn \__pdfmeta_xmp_property_new:nnnnn
```

(End of definition for \pdfmeta_xmp_property_new:nnnnn. This function is documented on page 10.)

\pdfmeta_xmp_add_declaration:n

\pdfmeta_xmp_add_declaration:e

```
1673 \cs_new_protected:Npn \pdfmeta_xmp_add_declaration:n #1 %conformsTo uri
1674 {
1675   \__pdfmeta_xmp_schema_enable_pdfd:
1676   \prop_gput:Nnn\g__pdfmeta_xmp_pdfd_data_prop{#1}{-}
1677 }
1678 \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:nnnnn

\pdfmeta_xmp_add_declaration:ennnn

```
1679 \cs_new_protected:Npn \pdfmeta_xmp_add_declaration:nnnnn #1#2#3#4#5
1680 %#1=conformsTo uri, #2 claimBy, #3 claimDate #4 claimCredentials #5 claimReport
1681 {
1682   \__pdfmeta_xmp_schema_enable_pdfd:
1683   \tl_set:Nn \l__pdfmeta_tmpa_tl
1684     {
1685     \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1686     \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimBy}{#2}
1687     \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimDate}{#3}
1688     \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimCredentials}{#4}
1689     \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimReport}{#5}
1690     \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1691   }
1692   \prop_get:NnNT \g__pdfmeta_xmp_pdfd_data_prop {#1}\l__pdfmeta_tmpb_tl
1693   {
1694     \tl_concat:NNN \l__pdfmeta_tmpa_tl \l__pdfmeta_tmpa_tl \l__pdfmeta_tmpb_tl
1695   }
1696   \prop_gput:Nno\g__pdfmeta_xmp_pdfd_data_prop{#1}
1697     {
1698     \l__pdfmeta_tmpa_tl
1699   }
1700 }
1701 \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:
dfmeta_xmp_wtpdf_accessibility_declaration:
1702 \cs_new:Npn \__pdfmeta_xmp_iso_today:
1703 {
1704   \int_use:N\c_sys_year_int-
1705   \int_compare:nNnT {\c_sys_month_int} < {10}{0} \int_use:N\c_sys_month_int -
1706   \int_compare:nNnT {\c_sys_day_int} < {10}{0} \int_use:N\c_sys_day_int
1707 }
1708 \cs_new_protected:Npn \__pdfmeta_xmp_wtpdf_reuse_declaration:
1709 {
1710   \pdfmeta_xmp_add_declaration:eeenn
1711   {http://pdfa.org/declarations/wtpdf\c_hash_str reuse1.0}
1712   {LaTeX-Project}
1713   {\__pdfmeta_xmp_iso_today:}{}{}
1714 }
1715 \cs_new_protected:Npn \__pdfmeta_xmp_wtpdf_accessibility_declaration:
1716 {
1717   \pdfmeta_xmp_add_declaration:ennnn
1718   {http://pdfa.org/declarations/wtpdf\c_hash_str accessibility1.0}
1719   {LaTeX-Project}
1720   {\__pdfmeta_xmp_iso_today:}{}{}
1721 }

(End of definition for \__pdfmeta_xmp_wtpdf_reuse_declaration: and \__pdfmeta_xmp_wtpdf_
accessibility_declaration:.)

1722 </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	B
\& 812	BaseUrl/baseurl <u>1372</u>
\' 747	bitset commands:
\+ 747	\bitset_set_false:Nn 97, 98, 99
\- 747, 828	\bitset_set_true:Nn 96
\[. 828	\bitset_to_arabic:N
\\ 12 100, 101, 102, 103, 104
\] 828	bool commands:
A	\bool_gset_false:N 661, 696
\A 828	\bool_gset_true:N 560, 660, 691, 700
\AddToDocumentProperties	\bool_if:NTF 1635, 1650
. 572, 577, 582, 590, 595,	\bool_lazy_or:nnTF 354, 708
600, 605, 610, 615, 620, 625, 630, 644	\bool_new:N 559, 683
\AddToHook 328, 350, 505, 631, 645, 647, 1633	

C		G	
char commands:		<code>\GetDocumentProperties</code>	786,
<code>\char_generate:nn</code>	713, 718, 719, 720	973, 974, 1369, 1377, 1380, 1402,	
clist commands:		1443, 1445, 1449, 1453, 1458, 1468,	
<code>\clist_if_empty:nTF</code>	925, 942	1470, 1475, 1479, 1481, 1492, 1495,	
<code>\clist_map_inline:nn</code> 489, 929, 946, 957		1499, 1507, 1509, 1516, 1521, 1535,	
complianceProfile	<u>1574</u>	1539, 1543, 1547, 1551, 1555, 1559,	
CreatorTool/pdfcreator	<u>1372</u>	1580, 1587, 1595, 1598, 1601, 1604,	
cs commands:		1607, 1610, 1613, 1616, 1619, 1620	
<code>\cs_generate_variant:Nn</code>	752,	group commands:	
824, 843, 852, 860, 866, 873, 888,		<code>\group_begin:</code>	336, 484, 811
898, 908, 921, 938, 970, 1678, 1701		<code>\group_end:</code>	345, 503, 820
<code>\cs_gset_eq:NN</code>	1362		
<code>\cs_if_exist:NTF</code>	44	H	
<code>\cs_if_exist_use:N</code>	1073	hook commands:	
<code>\cs_new:Npn</code>		<code>\hook_gput_code:nnn</code>	106, 561
. 22, 712, 716, 724, 730, 753, 1702		<code>\hook_new:n</code>	1632
<code>\cs_new_protected:Npn</code> 26, 61, 69, 76,		<code>\hook_use:n</code>	1645
82, 88, 94, 467, 482, 557, 736, 741,			
748, 783, 796, 808, 829, 845, 853,		I	
861, 867, 874, 879, 889, 899, 909,		int commands:	
922, 939, 971, 1020, 1051, 1079,		<code>\int_compare:nNnTF</code>	
1102, 1153, 1271, 1306, 1319, 1364,	 1398, 1476, 1482, 1705, 1706	
1372, 1393, 1412, 1426, 1440, 1447,		<code>\int_decr:N</code>	743
1472, 1511, 1529, 1563, 1574, 1626,		<code>\int_if_zero:nTF</code>	370
1658, 1665, 1673, 1679, 1708, 1715		<code>\int_if_zero_p:n</code>	355, 356
<code>\cs_set_eq:NN</code> 674, 680, 976, 1671, 1672		<code>\int_incr:N</code>	738
		<code>\int_new:N</code>	723
		<code>\int_set:Nn</code>	1013, 1630
		<code>\int_use:N</code>	1704, 1705, 1706
		<code>\int_zero:N</code>	1015, 1628
D		iow commands:	
<code>\d</code>	747	<code>\iow_close:N</code>	1654
dc commands:		<code>\iow_newline:</code>	362, 381, 388, 726, 732
<code>dc:description/pdfsubject</code>	<u>1472</u>	<code>\iow_now:Nn</code>	1653
<code>dc:identifier/pdfidentifier</code>	<u>1472</u>	<code>\iow_open:Nn</code>	1652
<code>dc:language/lang/pdflang</code>	<u>1472</u>	<code>\g_tmpa_iow</code>	1652, 1653, 1654
<code>dc:Nreator/pdfauthor</code>	<u>1472</u>	<code>iptc_U(schema)</code>	<u>1261</u>
<code>dc:publisher/pdfpublisher</code>	<u>1472</u>		
<code>dc:subject/pdfkeywords</code>	<u>1472</u>	J	
<code>dc:type/pdftype</code>	<u>1472</u>	<code>\jobname</code>	1453, 1462, 1642
<code>\DocumentMetadata</code>	2, 4		
E		K	
exp commands:		kernel internal commands:	
<code>\exp_args:NNe</code>	525, 530, 536	<code>\g_kernel_pdfmanagement_end_-</code>	
<code>\exp_args:Nne</code>	344	<code>run_code_tl</code>	332
<code>\exp_args:Nnne</code>	46	keys commands:	
<code>\exp_args:NNo</code>	442, 1653	<code>\keys_define:nn</code> 397, 404, 566, 668, 686	
<code>\exp_args:No</code>	1647	<code>\l_keys_key_str</code>	444
<code>\exp_args:NV</code>	543, 546	<code>\keys_set:nn</code>	401, 665
<code>\exp_last_unbraced:No</code>	1406, 1408		
<code>\exp_not:n</code>	849, 857, 1025	M	
F		msg commands:	
file commands:		<code>\msg_error:nnn</code>	586
<code>\file_get_timestamp:nN</code>	1642	<code>\msg_new:nnn</code>	7, 8, 10, 705, 706, 707

\msg_warning:nnn 360, 379, 386, 520, 553
 \msg_warning:nnnn .. 1056, 1099, 1668
 \msg_warning:nnnnn 118, 128, 636, 653

P

pdf commands:
 \pdf_object_if_exist:nTF 469
 \pdf_object_new:n 471
 \pdf_object_ref:n 488
 \pdf_object_ref_last: 502, 1649
 \pdf_object_unnamed_write:nn ... 501
 \pdf_object_write:nnn 472
 \pdf_string_from_unicode:nnN ... 496
 \pdf_version:
 4, 117, 119, 127, 129, 638, 655, 1370
 \pdf_version_compare:NnTF
 63, 71, 634, 651

pdf internal commands:
 __pdf_backend_metadata_stream:n
 1648
 __pdf_backend_Names_gpush:nn .. 344
 __pdf_backend_omit_charset:n .. 113
 __pdf_backend_omit_cidset:n ... 115
 __pdf_backend_omit_info:n 111
 __pdf_backend_set_regression_-
 data: 558

pdfaid~(schema) 1121

pdfannot commands:
 \pdfannot_dict_put:nnn
 100, 101, 102, 103, 104
 \l_pdfannot_F_bitset 96,
 97, 98, 99, 100, 101, 102, 103, 104

pdfdict commands:
 \pdfdict_if_empty:nTF 334
 \pdfdict_new:n 448
 \pdfdict_put:nnn
 337, 338, 449, 485, 486, 497
 \pdfdict_use:n 501

pdffile commands:
 \g_pdffile_embed_nonpdfa_int 356, 370
 \g_pdffile_embed_pdfa_int 355
 \pdffile_embed_stream:nnN 339

pdfmanagement commands:
 \pdfmanagement_add:nnn
 502, 563, 564, 1384, 1388, 1649
 \pdfmanagement_get_documentproperties:nNTF
 1403, 1484
 \pdfmanagement_remove:nn . 1477, 1483

pdfmeta commands:
 \pdfmeta_set_regression_data: 5, 557
 \pdfmeta_standard_get:nN .. 2, 26, 26
 \pdfmeta_standard_item:n
 2, 22, 22, 121,

123, 131, 133, 528, 533, 539, 1395,
 1397, 1398, 1399, 1400, 1476, 1482
 \pdfmeta_standard_verify:n ... 2, 30
 \pdfmeta_standard_verify:nn ... 2, 40
 \pdfmeta_standard_verify:nnN 2
 \pdfmeta_standard_verify:nnTF ..
 2, 40, 116, 126
 \pdfmeta_standard_verify:nTF ...
 2, 30,
 108, 110, 112, 114, 330, 352, 368, 507
 \pdfmeta_standard_verify_p:n .. 2, 30
 \pdfmeta_xmp_add:n 9, 1658, 1658
 \pdfmeta_xmp_add_declaration:n .
 9, 1673, 1673, 1678
 \pdfmeta_xmp_add_declaration:nnnnn
 9, 1679, 1679, 1701, 1710, 1717
 \pdfmeta_xmp_property_new:nnnnn
 10, 1672, 1672
 \pdfmeta_xmp_schema_new:nnn
 10, 1671, 1671
 \pdfmeta_xmp_xmlns_new:nn
 9, 1665, 1665

pdfmeta internal commands:
 __pdfmeta_embed_colorprofile:n
 467, 467, 513, 543
 \g__pdfmeta_outputintents_prop .
 396, 410, 418,
 426, 434, 443, 509, 527, 532, 538, 544
 \g__pdfmeta_standard_pdf/A-1B_-
 prop 137
 \g__pdfmeta_standard_pdf/A-2A_-
 prop 137
 \g__pdfmeta_standard_pdf/A-2B_-
 prop 137
 \g__pdfmeta_standard_pdf/A-2U_-
 prop 137
 \g__pdfmeta_standard_pdf/A-3A_-
 prop 137
 \g__pdfmeta_standard_pdf/A-3B_-
 prop 137
 \g__pdfmeta_standard_pdf/A-3U_-
 prop 137
 \g__pdfmeta_standard_pdf/A-4_-
 prop 137
 \g__pdfmeta_standard_pdf/A-4F_-
 prop 137
 \g__pdfmeta_standard_prop
 21, 24, 28, 32,
 42, 50, 358, 372, 571, 576, 581, 633, 646
 __pdfmeta_standard_verify_-
 handler_annot_action_A:n . 82, 82
 __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, 76
\__pdfmeta_standard_verify_-
  handler_outputintent_subtype:nn
  ..... 88, 88
\l__pdfmeta_tmpa_seq .....
  15, 832, 833, 839, 840, 1382, 1383,
  1386, 1387, 1390, 1391, 1500, 1502
\g__pdfmeta_tmpa_str .....
  ..... 18, 815, 816, 817, 818, 819, 821
\l__pdfmeta_tmpa_str .....
  ..... 15, 496, 498, 884,
  885, 894, 895, 904, 905, 1449, 1450,
  1454, 1457, 1458, 1459, 1463, 1466
\l__pdfmeta_tmpa_tl .....
  ..... 15, 343, 344, 358,
  363, 372, 374, 389, 494, 496, 814,
  815, 914, 917, 919, 948, 949, 954,
  959, 960, 963, 1084, 1382, 1384,
  1386, 1388, 1390, 1403, 1406, 1408,
  1484, 1486, 1500, 1581, 1584, 1588,
  1591, 1620, 1623, 1683, 1694, 1698
\l__pdfmeta_tmpb_seq ..... 15
\l__pdfmeta_tmpb_tl . 15, 540, 541,
  543, 548, 553, 948, 952, 954, 959,
  963, 1581, 1585, 1588, 1592, 1692, 1694
\__pdfmeta_verify_pdfa_annot_-
  flags: ..... 94, 109
\__pdfmeta_write_outputintent:nn
  ..... 467, 482, 515, 547
\__pdfmeta_xmp_add_packet_-
  chunk:n . 845, 845, 852, 863, 870,
  877, 885, 905, 982, 1014, 1016, 1662
\__pdfmeta_xmp_add_packet_-
  chunk:nN ..... 853, 853, 860, 895
\__pdfmeta_xmp_add_packet_-
  close:nn .....
  .... 874, 874, 934, 935, 966, 967,
  995, 996, 1010, 1011, 1012, 1071,
  1072, 1074, 1094, 1109, 1300, 1301,
  1302, 1303, 1304, 1340, 1341, 1342,
  1356, 1357, 1358, 1359, 1360, 1422,
  1423, 1435, 1436, 1438, 1570, 1690
\__pdfmeta_xmp_add_packet_-
  field:nnn 1102, 1102, 1284, 1286,
  1288, 1290, 1292, 1294, 1296, 1298,
  1332, 1334, 1336, 1338, 1352, 1354
\__pdfmeta_xmp_add_packet_-
  line:nnn ... 879, 879, 888, 919,
  931, 1065, 1066, 1067, 1090, 1091,
  1092, 1093, 1106, 1107, 1108, 1276,
  1277, 1279, 1280, 1324, 1325, 1327,
  1328, 1344, 1345, 1347, 1348, 1370,
  1383, 1387, 1391, 1395, 1396, 1399,
  1400, 1401, 1405, 1407, 1429, 1442,
  1444, 1456, 1465, 1467, 1469, 1498,
  1503, 1513, 1517, 1576, 1593, 1596,
  1599, 1602, 1605, 1608, 1611, 1614,
  1617, 1621, 1686, 1687, 1688, 1689
\__pdfmeta_xmp_add_packet_-
  line:nnnN .. 889, 889, 898, 1533,
  1537, 1541, 1545, 1549, 1553, 1557
\__pdfmeta_xmp_add_packet_line_-
  attr:nnnn ..... 899,
  899, 908, 951, 953, 962, 1582, 1589
\__pdfmeta_xmp_add_packet_line_-
  default:nnnn ..... 909,
  909, 921, 1366, 1374, 1378, 1504
\__pdfmeta_xmp_add_packet_-
  list:nnnn 939, 970, 1478, 1493, 1508
\__pdfmeta_xmp_add_packet_list_-
  simple:nnnn ... 922, 938, 1474,
  1480, 1486, 1489, 1491, 1496, 1501
\__pdfmeta_xmp_add_packet_-
  open:nn ..... 861, 861, 866,
  927, 928, 944, 945, 984, 985, 989,
  990, 1068, 1069, 1089, 1273, 1274,
  1282, 1283, 1321, 1322, 1330, 1331,
  1350, 1351, 1416, 1417, 1432, 1433
\__pdfmeta_xmp_add_packet_open_-
  attr:nnn .....
  .. 867, 867, 873, 987, 1064, 1105,
  1275, 1323, 1343, 1428, 1567, 1685
\__pdfmeta_xmp_add_packet_pdfid: . 591,
  596, 601, 606, 611, 616, 621, 626, 1153
\g__pdfmeta_xmp_bool .....
  ..... 559, 660, 661, 1635
\__pdfmeta_xmp_build_dc: .....
  ..... 1002, 1472, 1472
\__pdfmeta_xmp_build_iptc: ....
  ..... 1007, 1563, 1563
\__pdfmeta_xmp_build_iptc_data:N
  ..... 977, 1529, 1529
\__pdfmeta_xmp_build_packet: ...
  ..... 971, 971, 1646
\__pdfmeta_xmp_build_pdf: .....
  ..... 998, 1364, 1364
\__pdfmeta_xmp_build_pdfd: ....
  ..... 1001, 1412, 1412
\__pdfmeta_xmp_build_pdfd_-
  claim:nn ..... 1420, 1426, 1426
\__pdfmeta_xmp_build_photoshop:
  ..... 1003, 1440, 1440
\__pdfmeta_xmp_build_prism: ....
  ..... 1006, 1574, 1574

```

_pdfmeta_xmp_build_standards:	_pdfmeta_xmp_property_new:nnnnn
..... 1000, 1393 , 1393 1078 , 1079 ,
_pdfmeta_xmp_build_user:	1115, 1125, 1131, 1141, 1147, 1160,
..... 1008, 1626 , 1626	1171, 1177, 1183, 1189, 1195, 1201,
_pdfmeta_xmp_build_xmp:	1207, 1213, 1219, 1225, 1231, 1237,
..... 1004, 1372 , 1372	1243, 1249, 1255, 1265, 1313, 1672
_pdfmeta_xmp_build_xmpMM:	_pdfmeta_xmp_sanitize:nN
..... 1005, 1447 , 1447 808 , 808, 824, 884, 894, 904
_pdfmeta_xmp_build_xmpRights:	_pdfmeta_xmp_schema_enable_-
..... 999, 1511 , 1511	pdfd: 1306 , 1362 , 1675 , 1682
_pdfmeta_xmp_create_uuid:nN ..	_pdfmeta_xmp_schema_new:nnn ..
..... 796 , 796, 1452, 1461 1051 , 1051, 1111, 1121,
\l_pdfmeta_xmp_currentdate_seq	1137 , 1156 , 1167 , 1261 , 1309 , 1671
..... 781 , 789, 1644	\g_pdfmeta_xmp_schema_property_-
\l_pdfmeta_xmp_currentdate_tl .	prop 1078 , 1084 , 1086
.... 781 , 790, 1462, 1639, 1642, 1644	\l_pdfmeta_xmp_schema_seq
_pdfmeta_xmp_date_get:nNN 980 , 991 , 1050 , 1059
.... 783 , 783, 1381, 1385, 1389, 1500	\g_pdfmeta_xmp_user_packet_str 1625
\l_pdfmeta_xmp_date_regex 745 , 750	\g_pdfmeta_xmp_user_packet_tl .
_pdfmeta_xmp_date_split:nN 1625 , 1629 , 1660
..... 748 , 748, 752, 793, 1644	_pdfmeta_xmp_wtpdf_accessibility_-
_pdfmeta_xmp_decr_indent:	declaration:
..... 724 , 741, 876, 1561 649 , 677 , 680 , 1702 , 1715
\l_pdfmeta_xmp_doclang_tl	_pdfmeta_xmp_wtpdf_reuse_-
..... 825 , 973, 976, 1497	declaration:
\g_pdfmeta_xmp_export_bool 650 , 671 , 674 , 1702 , 1708
..... 683 , 691, 696, 700, 1650	_pdfmeta_xmp_xmlns_new:nn
\g_pdfmeta_xmp_export_str 1020 , 1020,
..... 684 , 692, 701, 1652	1028 , 1029 , 1030 , 1031 , 1032 , 1033 ,
_pdfmeta_xmp_generate_bom: ...	1034 , 1036 , 1037 , 1038 , 1039 , 1040 ,
..... 708 , 712, 716, 983	1041 , 1042 , 1043 , 1044 , 1045 , 1046 ,
_pdfmeta_xmp_incr_indent:	1047 , 1048 , 1049 , 1155 , 1308 , 1669
..... 724 , 736, 864, 871, 1532	\g_pdfmeta_xmp_xmlns_prop
_pdfmeta_xmp_indent: 1018 , 1022 , 1667
..... 724 , 724, 849, 857	\g_pdfmeta_xmp_xmlns_tl
_pdfmeta_xmp_indent:n 724 , 730, 1025 988 , 1018 , 1023
\l_pdfmeta_xmp_indent_int . 723 ,	pdfmetatmpa internal commands:
727 , 738 , 743 , 1013 , 1015 , 1628 , 1630	\g_pdfmetatmpa_str 15
\l_pdfmeta_xmp_iptc_data_tl ...	pdfuaid~(schema) 1137
..... 977 , 978, 1528 , 1565, 1569	PDFversion 1364
_pdfmeta_xmp_iso_today:	pdfxid~(schema) 1153
..... 1702 , 1713 , 1720	photoshop commands:
_pdfmeta_xmp_lang_get:nNN	photoshop:AuthorsPosition/pdfauthortitle
..... 829 , 843, 948, 959, 1579, 1586 1472
\l_pdfmeta_xmp_lang_regex 827 , 832	photoshop:CaptionWriter/pdfcaptionwriter
\l_pdfmeta_xmp_metalang_tl 1472
.... 825 , 835, 949, 960, 974, 975, 976	\PreviousTotalPages 1623
\g_pdfmeta_xmp_packet_tl	prg commands:
..... 844 , 847, 1569, 1648, 1653	\prg_do_nothing: 674 , 680, 1362
\g_pdfmeta_xmp_pdfd_data_prop .	\prg_new_conditional:Npnn 30
.. 1411 , 1414, 1418, 1676, 1692, 1696	\prg_new_protected_conditional:Npnn
_pdfmeta_xmp_print_date:N 40
.... 753 , 753, 1383, 1387, 1391, 1502	\prg_replicate:nn 727 , 733, 1014

		U
\tl_set:Nn	786, 814, 835, 836, 839, 840, 914, 917, 973, 974, 1620, 1683	use commands:
\tl_set_eq:NN	790, 1639	\use:N 47
\tl_to_str:N	812, 815	\use_i:nn 1406
\tl_use:N	993, 1070	\use_ii:nn 1408