

The `l3pdfmeta` module

PDF standards

L^AT_EX PDF management testphase bundle

The L^AT_EX Project*

Version 0.96n, released 2024-10-27

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 l3pdfmeta for the files it embeds.* The restriction is set for only PDF/A-1b. PDF/A-2b and PDF/A3-b lifted this restriction: PDF/A-2b allows to embed other PDF documents conforming to either PDF/A-1 or PDF/A-2, and PDF/A-3 allows any embedded files. I don't see a way to test the PDF/A-2b 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:nm {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
    X = FOGRA39L_coated.icc, % or x or longer GTS_PDFX
    ISO_PDFE1 = whatever.icc
  }
}
```

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

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 be added yet. This will be decided later.

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

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

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

1.3 Regression tests

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

`\pdfmeta_set_regression_data:` `\pdfmeta_set_regression_data:`

This sets various metadata to values needed by the L^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.

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.

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.

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 "grÃ¼Ãe". 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 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/>.

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

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:enenn
  {https://pdfa.org/declarations\c_hash_str wcag21A}{2023-11-20}{}
\pdfmeta_xmp_add_declaration:nenenn
  {https://github.com/TikZlings/no-duck-harmed}
  {Ulrike-Fischer}{2023-11-20}{Bär}{https://github.com/u-fischer/bearwear}
\pdfmeta_xmp_add_declaration:nenenn
  {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}{2024-10-27}{0.96n}  
4 {PDF-Standards---LaTeX PDF management testphase bundle}  
5 </header>
```

Message for unknown standards

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

Message for not fitting pdf version

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

```
\l__pdfmeta_tmpa_tl  
\l__pdfmeta_tmpb_tl  
\l__pdfmeta_tmpa_str  
\g__pdfmetatmpa_str  
\l__pdfmeta_tmpa_seq  
\l__pdfmeta_tmpb_seq  
10 \tl_new:N \l__pdfmeta_tmpa_tl  
11 \tl_new:N \l__pdfmeta_tmpb_tl  
12 \str_new:N \l__pdfmeta_tmpa_str  
13 \str_new:N \g__pdfmeta_tmpa_str  
14 \seq_new:N \l__pdfmeta_tmpa_seq  
15 \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  
16 \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`

```
17 \cs_new:Npn \pdfmeta_standard_item:n #1
18 {
19   \prop_item:Nn \g__pdfmeta_standard_prop {#1}
20 }
```

(End of definition for `\pdfmeta_standard_item:n`. This function is documented on page 2.)

`\pdfmeta_standard_get:nN`

```
21 \cs_new_protected:Npn \pdfmeta_standard_get:nN #1 #2
22 {
23   \prop_get:NnN \g__pdfmeta_standard_prop {#1} #2
24 }
```

(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.

```
25 \prg_new_conditional:Npnn \pdfmeta_standard_verify:n #1 {T,F,TF}
26 {
27   \prop_if_in:NnTF \g__pdfmeta_standard_prop {#1}
28   {
29     \prg_return_false:
30   }
31   {
32     \prg_return_true:
33   }
34 }
```

(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.

```
35 \prg_new_protected_conditional:Npnn \pdfmeta_standard_verify:nn #1 #2 {T,F,TF}
36 {
37   \prop_if_in:NnTF \g__pdfmeta_standard_prop {#1}
38   {
39     \cs_if_exist:cTF {__pdfmeta_standard_verify_handler_#1:nn}
40     {
41       \exp_args:Nnne
42       \use:c
43       {__pdfmeta_standard_verify_handler_#1:nn}
44       { #2 }
45       { \prop_item:Nn \g__pdfmeta_standard_prop {#1} }
46     }
47     {
48       \prg_return_false:
49     }
50   }
51   {
52     \prg_return_true:
53   }
54 }
```

(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`

```
55 %
56 \cs_new_protected:Npn \__pdfmeta_standard_verify_handler_min_pdf_version:nn #1 #2
57 {
58   \pdf_version_compare:NnTF <
59     { #2 }
60     {\prg_return_false:}
61     {\prg_return_true:}
62 }
```

(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 too high

`_standard_verify_handler_max_pdf_version:nn`

```
63 %
64 \cs_new_protected:Npn \__pdfmeta_standard_verify_handler_max_pdf_version:nn #1 #2
65 {
66   \pdf_version_compare:NnTF >
67     { #2 }
68     {\prg_return_false:}
69     {\prg_return_true:}
70 }
```

(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`

```
71
72 \cs_new_protected:Npn \__pdfmeta_standard_verify_handler_named_actions:nn #1 #2
73 {
74   \tl_if_in:nnTF { #2 } { #1 }
75     {\prg_return_true:}
76     {\prg_return_false:}
77 }
```

(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`

```
78 \cs_new_protected:Npn \__pdfmeta_standard_verify_handler_annot_action_A:nn #1 #2
79 {
80   \tl_if_in:nnTF { #2 } { #1 }
81     {\prg_return_true:}
82     {\prg_return_false:}
83 }
```

(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

```
84 \cs_new_protected:Npn \__pdfmeta_standard_verify_handler_outputintent_subtype:nn #1 #2
85 {
86   \tl_if_eq:nnTF { #2 }{ #1 }
87     {\prg_return_true:}
88     {\prg_return_false:}
89 }
```

(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:

```
90 \cs_new_protected:Npn \__pdfmeta_verify_pdfa_annot_flags:
91 {
92   \bitset_set_true:Nn \l_pdfannot_F_bitset {Print}
93   \bitset_set_false:Nn \l_pdfannot_F_bitset {Hidden}
94   \bitset_set_false:Nn \l_pdfannot_F_bitset {Invisible}
95   \bitset_set_false:Nn \l_pdfannot_F_bitset {NoView}
96   \pdfannot_dict_put:nnn {link/URI}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
97   \pdfannot_dict_put:nnn {link/GoTo}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
98   \pdfannot_dict_put:nnn {link/GoToR}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
99   \pdfannot_dict_put:nnn {link/Launch}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
100  \pdfannot_dict_put:nnn {link/Named}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
101 }
```

At begin document this should be checked:

```
102 \hook_gput_code:nnn {begindocument} {pdf}
103 {
104   \pdfmeta_standard_verify:nF { annot_flags }
105   { \__pdfmeta_verify_pdfa_annot_flags: }
106   \pdfmeta_standard_verify:nF { Trailer_no_Info }
107   { \__pdf_backend_omit_info:n {1} }
108   \pdfmeta_standard_verify:nF { no_CharSet }
109   { \__pdf_backend_omit_charset:n {1} }
110   \pdfmeta_standard_verify:nF { omit_CID }
111   { \__pdf_backend_omit_cidset:n {1} }
112   \pdfmeta_standard_verify:nnF { min_pdf_version }
113   { \pdf_version: }
114   { \msg_warning:nneee {pdf}{wrong-pdfversion}
115     {\pdf_version:}{low}
116     {
117       \pdfmeta_standard_item:n{type}
118       -
119       \pdfmeta_standard_item:n{level}
120     }
121   }
122   \pdfmeta_standard_verify:nnF { max_pdf_version }
123   { \pdf_version: }
124   { \msg_warning:nneee {pdf}{wrong-pdfversion}
125     {\pdf_version:}{high}
```

```

126     {
127     \pdfmeta_standard_item:n{type}
128     -
129     \pdfmeta_standard_item:n{level}
130     }
131 }
132 }

```

3.1.4 pdf/A

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

```

\g__pdfmeta_standard_pdf/A-1B_prop
\g__pdfmeta_standard_pdf/A-2A_prop
\g__pdfmeta_standard_pdf/A-2B_prop
\g__pdfmeta_standard_pdf/A-2U_prop
\g__pdfmeta_standard_pdf/A-3A_prop
\g__pdfmeta_standard_pdf/A-3B_prop
\g__pdfmeta_standard_pdf/A-3U_prop
\g__pdfmeta_standard_pdf/A-4_prop
133 \prop_new:c { g__pdfmeta_standard_pdf/A-1B_prop }
134 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/A-1B_prop }
135 {
136   ,name           = pdf/A-1B
137   ,type           = A
138   ,level          = 1
139   ,conformance   = B
140   ,year           = 2005
141   ,min_pdf_version = 1.4      %minimum
142   ,max_pdf_version = 1.4      %minimum
143   ,no_encryption  =
144   ,no_external_content = % no F, FFilter, or FDecodeParms in stream dicts
145   ,no_embed_content = % no EF key in filespec, no /Type/EmbeddedFiles
146   ,max_string_size = 65535
147   ,max_array_size = 8191
148   ,max_dict_size  = 4095
149   ,max_obj_num    = 8388607
150   ,max_nest_qQ    = 28
151   ,named_actions  = {NextPage, PrevPage, FirstPage, LastPage}
152   ,annot_flags    =
153   %booleans. Only the existence of the key matter.
154   %If the entry is added it means a requirements is there
155   %(in most cases "don't use ...")
156   %
157   %=====
158   % Rule 6.1.13-1 CosDocument, isOptionalContentPresent == false
159   ,Catalog_no_OCProperties =
160   % Rule 6.9-4 The AS key shall not appear in any optional content configuration dictionary
161   % actually only starting with A-2 but doesn't harm here either
162   ,Catalog_OCProperties_no_AS=
163   %=====
164   % Rule 6.6.1-1: PDAction, S == "GoTo" || S == "GoToR" || S == "Thread"
165   %           || S == "URI" || S == "Named" || S == "SubmitForm"
166   % means: no /S/Launch, /S/Sound, /S/Movie, /S/ResetForm, /S/ImportData,
167   %           /S/JavaScript, /S/Hide
168   ,annot_action_A = {GoTo,GoToR,Thread,URI,Named,SubmitForm}
169   %=====
170   % Rule 6.6.2-1: PDAnnot, Subtype != "Widget" || AA_size == 0
171   % means: no AA dictionary
172   ,annot_widget_no_AA =

```

```

173 %=====
174 % Rule 6.9-2: PDAnnot, Subtype != "Widget" || (A_size == 0 && AA_size == 0)
175 % (looks like a tightening of the previous rule)
176 % ,annot_widget_no_A_AA =
177 %=====
178 % Rule 6.9-1 PDAcroForm, NeedAppearances == null || NeedAppearances == false
179 % ,form_no_NeedAppearances =
180 %=====
181 %Rule 6.9-3 PDFormField, AA_size == 0
182 % ,form_no_AA =
183 %=====
184 % to be continued https://docs.verapdf.org/validation/pdfa-part1/
185 % - Outputintent/colorprofiles requirements
186 % an outputintent should be loaded and is unique.
187 % ,outputintent_A = {GTS_PDFa1}
188 % - no Alternates key in image dictionaries
189 % - no OPI, Ref, Subtype2 with PS key in xobjects
190 % - Interpolate = false in images
191 % - no TR, TR2 in ExtGstate
192 }
193
194 %A-2b =====
195 \prop_new:c { g__pdfmeta_standard_pdf/A-2B_prop }
196 \prop_gset_eq:cc
197 { g__pdfmeta_standard_pdf/A-2B_prop }
198 { g__pdfmeta_standard_pdf/A-1B_prop }
199 \prop_gput:cnn
200 { g__pdfmeta_standard_pdf/A-2B_prop }{name}{pdf/A-2B}
201 \prop_gput:cnn
202 { g__pdfmeta_standard_pdf/A-2B_prop }{year}{2011}
203 \prop_gput:cnn
204 { g__pdfmeta_standard_pdf/A-2B_prop }{level}{2}
205 % embedding files is allowed (with restrictions)
206 \prop_gremove:cn
207 { g__pdfmeta_standard_pdf/A-2B_prop }
208 { embed_content}
209 \prop_gput:cnn
210 { g__pdfmeta_standard_pdf/A-2B_prop }{max_pdf_version}{1.7}
211 \prop_gput:cnn
212 { g__pdfmeta_standard_pdf/A-2B_prop }{omit_CID}{}
213 % OCG layers are allowed (with restrictions)
214 \prop_gremove:cn
215 { g__pdfmeta_standard_pdf/A-2B_prop }
216 { Catalog_no_OCProperties }
217
218 %A-2u =====
219 \prop_new:c { g__pdfmeta_standard_pdf/A-2U_prop }
220 \prop_gset_eq:cc
221 { g__pdfmeta_standard_pdf/A-2U_prop }
222 { g__pdfmeta_standard_pdf/A-2B_prop }
223 \prop_gput:cnn
224 { g__pdfmeta_standard_pdf/A-2U_prop }{name}{pdf/A-2U}
225 \prop_gput:cnn
226 { g__pdfmeta_standard_pdf/A-2U_prop }{conformance}{U}

```

```

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

```



```

281
282 %A-4 =====
283 \prop_new:c { g__pdfmeta_standard_pdf/A-4_prop }
284 \prop_gset_eq:cc
285   { g__pdfmeta_standard_pdf/A-4_prop }
286   { g__pdfmeta_standard_pdf/A-3U_prop }
287 \prop_gput:cnn
288   { g__pdfmeta_standard_pdf/A-4_prop }{name}{pdf/A-4}
289 \prop_gput:cnn
290   { g__pdfmeta_standard_pdf/A-4_prop }{level}{4}
291 \prop_gput:cnn
292   { g__pdfmeta_standard_pdf/A-4_prop }{min_pdf_version}{2.0}
293 \prop_gput:cnn
294   { g__pdfmeta_standard_pdf/A-4_prop }{year}{2020}
295 \prop_gput:cnn
296   { g__pdfmeta_standard_pdf/A-4_prop }{no_CharSet}{}
297 \prop_gput:cnn
298   { g__pdfmeta_standard_pdf/A-4_prop }{Trailer_no_Info}{}
299 \prop_gremove:cn
300   { g__pdfmeta_standard_pdf/A-4_prop }{conformance}
301 \prop_gremove:cn
302   { g__pdfmeta_standard_pdf/A-4_prop }{max_pdf_version}
303 \prop_gremove:cn
304   { g__pdfmeta_standard_pdf/A-4_prop }{Catalog_OCProperties_no_AS}
305 %A-4f =====
306 \prop_new:c { g__pdfmeta_standard_pdf/A-4F_prop }
307 \prop_gset_eq:cc
308   { g__pdfmeta_standard_pdf/A-4F_prop }
309   { g__pdfmeta_standard_pdf/A-4_prop }
310 \prop_gput:cnn
311   { g__pdfmeta_standard_pdf/A-4F_prop }{conformance}{F}
312 % containsEmbeddedFiles == true ISO 19005-4:2020, Clause: 6.9, Test number: 5
313 \prop_gput:cnn
314   { g__pdfmeta_standard_pdf/A-4F_prop }{Catalog_EmbeddedFiles}{}

```

(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

```

315 \AddToHook{begindocument/end}
316 {
317   \pdfmeta_standard_verify:nF{Catalog_EmbeddedFiles}
318   {
319     \tl_gput_right:Nn\g__kernel_pdfmanagement_end_run_code_tl
320     {
321       \bool_if:NT \g__pdfmanagement_active_bool
322       {
323         \pdfdict_if_empty:nT { g__pdf_Core/Catalog/Names/EmbeddedFiles }
324         {
325           \group_begin:
326           \pdfdict_put:nne {l_pdffile/Filespec} {Desc}{(note-about~PDF/A-4F)}

```

```

327         \pdfdict_put:nnn { l_pdffile/Filespec }{AFRelationship} { /Unspecified }
328         \pdffile_embed_stream:nnN {PDF~standard~A~4F~requires~a~file}{readme.txt}\l__pdf
329         \exp_args:Nne \__pdf_backend_Names_gpush:nn{EmbeddedFiles}{(readme)~\l__pdfmeta_
330         \group_end:
331     }
332 }
333 }
334 }
335 }

```

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 only color profile.

```
336 \prop_new:N \g__pdfmeta_outputintents_prop
```

(End of definition for `\g__pdfmeta_outputintents_prop`.)

Some keys to fill the property.

```

337 \keys_define:nn { document / metadata }
338 {
339     colorprofiles .code:n =
340     {
341         \keys_set:nn { document / metadata / colorprofiles }{#1}
342     }
343 }
344 \keys_define:nn { document / metadata / colorprofiles }

```

```

345 {
346   ,A .code:n =
347     {
348       \tl_if_blank:nF {#1}
349       {
350         \prop_gput:Nnn \g__pdfmeta_outputintents_prop
351         { GTS_PDFA1 } {#1}
352       }
353     }
354   ,a .code:n =
355     {
356       \tl_if_blank:nF {#1}
357       {
358         \prop_gput:Nnn \g__pdfmeta_outputintents_prop
359         { GTS_PDFA1 } {#1}
360       }
361     }
362   ,X .code:n =
363     {
364       \tl_if_blank:nF {#1}
365       {
366         \prop_gput:Nnn \g__pdfmeta_outputintents_prop
367         { GTS_PDFX } {#1}
368       }
369     }
370   ,x .code:n =
371     {
372       \tl_if_blank:nF {#1}
373       {
374         \prop_gput:Nnn \g__pdfmeta_outputintents_prop
375         { GTS_PDFX } {#1}
376       }
377     }
378   ,unknown .code:n =
379     {
380       \tl_if_blank:nF {#1}
381       {
382         \exp_args:NNo
383         \prop_gput:Nnn \g__pdfmeta_outputintents_prop
384         { \l_keys_key_str } {#1}
385       }
386     }
387 }

```

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

```

388 \pdfdict_new:n {l_pdfmeta/outputintent}
389 \pdfdict_put:nnn {l_pdfmeta/outputintent}
390 {Type}{/OutputIntent}
391 \prop_const_from_keyval:cn { c__pdfmeta_colorprofile_sRGB.icc}
392 {
393   ,OutputConditionIdentifier=IEC~sRGB
394   ,Info=IEC-61966-2.1-Default~RGB~colour~space~~sRGB
395   ,RegistryName=http://www.iec.ch

```

```

396     ,N = 3
397   }
398   \prop_const_from_keyval:cn { c__pdfmeta_colorprofile_FOGRA39L_coated.icc}
399   {
400     ,OutputConditionIdentifier=FOGRA39L~Coated
401     ,Info={Offset~printing,~according~to~ISO~12647~2:2004/Amd~1,~OFCOM,~ %
402           paper~type~1~or~2~==~coated~art,~115~g/m2,~tone~value~increase~
403           curves~A~(CMY)~and~B~(K)}
404     ,RegistryName=http://www.fogra.org
405     ,N = 4
406   }

```

`__pdfmeta_embed_colorprofile:n`
`__pdfmeta_write_outputintent:nn`

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

```

407   \cs_new_protected:Npn \__pdfmeta_embed_colorprofile:n #1#1 file name
408   {
409     \pdf_object_if_exist:nF { __color_icc_ #1 }
410     {
411       \pdf_object_new:n { __color_icc_ #1 }
412       \pdf_object_write:nne { __color_icc_ #1 } { fstream }
413       {
414         {/N\c_space_tl
415           \prop_item:cn{c__pdfmeta_colorprofile_#1}{N}
416         }
417       }
418     }
419   }
420 }
421
422   \cs_new_protected:Npn \__pdfmeta_write_outputintent:nn #1 #2 % #1 file name, #2 subtype
423   {
424     \group_begin:
425     \pdfdict_put:nne {l_pdfmeta/outputintent}{S}{/\str_convert_pdfname:n{#2}}
426     \pdfdict_put:nne {l_pdfmeta/outputintent}
427     {DestOutputProfile}
428     {\pdf_object_ref:n{ __color_icc_ #1 }}
429     \clist_map_inline:nn { OutputConditionIdentifier, Info, RegistryName }
430     {
431       \prop_get:cnNT
432       { c__pdfmeta_colorprofile_#1}
433       { ##1 }
434       \l__pdfmeta_tmpa_tl
435       {
436         \pdf_string_from_unicode:nVN {utf8/string}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_st
437         \pdfdict_put:nne
438         {l_pdfmeta/outputintent}{##1}{\l__pdfmeta_tmpa_str}
439       }
440     }
441     \pdf_object_unnamed_write:ne {dict}{\pdfdict_use:n {l_pdfmeta/outputintent} }
442     \pdfmanagement_add:nne {Catalog}{OutputIntents}{\pdf_object_ref_last:}
443     \group_end:
444   }

```

(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

```

445
446 \AddToHook{begindocument/end}
447 {
448   \pdfmeta_standard_verify:nTF {outputintent_A}
449   {
450     \prop_map_inline:Nn \g__pdfmeta_outputintents_prop
451     {
452       \prop_if_exist:cTF {c__pdfmeta_colorprofile_#2}
453       {
454         \_pdfmeta_embed_colorprofile:n
455         {#2}
456         \_pdfmeta_write_outputintent:nn
457         {#2}
458         {#1}
459       }
460       {
461         \msg_warning:nnn{pdfmeta}{colorprofile-undefined}{#2}
462       }
463     }
464   }

```

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.

```

465   {
466     \exp_args:NNe
467     \prop_if_in:NnF
468     \g__pdfmeta_outputintents_prop
469     { \pdfmeta_standard_item:n { outputintent_A } }
470     {
471       \exp_args:NNe
472       \prop_gput:Nnn
473       \g__pdfmeta_outputintents_prop
474       { \pdfmeta_standard_item:n { outputintent_A } }
475       { sRGB.icc }
476     }
477     \exp_args:NNe
478     \prop_get:NnN
479     \g__pdfmeta_outputintents_prop
480     { \pdfmeta_standard_item:n { outputintent_A } }
481     \l__pdfmeta_tmpb_tl
482     \prop_if_exist:cTF {c__pdfmeta_colorprofile_\l__pdfmeta_tmpb_tl}
483     {
484       \exp_args:NV \_pdfmeta_embed_colorprofile:n \l__pdfmeta_tmpb_tl
485       \prop_map_inline:Nn \g__pdfmeta_outputintents_prop
486       {
487         \exp_args:NV
488         \_pdfmeta_write_outputintent:nn
489         \l__pdfmeta_tmpb_tl
490         { #1 }
491       }
492     }

```

```

493     {
494     \msg_warning:nne{pdfmeta}{colorprofile-undefined}{\l__pdfmeta_tmpb_tl}
495     }
496   }
497 }

```

3.2 Regression test

This is simply a copy of the backend function.

```

498 \cs_new_protected:Npn \pdfmeta_set_regression_data:
499   { \__pdf_backend_set_regression_data: }

```

4 XMP-Metadata implementation

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

```

500 \bool_new:N\g__pdfmeta_xmp_bool
501 \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.

```

502 \hook_gput_code:nnn{pdfmanagement/add}{pdfmanagement}
503   {
504   \pdfmanagement_add:nne {Info}{Producer}{(\c_sys_engine_exec_str-\c_sys_engine_version_str)}
505   \pdfmanagement_add:nne {Info}{Creator}{(LaTeX)}
506   }

```

4.1 New document keys

```

507 \keys_define:nn { document / metadata }
508 {
509   _pdfstandard / X-4 .code:n =
510   {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-4}},
511   _pdfstandard / X-4p .code:n =
512   {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-4p}},
513   _pdfstandard / X-5g .code:n =
514   {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5g}},
515   _pdfstandard / X-5n .code:n =
516   {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5n}},
517   _pdfstandard / X-5pg .code:n =
518   {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5pg}},
519   _pdfstandard / X-6 .code:n =
520   {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6p}},
521   _pdfstandard / X-6n .code:n =
522   {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6n}},
523   _pdfstandard / X-6p .code:n =
524   {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6p}},
525   _pdfstandard / UA-1 .code:n =
526   {
527     \AddToDocumentProperties [document]{pdfstandard-UA}{{1}}{}
528     \AddToHook{begindocument/before}
529     {
530       \pdf_version_compare:NnF < {2.0}

```

```

531         {
532             \msg_warning:nneee
533             {pdf}{wrong-pdfversion}
534             {\pdf_version:}{high}{UA-1}
535         }
536     }
537 },

```

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.

```

538 _pdfstandard / UA-2 .code:n =
539 {
540     \AddToDocumentProperties [document]{pdfstandard-UA}{2}{2024}
541     \AddToHook{begindocument/before}
542     {\prop_gput:Nnn \g__pdfmeta_standard_prop {Trailer_no_Info}{}}
543     \AddToHook{begindocument/before}
544     {
545         \__pdfmeta_xmp_wtpdf_accessibility_declaration:
546         \__pdfmeta_xmp_wtpdf_reuse_declaration:
547         \pdf_version_compare:NnT < {2.0}
548         {
549             \msg_warning:nneee
550             {pdf}{wrong-pdfversion}
551             {\pdf_version:}{low}{UA-2}
552         }
553     }
554 },
555 xmp .choice:,
556 xmp / true .code:n = { \bool_gset_true:N \g__pdfmeta_xmp_bool },
557 xmp / false .code:n = { \bool_gset_false:N \g__pdfmeta_xmp_bool},
558 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.

```

559 xmp / wtpdf .code:n =
560 {
561     \keys_set:nn {__pdfmeta/xmp}{#1}
562 },
563 }
564 \keys_define:nn {__pdfmeta/xmp}
565 {
566     reuse .choice:,
567     reuse / true .code:n = \__pdfmeta_xmp_wtpdf_reuse_declaration:,
568     reuse / false .code:n =
569     {
570         \cs_set_eq:NN \__pdfmeta_xmp_wtpdf_reuse_declaration: \prg_do_nothing:
571     },
572     accessibility .choice:,
573     accessibility / true .code:n = \__pdfmeta_xmp_wtpdf_accessibility_declaration:,
574     accessibility /false .code:n =
575     {
576         \cs_set_eq:NN \__pdfmeta_xmp_wtpdf_accessibility_declaration: \prg_do_nothing:
577     },
578 }

```

XMP debugging option

```
579 \bool_new:N \g__pdfmeta_xmp_export_bool
580 \str_new:N \g__pdfmeta_xmp_export_str
581
582 \keys_define:nn { document / metadata }
583 {
584   ,debug / xmp-export .choice:
585   ,debug / xmp-export / true .code:n=
586   {
587     \bool_gset_true:N \g__pdfmeta_xmp_export_bool
588     \str_gset_eq:NN \g__pdfmeta_xmp_export_str \c_sys_jobname_str
589   }
590   ,debug / xmp-export / false .code:n =
591   {
592     \bool_gset_false:N \g__pdfmeta_xmp_export_bool
593   }
594   ,debug / xmp-export / unknown .code:n =
595   {
596     \bool_gset_true:N \g__pdfmeta_xmp_export_bool
597     \str_gset:Nn \g__pdfmeta_xmp_export_str { #1 }
598   }
599   ,debug / xmp-export .default:n = true
600 }
```

4.2 Messages

```
601 \msg_new:nnn{pdfmeta}{xmp-defined}{The~XMP~#1~'#2'~is~already~declared}
602 \msg_new:nnn{pdfmeta}{xmp-undefined}{The~XMP~#1~'#2'~is~undefined}
603 \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:

```
604 \bool_lazy_or:nnTF
605 { \sys_if_engine luatex_p: }
606 { \sys_if_engine xetex_p: }
607 {
608   \cs_new:Npn \_pdfmeta_xmp_generate_bom:
609     { \char_generate:nn {"FEFF"}{12} }
610 }
611 {
612   \cs_new:Npn \_pdfmeta_xmp_generate_bom:
613     {
614       \char_generate:nn {"EF"}{12}
615       \char_generate:nn {"BB"}{12}
616       \char_generate:nn {"BF"}{12}
617     }
618 }
```

(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
    619 \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:
    620 \cs_new:Npn \__pdfmeta_xmp_indent:
    621 {
    622     \iow_newline:
    623     \prg_replicate:nn {\l__pdfmeta_xmp_indent_int}{\c_space_tl}
    624 }
    625
    626 \cs_new:Npn \__pdfmeta_xmp_indent:n #1
    627 {
    628     \iow_newline:
    629     \prg_replicate:nn {#1}{\c_space_tl}
    630 }
    631
    632 \cs_new_protected:Npn \__pdfmeta_xmp_incr_indent:
    633 {
    634     \int_incr:N \l__pdfmeta_xmp_indent_int
    635 }
    636
    637 \cs_new_protected:Npn \__pdfmeta_xmp_decr_indent:
    638 {
    639     \int_decr:N \l__pdfmeta_xmp_indent_int
    640 }
    (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 extend the regex can also handle incomplete dates.

```
\l__pdfmeta_xmp_date_regex
    641 \regex_new:N \l__pdfmeta_xmp_date_regex
    642 \regex_set:Nn \l__pdfmeta_xmp_date_regex
    643 {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.
    644 \cs_new_protected:Npn \__pdfmeta_xmp_date_split:nN #1 #2 %#1 date, #2 seq
    645 {
    646     \regex_split:NnN \l__pdfmeta_xmp_date_regex {#1} #2
    647 }
    648 \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.

```
649 \cs_new:Npn\_pdfmeta_xmp_print_date:N #1 % seq
650 {
651   \tl_if_blank:eTF { \seq_item:Nn #1 {1} }
652   {
653     \seq_item:Nn #1 {2} %year
654     -
655     \seq_item:Nn #1 {3} %month
656     -
657     \seq_item:Nn #1 {4} % day
658     \tl_if_blank:eF
659     { \seq_item:Nn #1 {5} }
660     { T \seq_item:Nn #1 {5} } %hour
661     \tl_if_blank:eF
662     { \seq_item:Nn #1 {6} }
663     { : \seq_item:Nn #1 {6} } %minutes
664     \tl_if_blank:eF
665     { \seq_item:Nn #1 {7} }
666     { : \seq_item:Nn #1 {7} } %seconds
667     \seq_item:Nn #1 {8} %Z,+,-
668     \seq_item:Nn #1 {9}
669     \tl_if_blank:eF
670     { \seq_item:Nn #1 {10} }
671     { : \seq_item:Nn #1 {10} }
672   }
673   {
674     \seq_item:Nn #1 {1}
675   }
676 }
```

(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.

```
677 \tl_new:N \l_pdfmeta_xmp_currentdate_tl
678 \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.

```
679 \cs_new_protected:Npn\_pdfmeta_xmp_date_get:nNN #1 #2 #3
680 %#1 property, #2 tl var with PDF date, #3 seq for split date
681 {
682   \tl_set:Ne #2 { \GetDocumentProperties{#1} }
683   \tl_if_blank:VTF #2
684   {
685     \seq_set_eq:NN #3 \l_pdfmeta_xmp_currentdate_seq
686     \tl_set_eq:NN #2 \l_pdfmeta_xmp_currentdate_tl
687   }
688   {
689     \_pdfmeta_xmp_date_split:VN #2 #3
690   }
691 }
```

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

4.3.4 UUID

We need a command to generate an uuid

`__pdfmeta_xmp_create_uuid:nN`

```
692 \cs_new_protected:Npn \__pdfmeta_xmp_create_uuid:nN #1 #2
693 {
694   \str_set:N#2 {\str_lowercase:f{\tex_mdffivesum:D{#1}}}
695   \str_set:N#2
696     { uuid:
697       \str_range:Nnn #2{1}{8}
698       -\str_range:Nnn#2{9}{12}
699       -4\str_range:Nnn#2{13}{15}
700       -8\str_range:Nnn#2{16}{18}
701       -\str_range:Nnn#2{19}{30}
702     }
703 }
```

(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.

```
704 \cs_new_protected:Npn \__pdfmeta_xmp_sanitize:nN #1 #2
705   %#1 input string, #2 str with the output
706 {
707   \group_begin:
708   \text_declare_purify_equivalent:Nn \& {\tl_to_str:N & }
709   \text_declare_purify_equivalent:Nn \texttilde {\c_tilde_str}
710   \tl_set:N#1 \l__pdfmeta_tmpa_tl { \text_purify:n {#1} }
711   \str_gset:N#2 \g__pdfmeta_tmpa_str { \tl_to_str:N \l__pdfmeta_tmpa_tl }
712   \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {\&}{&}
713   \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {<}{&lt;}
714   \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {>}{&gt;}
715   \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {"}{&quot;}
716   \group_end:
717   \str_set_eq:NN #2 \g__pdfmeta_tmpa_str
718 }
719
720 \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`

```
721 \tl_new:N \l__pdfmeta_xmp_doclang_tl
722 \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`

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

(End of definition for \l__pdfmeta_xmp_lang_regex.)

725 \cs_new_protected:Npn \__pdfmeta_xmp_lang_get:nNN #1 #2 #3
726 % #1 text, #2 tl var for lang match (or default), #3 tl var for text
727 {
728   \regex_extract_once:NnN \l__pdfmeta_xmp_lang_regex {#1}\l__pdfmeta_tmpa_seq
729   \seq_if_empty:NTF \l__pdfmeta_tmpa_seq
730   {
731     \tl_set:Nn #2 \l__pdfmeta_xmp_metalang_tl
732     \tl_set:Nn #3 {#1}
733   }
734   {
735     \tl_set:Ne #2 {\seq_item:Nn\l__pdfmeta_tmpa_seq{2}}
736     \tl_set:Ne #3 {\seq_item:Nn\l__pdfmeta_tmpa_seq{3}}
737   }
738 }
739 \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`

```
740 \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.

```
741 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_chunk:n #1
742 {
743   \tl_gput_right:Ne\g__pdfmeta_xmp_packet_tl
744   {
745     \__pdfmeta_xmp_indent: \exp_not:n{#1}
746   }
747 }
748 \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.

```
749 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_chunk:nN #1 #2
750 {
751   \tl_put_right:Ne#2
```

```

752     {
753       \__pdfmeta_xmp_indent: \exp_not:n{#1}
754     }
755   }
756 \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.

```

757 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_open:nn #1 #2 %1 prefix #2 name
758   {
759     \__pdfmeta_xmp_add_packet_chunk:n {<#1:#2>}
760     \__pdfmeta_xmp_incr_indent:
761   }
762 \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.

```

763 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_open_attr:nnn #1 #2 #3
764   %1 prefix #2 name #3 attr
765   {
766     \__pdfmeta_xmp_add_packet_chunk:n {<#1:#2~#3>}
767     \__pdfmeta_xmp_incr_indent:
768   }
769 \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.

```

770 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_close:nn #1 #2 %1 prefix #2:name
771   {
772     \__pdfmeta_xmp_decr_indent:
773     \__pdfmeta_xmp_add_packet_chunk:n {</#1:#2>}
774   }

```

(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.

```

775 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_line:nnn #1 #2 #3
776   %1 prefix #2 name #3 content
777   {
778     \t1_if_blank:nF {#3}
779     {
780       \__pdfmeta_xmp_sanitizize:nN {#3}\l__pdfmeta_tmpa_str
781       \__pdfmeta_xmp_add_packet_chunk:e {<#1:#2>\l__pdfmeta_tmpa_str</#1:#2>}
782     }
783   }
784 \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.

```

785 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_line:nnnN #1 #2 #3 #4
786   % #1 prefix #2 name #3 content #4 tl_var to prebuilt.
787   {
788     \tl_if_blank:nF {#3}
789     {
790       \_pdfmeta_xmp_sanitizе:nN {#3}\l__pdfmeta_tmpa_str
791       \_pdfmeta_xmp_add_packet_chunk:eN {<#1:#2>\l__pdfmeta_tmpa_str</#1:#2>} #4
792     }
793   }
794 \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

```

795 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_line_attr:nnnn #1 #2 #3 #4
796   % #1 prefix #2 name #3 attribute #4 content
797   {
798     \tl_if_blank:nF {#4}
799     {
800       \_pdfmeta_xmp_sanitizе:nN {#4}\l__pdfmeta_tmpa_str
801       \_pdfmeta_xmp_add_packet_chunk:e {<#1:#2-#3>\l__pdfmeta_tmpa_str</#1:#2>}
802     }
803   }
804 \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`

```

805 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_line_default:nnnn #1 #2 #3 #4
806   % #1 prefix #2 name #3 default #4 content
807   {
808     \tl_if_blank:nTF { #4 }
809     {
810       \tl_set:Nn \l__pdfmeta_tmpa_tl {#3}
811     }
812     {
813       \tl_set:Nn \l__pdfmeta_tmpa_tl {#4}
814     }
815     \_pdfmeta_xmp_add_packet_line:nnV {#1}{#2}\l__pdfmeta_tmpa_tl
816   }
817 \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:

```

818 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_list_simple:nnnn #1 #2 #3 #4
819   % #1 prefix, #2 name, #3 type (Seq/Bag/Alt) #4 a clist
820   {
821     \clist_if_empty:nF { #4 }
822     {
823       \_pdfmeta_xmp_add_packet_open:nn {#1}{#2}

```

```

824     \_pdfmeta_xmp_add_packet_open:nn {rdf}{#3}
825     \clist_map_inline:nn {#4}
826     {
827         \_pdfmeta_xmp_add_packet_line:nnn
828         {rdf}{li}{##1}
829     }
830     \_pdfmeta_xmp_add_packet_close:nn{rdf}{#3}
831     \_pdfmeta_xmp_add_packet_close:nn {#1}{#2}
832 }
833 }
834 \cs_generate_variant:Nn \_pdfmeta_xmp_add_packet_list_simple:nnnn {nnnV,nnne}

```

Here we check also for the language.

```

835 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_list:nnnn #1 #2 #3 #4
836   %#1 prefix, #2 name, #3 type (Seq/Bag/Alt) #4 a clist
837   {
838     \clist_if_empty:nF { #4 }
839     {
840         \_pdfmeta_xmp_add_packet_open:nn {#1}{#2}
841         \_pdfmeta_xmp_add_packet_open:nn {rdf}{#3}
842         \clist_map_inline:nn {#4}
843         {
844             \_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!

```

845             \tl_if_eq:eeTF{\l__pdfmeta_tmpa_tl}{\l__pdfmeta_xmp_metalang_tl}
846             {
847                 \_pdfmeta_xmp_add_packet_line_attr:nneV
848                 {rdf}{li}{xml:lang="x-default" }\l__pdfmeta_tmpb_tl
849             }
850             {
851                 \_pdfmeta_xmp_add_packet_line_attr:nneV
852                 {rdf}{li}{xml:lang="\l__pdfmeta_tmpa_tl" }\l__pdfmeta_tmpb_tl
853             }
854         }
855         \_pdfmeta_xmp_add_packet_close:nn{rdf}{#3}
856         \_pdfmeta_xmp_add_packet_close:nn {#1}{#2}
857     }
858 }
859 \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.

```

860 \cs_new_protected:Npn \_pdfmeta_xmp_build_packet:
861   {

```

Get the main languages

```

862   \tl_set:Ne \l__pdfmeta_xmp_doclang_tl  {\GetDocumentProperties{document/lang}}
863   \tl_set:Ne \l__pdfmeta_xmp_metalang_tl  {\GetDocumentProperties{hyperref/pdfmetalang}}
864   \tl_if_blank:VT \l__pdfmeta_xmp_metalang_tl
865   { \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

```

866 \__pdfmeta_xmp_build_iptc_data:N \l__pdfmeta_xmp_iptc_data_tl
867 \tl_if_empty:NT \l__pdfmeta_xmp_iptc_data_tl
868 {
869   \seq_remove_all:Nn \l__pdfmeta_xmp_schema_seq { Iptc4xmpCore }
870 }

```

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

```

871 \__pdfmeta_xmp_add_packet_chunk:e
872 {<?xpacket~begin="\__pdfmeta_xmp_generate_bom:"~id="W5MOMpCehiHzreSzNTczkc9d"?>}
873 \__pdfmeta_xmp_add_packet_open:nn{x}{xmpmeta~xmlns:x="adobe:ns:meta/"}
874 \__pdfmeta_xmp_add_packet_open:ne{rdf}
875 {RDF~xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\c_hash_str"}

```

The rdf namespaces

```

876 \__pdfmeta_xmp_add_packet_open_attr:nne
877 {rdf}{Description}{rdf:about="" \g__pdfmeta_xmp_xmlns_tl}

```

The extensions

```

878 \__pdfmeta_xmp_add_packet_open:nn{pdfaExtension}{schemas}
879 \__pdfmeta_xmp_add_packet_open:nn {rdf}{Bag}
880 \seq_map_inline:Nn \l__pdfmeta_xmp_schema_seq
881 {
882   \tl_use:c { g__pdfmeta_xmp_schema_##1_tl }
883 }
884 \__pdfmeta_xmp_add_packet_close:nn {rdf}{Bag}
885 \__pdfmeta_xmp_add_packet_close:nn {pdfaExtension}{schemas}

```

Now starts the part with the data.

```

886 % data
887 \__pdfmeta_xmp_build_pdf:
888 \__pdfmeta_xmp_build_xmpRights:
889 \__pdfmeta_xmp_build_standards: %pdfaid,pdfxid,pdfuaid
890 \__pdfmeta_xmp_build_pdfd:
891 \__pdfmeta_xmp_build_dc:
892 \__pdfmeta_xmp_build_photoshop:
893 \__pdfmeta_xmp_build_xmp:
894 \__pdfmeta_xmp_build_xmpMM:
895 \__pdfmeta_xmp_build_prism:
896 \__pdfmeta_xmp_build_iptc:
897 \__pdfmeta_xmp_build_user: %user additions
898 % end
899 \__pdfmeta_xmp_add_packet_close:nn {rdf}{Description}
900 \__pdfmeta_xmp_add_packet_close:nn {rdf}{RDF}
901 \__pdfmeta_xmp_add_packet_close:nn {x}{xmpmeta}
902 \int_set:Nn \l__pdfmeta_xmp_indent_int{20}
903 \prg_replicate:nn{10}{\__pdfmeta_xmp_add_packet_chunk:n {}}
904 \int_zero:N \l__pdfmeta_xmp_indent_int
905 \__pdfmeta_xmp_add_packet_chunk:n {<?xpacket~end="w"?>}
906 }

```

(End of definition for __pdfmeta_xmp_build_packet:.)

4.6 Building the chunks: rdf namespaces

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

```
\g_pdfmeta_xmp_xmlns_tl  
\g_pdfmeta_xmp_xmlns_prop
```

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

```
907 \str_new:N \g_pdfmeta_xmp_xmlns_tl  
908 \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
```

```
909 \cs_new_protected:Npn \_pdfmeta_xmp_xmlns_new:nn #1 #2  
910 {  
911   \prop_gput:Nnn \g_pdfmeta_xmp_xmlns_prop {#1}{#2}  
912   \tl_gput_right:Ne \g_pdfmeta_xmp_xmlns_tl  
913   {  
914     \_pdfmeta_xmp_indent:n{4} xmlns:\exp_not:n{#1="#2"}  
915   }  
916 }
```

(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

```
917 \_pdfmeta_xmp_xmlns_new:nn {pdf}      {http://ns.adobe.com/pdf/1.3/}  
918 \_pdfmeta_xmp_xmlns_new:nn {xmpRights}{http://ns.adobe.com/xap/1.0/rights/}  
919 \_pdfmeta_xmp_xmlns_new:nn {dc}      {http://purl.org/dc/elements/1.1/}  
920 \_pdfmeta_xmp_xmlns_new:nn {photoshop}{http://ns.adobe.com/photoshop/1.0/}  
921 \_pdfmeta_xmp_xmlns_new:nn {xmp}     {http://ns.adobe.com/xap/1.0/}  
922 \_pdfmeta_xmp_xmlns_new:nn {xmpMM}   {http://ns.adobe.com/xap/1.0/mm/}  
923 \_pdfmeta_xmp_xmlns_new:nn {stEvt}  
924   {http://ns.adobe.com/xap/1.0/sType/ResourceEvent\c_hash_str}  
925 \_pdfmeta_xmp_xmlns_new:nn {pdfaid}   {http://www.aiim.org/pdfa/ns/id/}  
926 \_pdfmeta_xmp_xmlns_new:nn {pdfuaid}  {http://www.aiim.org/pdfua/ns/id/}  
927 \_pdfmeta_xmp_xmlns_new:nn {pdfx}    {http://ns.adobe.com/pdfx/1.3/}  
928 \_pdfmeta_xmp_xmlns_new:nn {pdfxid}  {http://www.npes.org/pdfx/ns/id/}  
929 \_pdfmeta_xmp_xmlns_new:nn {prism}   {http://prismstandard.org/namespaces/basic/3.0/}  
930 %\_pdfmeta_xmp_xmlns_new:nn {jav}     {http://www.niso.org/schemas/jav/1.0/}  
931 %\_pdfmeta_xmp_xmlns_new:nn {xmpTPg}  {http://ns.adobe.com/xap/1.0/t/pg/}  
932 \_pdfmeta_xmp_xmlns_new:nn {stFnt}   {http://ns.adobe.com/xap/1.0/sType/Font\c_hash_str}  
933 \_pdfmeta_xmp_xmlns_new:nn {Iptc4xmpCore}{http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}  
934 \_pdfmeta_xmp_xmlns_new:nn {pdfaExtension}{http://www.aiim.org/pdfa/ns/extension/}  
935 \_pdfmeta_xmp_xmlns_new:nn {pdfaSchema}{http://www.aiim.org/pdfa/ns/schema\c_hash_str}  
936 \_pdfmeta_xmp_xmlns_new:nn {pdfaProperty}{http://www.aiim.org/pdfa/ns/property\c_hash_str}  
937 \_pdfmeta_xmp_xmlns_new:nn {pdfaType} {http://www.aiim.org/pdfa/ns/type\c_hash_str}  
938 \_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

```
939 \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.

```
940 \cs_new_protected:Npn \__pdfmeta_xmp_schema_new:nnn #1 #2 #3
```

```
941   % #1 name #2 prefix, #3 text
```

```
942   {
```

```
943     \tl_if_exist:cTF { g__pdfmeta_xmp_schema_#2_tl }  
944     {
```

```
945       \msg_warning:nnnn{pdfmeta}{xmp-defined}{schema}{#2}
```

```
946     }  
947     {
```

```
948       \seq_put_right:Nn \l__pdfmeta_xmp_schema_seq { #2 }
```

```
949       \tl_new:c { g__pdfmeta_xmp_schema_#2_tl }
```

```
950       \tl_new:c { g__pdfmeta_xmp_schema_#2_properties_tl }
```

```
951       \tl_gput_right:cn { g__pdfmeta_xmp_schema_#2_tl }
```

```
952       {
```

```
953         \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
```

```
954         \__pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{schema}{#1}
```

```
955         \__pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{prefix}{#2}
```

```
956         \__pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{namespaceURI}{#3}
```

```
957         \__pdfmeta_xmp_add_packet_open:nn {pdfaSchema}{property}
```

```
958         \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
```

```
959         \tl_use:c { g__pdfmeta_xmp_schema_#2_properties_tl }
```

```
960         \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
```

```
961         \__pdfmeta_xmp_add_packet_close:nn {pdfaSchema}{property}
```

```
962         \cs_if_exist_use:c {__pdfmeta_xmp_schema_#2_additions:}
```

```
963         \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
```

```
964       }
```

```
965     }
```

```
966   }
```

(End of definition for __pdfmeta_xmp_schema_new:nnn.)

`__pdfmeta_xmp_property_new:nnnnn` This adds a property to a schema.

```
967 \prop_new:N\g__pdfmeta_xmp_schema_property_prop
```

```
968 \cs_new_protected:Npn \__pdfmeta_xmp_property_new:nnnnn #1 #2 #3 #4 #5 %
```

```
969   % #1 schema #2 name, #3 type, #4 category #5 description
```

```
970   {
```

```
971     \tl_if_exist:cTF { g__pdfmeta_xmp_schema_#1_properties_tl }  
972     {
```

```
973       \prop_get:NeNF \g__pdfmeta_xmp_schema_property_prop {#1:#2}\l__pdfmeta_tmpa_tl
```

```
974       {
```

```
975         \prop_gput:Nee \g__pdfmeta_xmp_schema_property_prop {#1:#2}{#3}
```

```
976         \tl_gput_right:cn { g__pdfmeta_xmp_schema_#1_properties_tl }
```

```
977         {
```

```
978           \__pdfmeta_xmp_add_packet_open:nn {rdf}{li~rdf:parseType="Resource"}
```

```

979         \_pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{name}{#2}
980         \_pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{valueType}{#3}
981         \_pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{category}{#4}
982         \_pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{description}{#5}
983     \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
984 }
985 }
986 }
987 {
988     \msg_warning:nnnn{pdfmeta}{xmp-undefined}{schema}{#1}
989 }
990 }

```

(End of definition for _pdfmeta_xmp_property_new:nnnn.)

_pdfmeta_xmp_add_packet_field:nnn This adds a field to a schema.

```

991 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_field:nnn #1 #2 #3 %
992   %#1 name #2 valuetype #3 description
993   {
994     \_pdfmeta_xmp_add_packet_open_attr:nnn {rdf}{li}{rdf:parseType="Resource"}
995     \_pdfmeta_xmp_add_packet_line:nnn {pdfaField}{name}{#1}
996     \_pdfmeta_xmp_add_packet_line:nnn {pdfaField}{valueType}{#2}
997     \_pdfmeta_xmp_add_packet_line:nnn {pdfaField}{description}{#3}
998     \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
999   }

```

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

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

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

```

1000     \_pdfmeta_xmp_schema_new:nnn
1001     {XMP~Media~Management~Schema}
1002     {xmpMM}
1003     {http://ns.adobe.com/xap/1.0/mm/}
1004     \_pdfmeta_xmp_property_new:nnnnn
1005     {xmpMM}
1006     {OriginalDocumentID}
1007     {URI}
1008     {internal}
1009     {The~common~identifier~for~all~versions~and~renditions~of~a~document.}

```

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.

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)

```

1010     \_pdfmeta_xmp_schema_new:nnn
1011         {PDF/A-Identification-Schema}
1012         {pdfaid}
1013         {http://www.aiim.org/pdfa/ns/id/}
1014     \_pdfmeta_xmp_property_new:nnnnn
1015         {pdfaid}
1016         {year}
1017         {Integer}
1018         {internal}
1019         {Year-of~standard}
1020     \_pdfmeta_xmp_property_new:nnnnn
1021         {pdfaid}
1022         {rev}
1023         {Integer}
1024         {internal}
1025         {Revision~year~of~standard}

```

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

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

pdfuaid-(schema)

```

1026     \_pdfmeta_xmp_schema_new:nnn
1027         {PDF/UA-Universal-Accessibility-Schema}
1028         {pdfuaid}
1029         {http://www.aiim.org/pdfua/ns/id/}
1030     \_pdfmeta_xmp_property_new:nnnnn
1031         {pdfuaid}
1032         {part}
1033         {Integer}
1034         {internal}
1035         {Part-of~ISO~14289~standard}
1036     \_pdfmeta_xmp_property_new:nnnnn
1037         {pdfuaid}
1038         {rev}
1039         {Integer}
1040         {internal}
1041         {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

pdfxid-(schema)

```
1042 \_pdfmeta_xmp_schema_new:nnn
1043 {PDF/X-ID~Schema}
1044 {pdfxid}
1045 {http://www.npes.org/pdfx/ns/id/}
1046 \_pdfmeta_xmp_property_new:nnnnn
1047 {pdfxid}
1048 {GTS_PDFXVersion}
1049 {Text}
1050 {internal}
1051 {ID~of~PDF/X~standard}
```

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

prism-(schema)

```
1052 \_pdfmeta_xmp_schema_new:nnn
1053 {PRISM-Basic-Metadata}
1054 {prism}
1055 {http://prismstandard.org/namespaces/basic/3.0/}
1056 \_pdfmeta_xmp_property_new:nnnnn
1057 {prism}
1058 {complianceProfile}
1059 {Text}
1060 {internal}
1061 {PRISM~specification~compliance~profile~to~which~this~document~adheres}
1062 \_pdfmeta_xmp_property_new:nnnnn
1063 {prism}
1064 {publicationName}
1065 {Text}
1066 {external}
1067 {Publication-name}
1068 \_pdfmeta_xmp_property_new:nnnnn
1069 {prism}
1070 {aggregationType}
1071 {Text}
1072 {external}
1073 {Publication-type}
1074 \_pdfmeta_xmp_property_new:nnnnn
1075 {prism}
1076 {bookEdition}
1077 {Text}
1078 {external}
1079 {Edition-of-the-book-in-which-the-document-was-published}
1080 \_pdfmeta_xmp_property_new:nnnnn
1081 {prism}
1082 {volume}
1083 {Text}
1084 {external}
1085 {Publication-volume-number}
1086 \_pdfmeta_xmp_property_new:nnnnn
1087 {prism}
1088 {number}
1089 {Text}
```

```

1090     {external}
1091     {Publication~issue~number~within~a~volume}
1092 \_pdfmeta_xmp_property_new:nnnnn
1093     {prism}
1094     {pageRange}
1095     {Text}
1096     {external}
1097     {Page~range~for~the~document~within~the~print~version~of~its~publication}
1098 \_pdfmeta_xmp_property_new:nnnnn
1099     {prism}
1100     {issn}
1101     {Text}
1102     {external}
1103     {ISSN~for~the~printed~publication~in~which~the~document~was~published}
1104 \_pdfmeta_xmp_property_new:nnnnn
1105     {prism}
1106     {eIssn}
1107     {Text}
1108     {external}
1109     {ISSN~for~the~electronic~publication~in~which~the~document~was~published}
1110 \_pdfmeta_xmp_property_new:nnnnn
1111     {prism}
1112     {isbn}
1113     {Text}
1114     {external}
1115     {ISBN~for~the~publication~in~which~the~document~was~published}
1116 \_pdfmeta_xmp_property_new:nnnnn
1117     {prism}
1118     {doi}
1119     {Text}
1120     {external}
1121     {Digital~Object~Identifier~for~the~document}
1122 \_pdfmeta_xmp_property_new:nnnnn
1123     {prism}
1124     {url}
1125     {URL}
1126     {external}
1127     {URL~at~which~the~document~can~be~found}
1128 \_pdfmeta_xmp_property_new:nnnnn
1129     {prism}
1130     {byteCount}
1131     {Integer}
1132     {internal}
1133     {Approximate~file~size~in~octets}
1134 \_pdfmeta_xmp_property_new:nnnnn
1135     {prism}
1136     {pageCount}
1137     {Integer}
1138     {internal}
1139     {Number~of~pages~in~the~print~version~of~the~document}
1140 \_pdfmeta_xmp_property_new:nnnnn
1141     {prism}
1142     {subtitle}
1143     {Text}

```

```

1144     {external}
1145     {Document's~subtitle}

```

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

iptc_□(schema)iptc

```

1146     \_pdfmeta_xmp_schema_new:nnn
1147     {IPTC~Core~Schema}
1148     {Iptc4xmpCore}
1149     {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
1150     \_pdfmeta_xmp_property_new:nnnnn
1151     {Iptc4xmpCore}
1152     {CreatorContactInfo}
1153     {ContactInfo}
1154     {external}
1155     {Document~creator's~contact~information}
1156     \cs_new_protected:cpn { __pdfmeta_xmp_schema_Iptc4xmpCore_additions: }
1157     {
1158         \_pdfmeta_xmp_add_packet_open:nn{pdfaSchema}{valueType}
1159         \_pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1160         \_pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1161         \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{ContactInfo}
1162         \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
1163             {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
1164         \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{Iptc4xmpCore}
1165         \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
1166             {Basic~set~of~information~to~get~in~contact~with~a~person}
1167         \_pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
1168         \_pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1169         \_pdfmeta_xmp_add_packet_field:nnn{CiAdrCity}{Text}
1170             {Contact~information~city}
1171         \_pdfmeta_xmp_add_packet_field:nnn{CiAdrCtry}{Text}
1172             {Contact~information~country}
1173         \_pdfmeta_xmp_add_packet_field:nnn{CiAdrExtadr}{Text}
1174             {Contact~information~address}
1175         \_pdfmeta_xmp_add_packet_field:nnn{CiAdrPcode}{Text}
1176             {Contact~information~local~postal~code}
1177         \_pdfmeta_xmp_add_packet_field:nnn{CiAdrRegion}{Text}
1178             {Contact~information~regional~information~such~as~state~or~province}
1179         \_pdfmeta_xmp_add_packet_field:nnn{CiEmailWork}{Text}
1180             {Contact~information~email~address(es)}
1181         \_pdfmeta_xmp_add_packet_field:nnn{CiTelWork}{Text}
1182             {Contact~information~telephone~number(s)}
1183         \_pdfmeta_xmp_add_packet_field:nnn{CiUrlWork}{Text}
1184             {Contact~information~Web~URL(s)}
1185         \_pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1186         \_pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
1187         \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1188         \_pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1189         \_pdfmeta_xmp_add_packet_close:nn{pdfaSchema}{valueType}
1190     }

```

(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.)

```
1191 \cs_new_protected:Npn \__pdfmeta_xmp_schema_enable_pdfd:
1192 {
1193   \__pdfmeta_xmp_xmlns_new:nn {pdfd}{http://pdfa.org/declarations/}
1194   \__pdfmeta_xmp_schema_new:nnn
1195     {PDF-Declarations~Schema}
1196     {pdfd}
1197     {http://pdfa.org/declarations/}
1198   \__pdfmeta_xmp_property_new:nnnnn
1199     {pdfd}
1200     {declarations}
1201     {Bag~declaration}
1202     {external}
1203     {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.

```
1204 \cs_new_protected:cpn { __pdfmeta_xmp_schema_pdfd_additions: }
1205 {
1206   \__pdfmeta_xmp_add_packet_open:nn{pdfaSchema}{valueType}
1207   \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1208   \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1209   \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{claim}
1210   \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
1211     {http://pdfa.org/declarations/}
1212   \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{pdfd}
1213   \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
1214     {A~structure~describing~properties~of~an~individual~claim.}
1215   \__pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
1216   \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1217   \__pdfmeta_xmp_add_packet_field:nnn{claimReport}{Text}
1218     {A~URL~to~a~report~containing~details~of~the~specific~conformance~claim}
1219   \__pdfmeta_xmp_add_packet_field:nnn{claimCredentials}{Text}
1220     {The~claimant's~credentials.}
1221   \__pdfmeta_xmp_add_packet_field:nnn{claimDate}{Text}
1222     {A~date~identifying~when~the~claim~was~made.}
1223   \__pdfmeta_xmp_add_packet_field:nnn{claimBy}{Text}
1224     {The~name~of~the~organization~and/or~individual~and/or~software~making}
1225   \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1226   \__pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
1227   \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1228   \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
```



```

1229         \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{declaration}
1230         \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
1231             {http://pdfa.org/declarations/}
1232         \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{pdfd}
1233         \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
1234             {A~structure~describing~a~single~PDF~ Declaration~asserting~conformance~v
1235         \__pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
1236         \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1237         \__pdfmeta_xmp_add_packet_field:nnn{conformsTo}{Text}
1238             {A~property~containing~a~URI~specifying~the~standard~or~profile~by~the
1239         \__pdfmeta_xmp_add_packet_field:nnn{claimData}{Bag~claim}
1240             {An~unordered~array~of~claim~data,~where~each~claim~identifies~the~natu
1241         \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1242         \__pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
1243         \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1244         \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1245         \__pdfmeta_xmp_add_packet_close:nn{pdfaSchema}{valueType}
1246     }

```

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

```

1247         \cs_gset_eq:NN \__pdfmeta_xmp_schema_enable_pdfd: \prg_do_nothing:
1248     }

```

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 1249 \cs_new_protected:Npn \__pdfmeta_xmp_build_pdf:
  PDFversion           1250 {

```

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

```

1251     \__pdfmeta_xmp_add_packet_line_default:nnee
1252         {pdf}{Producer}
1253         {\c_sys_engine_exec_str-\c_sys_engine_version_str}
1254         {\GetDocumentProperties{hyperref/pdfproducer}}

```

Now the PDF version

```

1255     \__pdfmeta_xmp_add_packet_line:nne{pdf}{PDFVersion}{\pdf_version:}
1256 }

```

(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 1257 \cs_new_protected:Npn \__pdfmeta_xmp_build_xmp:
  BaseUrl/baseurl       1258 {

```

The creator

```
1259 \__pdfmeta_xmp_add_packet_line_default:nnee
1260   {xmp}{CreatorTool}
1261   {LaTeX}
1262   { \GetDocumentProperties{hyperref/pdfcreator} }
```

The baseurl

```
1263 \__pdfmeta_xmp_add_packet_line_default:nnee
1264   {xmp}{BaseURL}{}
1265   { \GetDocumentProperties{hyperref/baseurl} }
```

CreationDate

```
1266 \__pdfmeta_xmp_date_get:nNN
1267   {document/creationdate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1268 \__pdfmeta_xmp_add_packet_line:nne{xmp}{CreateDate}{\__pdfmeta_xmp_print_date:N\l__pdfme
1269 \pdfmanagement_add:nne{Info}{CreateDate}{(\l__pdfmeta_tmpa_tl)}
```

ModifyDate

```
1270 \__pdfmeta_xmp_date_get:nNN
1271   {document/moddate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1272 \__pdfmeta_xmp_add_packet_line:nne{xmp}{ModifyDate}{\__pdfmeta_xmp_print_date:N\l__pdfme
1273 \pdfmanagement_add:nne{Info}{ModDate}{(\l__pdfmeta_tmpa_tl)}
```

MetadataDate

```
1274 \__pdfmeta_xmp_date_get:nNN
1275   {hyperref/pdfmetadate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1276 \__pdfmeta_xmp_add_packet_line:nne{xmp}{MetadataDate}{\__pdfmeta_xmp_print_date:N\l__pdf
1277 }
```

(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:`

```
1278 \cs_new_protected:Npn \__pdfmeta_xmp_build_standards:
1279   {
1280   \__pdfmeta_xmp_add_packet_line:nne {pdfaid}{part}{\pdfmeta_standard_item:n{n{level}}}
1281   \__pdfmeta_xmp_add_packet_line:nne
1282     {pdfaid}{conformance}{\pdfmeta_standard_item:n{n{conformance}}}
1283   \int_compare:nNnTF {0\pdfmeta_standard_item:n{n{level}}}<{4}
1284     {\__pdfmeta_xmp_add_packet_line:nne {pdfaid}{year} {\pdfmeta_standard_item:n{n{year}}}}
1285     {\__pdfmeta_xmp_add_packet_line:nne {pdfaid}{rev} {\pdfmeta_standard_item:n{n{year}}}}
1286   \__pdfmeta_xmp_add_packet_line:nne
1287     {pdfxid}{GTS_PDFXVersion}{\GetDocumentProperties{document/pdfstandard-X}}
1288   \pdfmanagement_get_documentproperties:nNT {document/pdfstandard-UA}\l__pdfmeta_tmpa_tl
1289     {
1290     \__pdfmeta_xmp_add_packet_line:nne
1291       {pdfuaid}{part}{\exp_last_unbraced:No\use_i:nn \l__pdfmeta_tmpa_tl}
1292     \__pdfmeta_xmp_add_packet_line:nne
1293       {pdfuaid}{rev}{\exp_last_unbraced:No\use_ii:nn \l__pdfmeta_tmpa_tl}
```

```

1294     }
1295 }

```

(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.

```

1296 \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:

```

1297 \cs_new_protected:Npn \_pdfmeta_xmp_build_pdfd:
1298 {
1299   \prop_if_empty:NF\g_pdfmeta_xmp_pdfd_data_prop
1300   {
1301     \_pdfmeta_xmp_add_packet_open:nn{pdfd}{declarations}
1302     \_pdfmeta_xmp_add_packet_open:nn{rdf}{Bag}
1303     \prop_map_inline:Nn \g_pdfmeta_xmp_pdfd_data_prop
1304     {
1305       \_pdfmeta_xmp_build_pdfd_claim:nn{##1}{##2}
1306     }
1307     \_pdfmeta_xmp_add_packet_close:nn{rdf}{Bag}
1308     \_pdfmeta_xmp_add_packet_close:nn{pdfd}{declarations}
1309   }
1310 }

```

(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.

```

1311 \cs_new_protected:Npn \_pdfmeta_xmp_build_pdfd_claim:nn #1#2
1312 {
1313   \_pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1314   \_pdfmeta_xmp_add_packet_line:nnn{pdfd}{conformsTo}{#1}
1315   \tl_if_empty:NF {#2}
1316   {
1317     \_pdfmeta_xmp_add_packet_open:nn{pdfd}{claimData}
1318     \_pdfmeta_xmp_add_packet_open:nn{rdf}{Bag}
1319     #2
1320     \_pdfmeta_xmp_add_packet_close:nn{rdf}{Bag}
1321     \_pdfmeta_xmp_add_packet_close:nn{pdfd}{claimData}
1322   }
1323   \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1324 }

```

(End of definition for _pdfmeta_xmp_build_pdfd_claim:nn.)

4.10 Photoshop

_pdfmeta_xmp_build_photoshop:

```
1325 \cs_new_protected:Npn \_pdfmeta_xmp_build_photoshop:
1326 {
pdfauthor/photshop:AuthorsPosition
1327   \_pdfmeta_xmp_add_packet_line:nne{photoshop}{AuthorsPosition}
1328   { \GetDocumentProperties{hyperref/pdfauthor} }
pdfcaptionwriter/photshop:CaptionWriter
1329   \_pdfmeta_xmp_add_packet_line:nne{photoshop}{CaptionWriter}
1330   { \GetDocumentProperties{hyperref/pdfcaptionwriter} }
1331 }
```

(End of definition for _pdfmeta_xmp_build_photoshop:.)

4.11 XMP Media Management

_pdfmeta_xmp_build_xmpMM:

```
1332 \cs_new_protected:Npn \_pdfmeta_xmp_build_xmpMM:
1333 {
pdfdocumentid / xmpMM:DocumentID
1334   \str_set:Ne\l__pdfmeta_tmpa_str {\GetDocumentProperties{hyperref/pdfdocumentid}}
1335   \str_if_empty:NT \l__pdfmeta_tmpa_str
1336   {
1337     \_pdfmeta_xmp_create_uuid:nN
1338     {\jobname\GetDocumentProperties{hyperref/pdftitle}}
1339     \l__pdfmeta_tmpa_str
1340   }
1341   \_pdfmeta_xmp_add_packet_line:nnV{xmpMM}{DocumentID}
1342   \l__pdfmeta_tmpa_str
pdfinstanceid / xmpMM:InstanceID
1343   \str_set:Ne\l__pdfmeta_tmpa_str {\GetDocumentProperties{hyperref/pdfinstanceid}}
1344   \str_if_empty:NT \l__pdfmeta_tmpa_str
1345   {
1346     \_pdfmeta_xmp_create_uuid:nN
1347     {\jobname\l__pdfmeta_xmp_currentdate_tl}
1348     \l__pdfmeta_tmpa_str
1349   }
1350   \_pdfmeta_xmp_add_packet_line:nnV{xmpMM}{InstanceID}
1351   \l__pdfmeta_tmpa_str
pdfversionid/xmpMM:VersionID
1352   \_pdfmeta_xmp_add_packet_line:nne{xmpMM}{VersionID}
1353   { \GetDocumentProperties{hyperref/pdfversionid} }
pdfrendition/xmpMM:RenditionClass
1354   \_pdfmeta_xmp_add_packet_line:nne{xmpMM}{RenditionClass}
1355   { \GetDocumentProperties{hyperref/pdfrendition} }
1356 }
```

(End of definition for _pdfmeta_xmp_build_xmpMM:.)

4.12 Rest of dublin Core data

```
\_pdfmeta_xmp_build_dc:
  dc:creator/pdfauthor 1357 \cs_new_protected:Npn \_pdfmeta_xmp_build_dc:
  dc:subject/pdfkeywords 1358 {
    dc:type/pdfdtype pdfauthor/dc:creator
  dc:publisher/pdfpublisher 1359 \_pdfmeta_xmp_add_packet_list:nnne {dc}{creator}{Seq}
  dc:description/pdfsubject 1360 { \GetDocumentProperties{hyperref/pdfauthor} }
  dc:language/lang/pdflang 1361 \int_compare:nNnT {0\pdfmeta_standard_item:n{level}}={1}
dc:identifier/pdfidentifier 1362 { \pdfmanagement_remove:nn{Info}{Author} }
photoshop:AuthorsPosition/pdfauthortitle pdftitle/dc:title. This is rather complex as we want to support a list with different
photoshop:CaptionWriter/pdfcaptionwriter languages.
1363 \_pdfmeta_xmp_add_packet_list:nnne {dc}{title}{Alt}
1364 { \GetDocumentProperties{hyperref/pdftitle} }
pdfkeywords/dc:subject
1365 \_pdfmeta_xmp_add_packet_list:nnne {dc}{subject}{Bag}
1366 { \GetDocumentProperties{hyperref/pdfkeywords} }
1367 \int_compare:nNnT {0\pdfmeta_standard_item:n{level}}={1}
1368 { \pdfmanagement_remove:nn{Info}{Keywords} }
pdfdtype/dc:type
1369 \pdfmanagement_get_documentproperties:nNTF { hyperref/pdfdtype } \l__pdfmeta_tmpa_tl
1370 {
1371 \_pdfmeta_xmp_add_packet_list_simple:nnnV {dc}{type}{Bag}\l__pdfmeta_tmpa_tl
1372 }
1373 {
1374 \_pdfmeta_xmp_add_packet_list_simple:nnnn {dc}{type}{Bag}{Text}
1375 }
pdfpublisher/dc:publisher
1376 \_pdfmeta_xmp_add_packet_list:nnne {dc}{publisher}{Bag}
1377 { \GetDocumentProperties{hyperref/pdfpublisher} }
pdfsubject/dc:description
1378 \_pdfmeta_xmp_add_packet_list:nnne
1379 {dc}{description}{Alt}
1380 { \GetDocumentProperties{hyperref/pdfsubject} }
lang/pdflang/dc:language
1381 \_pdfmeta_xmp_add_packet_list_simple:nnnV
1382 {dc}{language}{Bag}\l__pdfmeta_xmp_doclang_tl
pdfidentifier/dc:identifier
1383 \_pdfmeta_xmp_add_packet_line:nne{dc}{identifier}
1384 { \GetDocumentProperties{hyperref/pdfidentifier} }
pdfdate/dc:date
1385 \_pdfmeta_xmp_date_get:nNN {hyperref/pdfdate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1386 \_pdfmeta_xmp_add_packet_list_simple:nnne
1387 {dc}{date}{Seq}{\_pdfmeta_xmp_print_date:N\l__pdfmeta_tmpa_seq}
The file format
1388 \_pdfmeta_xmp_add_packet_line:nnn{dc}{format}{application/pdf}
```

The source

```
1389 \_pdfmeta_xmp_add_packet_line_default:nnee
1390 {dc}{source}
1391 { \c_sys_jobname_str.tex }
1392 { \GetDocumentProperties{hyperref/pdfsourc} }
1393 \_pdfmeta_xmp_add_packet_list:nne{dc}{rights}{Alt}
1394 {\GetDocumentProperties{hyperref/pdfcopyright}}
1395 }
```

(End of definition for `_pdfmeta_xmp_build_dc`: and others.)

4.13 xmpRights

`_pdfmeta_xmp_build_xmpRights`:

```
1396 \cs_new_protected:Npn \_pdfmeta_xmp_build_xmpRights:
1397 {
1398   \_pdfmeta_xmp_add_packet_line:nne
1399     {xmpRights}
1400     {WebStatement}
1401     {\GetDocumentProperties{hyperref/pdflicenseurl}}
1402   \_pdfmeta_xmp_add_packet_line:nne
1403     {xmpRights}
1404     {Marked}
1405   {
1406     \str_case:en {\GetDocumentProperties{document/copyright}}
1407     {
1408       {true}{True}
1409       {false}{False}
1410     }
1411   }
1412 }
```

(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`

```
1413 \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`

```
1414 \cs_new_protected:Npn \_pdfmeta_xmp_build_iptc_data:N #1
1415 {
1416   \tl_clear:N #1
1417   \_pdfmeta_xmp_incr_indent:\_pdfmeta_xmp_incr_indent:\_pdfmeta_xmp_incr_indent:\_pdf
1418   \_pdfmeta_xmp_add_packet_line:nneN
1419     {Iptc4xmpCore}{CiAdrExtadr}
1420     {\GetDocumentProperties{hyperref/pdfcontactaddress}}
1421     #1
```

```

1422 \_pdfmeta_xmp_add_packet_line:nneN
1423 {Iptc4xmpCore}{CiAdrCity}
1424 {\GetDocumentProperties{hyperref/pdfcontactcity}}
1425 #1
1426 \_pdfmeta_xmp_add_packet_line:nneN
1427 {Iptc4xmpCore}{CiAdrPcode}
1428 {\GetDocumentProperties{hyperref/pdfcontactpostcode}}
1429 #1
1430 \_pdfmeta_xmp_add_packet_line:nneN
1431 {Iptc4xmpCore}{CiAdrCtry}
1432 {\GetDocumentProperties{hyperref/pdfcontactcountry}}
1433 #1
1434 \_pdfmeta_xmp_add_packet_line:nneN
1435 {Iptc4xmpCore}{CiTelWork}
1436 {\GetDocumentProperties{hyperref/pdfcontactphone}}
1437 #1
1438 \_pdfmeta_xmp_add_packet_line:nneN
1439 {Iptc4xmpCore}{CiEmailWork}
1440 {\GetDocumentProperties{hyperref/pdfcontactemail}}
1441 #1
1442 \_pdfmeta_xmp_add_packet_line:nneN
1443 {Iptc4xmpCore}{CiUrlWork}
1444 {\GetDocumentProperties{hyperref/pdfcontacturl}}
1445 #1
1446 \_pdfmeta_xmp_decr_indent:\_pdfmeta_xmp_decr_indent:\_pdfmeta_xmp_decr_indent:\_pdf
1447 }

```

(End of definition for _pdfmeta_xmp_build_iptc:N.)

_pdfmeta_xmp_build_iptc:

```

1448 \cs_new_protected:Npn \_pdfmeta_xmp_build_iptc:
1449 {
1450   \tl_if_empty:NF\l__pdfmeta_xmp_iptc_data_tl
1451   {
1452     \_pdfmeta_xmp_add_packet_open_attr:nnn
1453     {Iptc4xmpCore}{CreatorContactInfo}{rdf:parseType="Resource"}
1454     \tl_gput_right:Ne\g__pdfmeta_xmp_packet_tl { \l__pdfmeta_xmp_iptc_data_tl }
1455     \_pdfmeta_xmp_add_packet_close:nn
1456     {Iptc4xmpCore}{CreatorContactInfo}
1457   }
1458 }

```

(End of definition for _pdfmeta_xmp_build_iptc:.)

4.15 Prism

_pdfmeta_xmp_build_prism:
 complianceProfile
 prism:subtitle/pdfsubtitle

```

1459 \cs_new_protected:Npn \_pdfmeta_xmp_build_prism:
1460 {

```

The compliance profile is a fix value taken from hyperxmp

```

1461 \_pdfmeta_xmp_add_packet_line:nnn
1462 {prism}{complianceProfile}
1463 {three}

```

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

```
1464 \__pdfmeta_xmp_lang_get:eNN
1465 {\GetDocumentProperties{hyperref/pdfsubtitle}}
1466 \l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl
1467 \__pdfmeta_xmp_add_packet_line_attr:nneV
1468 {prism}{subtitle}
1469 {xml:lang="\l__pdfmeta_tmpa_tl"}
1470 \l__pdfmeta_tmpb_tl
```

Then publicationName

```
1471 \__pdfmeta_xmp_lang_get:eNN
1472 {\GetDocumentProperties{hyperref/pdfpublication}}
1473 \l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl
1474 \__pdfmeta_xmp_add_packet_line_attr:nneV
1475 {prism}{publicationName}
1476 {xml:lang="\l__pdfmeta_tmpa_tl"}
1477 \l__pdfmeta_tmpb_tl
```

Now the rest

```
1478 \__pdfmeta_xmp_add_packet_line:nne
1479 {prism}{bookEdition}
1480 {\GetDocumentProperties{hyperref/pdfbookedition}}
1481 \__pdfmeta_xmp_add_packet_line:nne
1482 {prism}{aggregationType}
1483 {\GetDocumentProperties{hyperref/pdfpubtype}}
1484 \__pdfmeta_xmp_add_packet_line:nne
1485 {prism}{volume}
1486 {\GetDocumentProperties{hyperref/pdfvolumenum}}
1487 \__pdfmeta_xmp_add_packet_line:nne
1488 {prism}{number}
1489 {\GetDocumentProperties{hyperref/pdfissuenum}}
1490 \__pdfmeta_xmp_add_packet_line:nne
1491 {prism}{pageRange}
1492 {\GetDocumentProperties{hyperref/pdfpagerange}}
1493 \__pdfmeta_xmp_add_packet_line:nne
1494 {prism}{issn}
1495 {\GetDocumentProperties{hyperref/pdfissn}}
1496 \__pdfmeta_xmp_add_packet_line:nne
1497 {prism}{eIssn}
1498 {\GetDocumentProperties{hyperref/pdfeissn}}
1499 \__pdfmeta_xmp_add_packet_line:nne
1500 {prism}{doi}
1501 {\GetDocumentProperties{hyperref/pdfdoi}}
1502 \__pdfmeta_xmp_add_packet_line:nne
1503 {prism}{url}
1504 {\GetDocumentProperties{hyperref/pdfurl}}
```

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

```
1505 \tl_set:Nc \l__pdfmeta_tmpa_tl { \GetDocumentProperties{hyperref/pdfnumpages} }
1506 \__pdfmeta_xmp_add_packet_line:nne
1507 {prism}{pageCount}
1508 {\tl_if_blank:VTF \l__pdfmeta_tmpa_tl {\PreviousTotalPages}{\l__pdfmeta_tmpa_tl}}
1509 }
```

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

4.15.1 User additions

`\g_pdfmeta_xmp_user_packet_str`

```
1510 \tl_new:N \g__pdfmeta_xmp_user_packet_tl
```

(End of definition for \g__pdfmeta_xmp_user_packet_str.)

`__pdfmeta_xmp_build_user:`

```
1511 \cs_new_protected:Npn \__pdfmeta_xmp_build_user:
1512 {
1513   \int_zero:N \l__pdfmeta_xmp_indent_int
1514   \g__pdfmeta_xmp_user_packet_tl
1515   \int_set:Nn \l__pdfmeta_xmp_indent_int {3}
1516 }
```

(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

```
1517 \AddToHook{shipout/lastpage}
1518 {
1519   \bool_if:NT\g__pdfmeta_xmp_bool
1520   {
1521     \str_if_exist:NTF\c_sys_timestamp_str
1522     {
1523       \tl_set_eq:NN \l__pdfmeta_xmp_currentdate_tl \c_sys_timestamp_str
1524     }
1525     {
1526       \file_get_timestamp:nN{\jobname.log}\l__pdfmeta_xmp_currentdate_tl
1527     }
1528     \__pdfmeta_xmp_date_split:VN\l__pdfmeta_xmp_currentdate_tl\l__pdfmeta_xmp_currentdate_tl
1529     \__pdfmeta_xmp_build_packet:
1530     \exp_args:No
1531     \__pdf_backend_metadata_stream:n {\g__pdfmeta_xmp_packet_tl}
1532     \pdfmanagement_add:nne {Catalog} {Metadata}{\pdf_object_ref_last:}
1533     \bool_if:NT \g__pdfmeta_xmp_export_bool
1534     {
1535       \iow_open:Nn\g_tmpa_iow{\g__pdfmeta_xmp_export_str.xmpi}
1536       \exp_args:NNo\iow_now:Nn\g_tmpa_iow{\g__pdfmeta_xmp_packet_tl}
1537       \iow_close:N\g_tmpa_iow
1538     }
1539   }
1540 }
```

4.17 User commands

`\pdfmeta_xmp_add:n`

```
1541 \cs_new_protected:Npn \pdfmeta_xmp_add:n #1
1542 {
1543   \tl_gput_right:Nn \g__pdfmeta_xmp_user_packet_tl
1544   {
1545     \__pdfmeta_xmp_add_packet_chunk:n { #1 }
1546   }
1547 }
```

```

1546     }
1547 }

```

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

`\pdfmeta_xmp_xmlns_new:nn`

```

1548 \cs_new_protected:Npn \pdfmeta_xmp_xmlns_new:nn #1 #2
1549 {
1550   \prop_if_in:NnTF \g__pdfmeta_xmp_xmlns_prop {#1}
1551     {\msg_warning:nnnn{pdfmeta}{xmp-defined}{xmlns-namespace}{#1}}
1552     {\__pdfmeta_xmp_xmlns_new:nn {#1}{#2}}
1553 }

```

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

`\pdfmeta_xmp_schema_new:nnn`

```

1554 \cs_set_eq:NN \pdfmeta_xmp_schema_new:nnn \__pdfmeta_xmp_schema_new:nnn

```

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

`\pdfmeta_xmp_property_new:nnnn`

```

1555 \cs_set_eq:NN \pdfmeta_xmp_property_new:nnnn \__pdfmeta_xmp_property_new:nnnn

```

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

`\pdfmeta_xmp_add_declaration:n`

`\pdfmeta_xmp_add_declaration:e`

```

1556 \cs_new_protected:Npn \pdfmeta_xmp_add_declaration:n #1 %conformsTo uri
1557 {
1558   \__pdfmeta_xmp_schema_enable_pdfd:
1559   \prop_gput:Nnn\g__pdfmeta_xmp_pdfd_data_prop{#1}{}
1560 }
1561 \cs_generate_variant:Nn \pdfmeta_xmp_add_declaration:n {e}

```

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

`\pdfmeta_xmp_add_declaration:nnnn`

`\pdfmeta_xmp_add_declaration:ennnn`

```

1562 \cs_new_protected:Npn \pdfmeta_xmp_add_declaration:nnnn #1#2#3#4#5
1563   %#1=conformsTo uri, #2 claimBy, #3 claimDate #4 claimCredentials #4 claimReport
1564   {
1565     \__pdfmeta_xmp_schema_enable_pdfd:
1566     \tl_set:Nn \l__pdfmeta_tmpa_tl
1567       {
1568         \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1569         \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimBy}{#2}
1570         \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimDate}{#3}
1571         \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimCredentials}{#4}
1572         \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimReport}{#5}
1573         \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1574       }
1575     \prop_get:NnNT \g__pdfmeta_xmp_pdfd_data_prop {#1}\l__pdfmeta_tmpb_tl
1576     {
1577       \tl_concat:NNN \l__pdfmeta_tmpa_tl \l__pdfmeta_tmpa_tl \l__pdfmeta_tmpb_tl
1578     }
1579     \prop_gput:Nno\g__pdfmeta_xmp_pdfd_data_prop{#1}
1580     {

```

```

1581         \l__pdfmeta_tmpa_tl
1582     }
1583 }
1584 \cs_generate_variant:Nn\pdfmeta_xmp_add_declaration:nnnnn {e,eee}

```

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

4.18 Default declarations

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

```

\__pdfmeta_xmp_wtpdf_reuse_declaration:
pdfmeta_xmp_wtpdf_accessibility_declaration:

```

```

1585 \cs_new:Npn \__pdfmeta_xmp_iso_today:
1586 {
1587     \int_use:N\c_sys_year_int-
1588     \int_compare:nNnT {\c_sys_month_int} < {10}{0} \int_use:N\c_sys_month_int -
1589     \int_compare:nNnT {\c_sys_day_int} < {10}{0} \int_use:N\c_sys_day_int
1590 }
1591 \cs_new_protected:Npn \__pdfmeta_xmp_wtpdf_reuse_declaration:
1592 {
1593     \pdfmeta_xmp_add_declaration:eeenn
1594     {http://pdfa.org/declarations/wtpdf\c_hash_str reuse1.0}
1595     {LaTeX-Project}
1596     {\__pdfmeta_xmp_iso_today:}{}{}
1597 }
1598 \cs_new_protected:Npn \__pdfmeta_xmp_wtpdf_accessibility_declaration:
1599 {
1600     \pdfmeta_xmp_add_declaration:ennnn
1601     {http://pdfa.org/declarations/wtpdf\c_hash_str accessibility1.0}
1602     {LaTeX-Project}
1603     {\__pdfmeta_xmp_iso_today:}{}{}
1604 }

```

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

```

1605 </package>

```

Index

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

Symbols		
<code>\&</code>	708	<code>\AddToDocumentProperties</code> . . . 510, 512, 514, 516, 518, 520, 522, 524, 527, 540
<code>\'</code>	643	<code>\AddToHook</code> . . . 315, 446, 528, 541, 543, 1517
<code>\+</code>	643	
<code>\-</code>	643, 724	B
<code>\[</code>	724	<code>BaseUrl/baseurl</code> <u>1257</u>
<code>\]</code>	724	bitset commands:
		<code>\bitset_set_false:Nn</code> 93, 94, 95
		<code>\bitset_set_true:Nn</code> 92
<code>\A</code>	724	<code>\bitset_to_arabic:N</code> 96, 97, 98, 99, 100
		A

bool commands:		\exp_args:NV	484, 487
\bool_gset_false:N	557, 592	\exp_last_unbraced:No	1291, 1293
\bool_gset_true:N	501, 556, 587, 596	\exp_not:n	745, 753, 914
\bool_if:NTF	321, 1519, 1533		
\bool_lazy_or:nnTF	604		
\bool_new:N	500, 579		
		F	
		file commands:	
		\file_get_timestamp:nN	1526
		G	
char commands:		\GetDocumentProperties	682,
\char_generate:nn	609, 614, 615, 616	862, 863, 1254, 1262, 1265, 1287,	
clist commands:		1328, 1330, 1334, 1338, 1343, 1353,	
\clist_if_empty:nTF	821, 838	1355, 1360, 1364, 1366, 1377, 1380,	
\clist_map_inline:nn	429, 825, 842	1384, 1392, 1394, 1401, 1406, 1420,	
complianceProfile	1459	1424, 1428, 1432, 1436, 1440, 1444,	
CreatorTool/pdfcreator	1257	1465, 1472, 1480, 1483, 1486, 1489,	
cs commands:		1492, 1495, 1498, 1501, 1504, 1505	
\cs_generate_variant:Nn	648,	group commands:	
720, 739, 748, 756, 762, 769, 784,		\group_begin:	325, 424, 707
794, 804, 817, 834, 859, 1561, 1584		\group_end:	330, 443, 716
\cs_gset_eq:NN	1247		
\cs_if_exist:NTF	39		
\cs_if_exist_use:N	962		
\cs_new:Npn		H	
17, 608, 612, 620, 626, 649, 1585		hook commands:	
\cs_new_protected:Npn	21,	\hook_gput_code:nnn	102, 502
56, 64, 72, 78, 84, 90, 407, 422, 498,			
632, 637, 644, 679, 692, 704, 725,			
741, 749, 757, 763, 770, 775, 785,			
795, 805, 818, 835, 860, 909, 940,			
968, 991, 1156, 1191, 1204, 1249,			
1257, 1278, 1297, 1311, 1325, 1332,			
1357, 1396, 1414, 1448, 1459, 1511,			
1541, 1548, 1556, 1562, 1591, 1598			
\cs_set_eq:NN	570, 576, 865, 1554, 1555		
		I	
		int commands:	
		\int_compare:nNnTF	
		1283, 1361, 1367, 1588, 1589	
		\int_decr:N	639
		\int_incr:N	634
		\int_new:N	619
		\int_set:Nn	902, 1515
		\int_use:N	1587, 1588, 1589
		\int_zero:N	904, 1513
		iow commands:	
\d	643	\iow_close:N	1537
dc commands:		\iow_newline:	622, 628
dc:description/pdfsubject	1357	\iow_now:Nn	1536
dc:identifier/pdfidentifier	1357	\iow_open:Nn	1535
dc:language/lang/pdflang	1357	\g_tmpa_iow	1535, 1536, 1537
dc:Ncreator/pdfauthor	1357	iptc _␣ (schema)	1146
dc:publisher/pdfpublisher	1357		
dc:subject/pdfkeywords	1357		
dc:type/pdftype	1357	J	
\DocumentMetadata	2, 4	\jobname	1338, 1347, 1526
		K	
		kernel internal commands:	
exp commands:		\g__kernel_pdfmanagement_end_-	
\exp_args:NNe	466, 471, 477	run_code_tl	319
\exp_args:Nne	329	keys commands:	
\exp_args:Nnne	41	\keys_define:nn	337, 344, 507, 564, 582
\exp_args:NNo	382, 1536	\l_keys_key_str	384
\exp_args:No	1530	\keys_set:nn	341, 561

M

msg commands:

- \msg_new:nnn 7, 8, 601, 602, 603
- \msg_warning:nnn 461, 494
- \msg_warning:nnnn 945, 988, 1551
- \msg_warning:nnnnn . 114, 124, 532, 549

P

pdf commands:

- \pdf_object_if_exist:nTF 409
- \pdf_object_new:n 411
- \pdf_object_ref:n 428
- \pdf_object_ref_last: 442, 1532
- \pdf_object_unnamed_write:nn . . 441
- \pdf_object_write:nnn 412
- \pdf_string_from_unicode:nnN . . . 436
- \pdf_version: 3, 4, 113, 115, 123, 125, 534, 551, 1255
- \pdf_version_compare:NnTF 58, 66, 530, 547

pdf internal commands:

- __pdf_backend_metadata_stream:n 1531
- __pdf_backend_Names_gpush:nn . . 329
- __pdf_backend_omit_charset:n . . 109
- __pdf_backend_omit_cidset:n . . 111
- __pdf_backend_omit_info:n 107
- __pdf_backend_set_regression_-data: 499

pdfaid~(schema) 1010

pdfannot commands:

- \pdfannot_dict_put:nnn 96, 97, 98, 99, 100
- \l_pdfannot_F_bitset 92, 93, 94, 95, 96, 97, 98, 99, 100

pdfdict commands:

- \pdfdict_if_empty:nTF 323
- \pdfdict_new:n 388
- \pdfdict_put:nnn 326, 327, 389, 425, 426, 437
- \pdfdict_use:n 441

pdffile commands:

- \pdffile_embed_stream:nnN 328

pdfmanagement commands:

- \pdfmanagement_add:nnn 442, 504, 505, 1269, 1273, 1532
- \pdfmanagement_get_documentproperties:nNTF 1288, 1369
- \pdfmanagement_remove:nn . 1362, 1368

pdfmanagement internal commands:

- \g__pdfmanagement_active_bool . . 321

pdfmeta commands:

- \pdfmeta_set_regression_data: 5, 498
- \pdfmeta_standard_get:nN . . . 2, 21, 21
- \pdfmeta_standard_item:n 2, 17, 17, 117, 119, 127, 129, 469, 474, 480, 1280, 1282, 1283, 1284, 1285, 1361, 1367
- \pdfmeta_standard_verify:n . . . 2, 25
- \pdfmeta_standard_verify:nn . . . 2, 35
- \pdfmeta_standard_verify:nnN 2
- \pdfmeta_standard_verify:nnTF 2, 35, 112, 122
- \pdfmeta_standard_verify:nTF 2, 25, 104, 106, 108, 110, 317, 448
- \pdfmeta_standard_verify_p:n . . . 2, 25
- \pdfmeta_xmp_add:n 9, 1541, 1541
- \pdfmeta_xmp_add_declaration:n 9, 1556, 1556, 1561
- \pdfmeta_xmp_add_declaration:nnnnn 9, 1562, 1562, 1584, 1593, 1600
- \pdfmeta_xmp_property_new:nnnnn 10, 1555, 1555
- \pdfmeta_xmp_schema_new:nnn 10, 1554, 1554
- \pdfmeta_xmp_xmlns_new:nn 9, 1548, 1548

pdfmeta internal commands:

- __pdfmeta_embed_colorprofile:n 407, 407, 454, 484
- \g__pdfmeta_outputintents_prop 336, 350, 358, 366, 374, 383, 450, 468, 473, 479, 485
- \g__pdfmeta_standard_pdf/A-1B_-prop 133
- \g__pdfmeta_standard_pdf/A-2A_-prop 133
- \g__pdfmeta_standard_pdf/A-2B_-prop 133
- \g__pdfmeta_standard_pdf/A-2U_-prop 133
- \g__pdfmeta_standard_pdf/A-3A_-prop 133
- \g__pdfmeta_standard_pdf/A-3B_-prop 133
- \g__pdfmeta_standard_pdf/A-3U_-prop 133
- \g__pdfmeta_standard_pdf/A-4_-prop 133
- \g__pdfmeta_standard_prop 16, 19, 23, 27, 37, 45, 542
- __pdfmeta_standard_verify_-handler_annot_action_A:nn . 78, 78
- __pdfmeta_standard_verify_-handler_max_pdf_version:nn 63, 64
- __pdfmeta_standard_verify_-handler_min_pdf_version:nn 55, 56

_pdfmeta_standard_verify_- handler_named_actions:nn .. [71](#), [72](#)
 _pdfmeta_standard_verify_- handler_outputintent_subtype:nn [84](#), [84](#)
 \l_pdfmeta_tmpa_seq [10](#), [728](#), [729](#), [735](#), [736](#), [1267](#), [1268](#), [1271](#), [1272](#), [1275](#), [1276](#), [1385](#), [1387](#)
 \g_pdfmeta_tmpa_str [13](#), [711](#), [712](#), [713](#), [714](#), [715](#), [717](#)
 \l_pdfmeta_tmpa_str [10](#), [436](#), [438](#), [780](#), [781](#), [790](#), [791](#), [800](#), [801](#), [1334](#), [1335](#), [1339](#), [1342](#), [1343](#), [1344](#), [1348](#), [1351](#)
 \l_pdfmeta_tmpa_tl [10](#), [328](#), [329](#), [434](#), [436](#), [710](#), [711](#), [810](#), [813](#), [815](#), [844](#), [845](#), [852](#), [973](#), [1267](#), [1269](#), [1271](#), [1273](#), [1275](#), [1288](#), [1291](#), [1293](#), [1369](#), [1371](#), [1385](#), [1466](#), [1469](#), [1473](#), [1476](#), [1505](#), [1508](#), [1566](#), [1577](#), [1581](#)
 \l_pdfmeta_tmpb_seq [10](#)
 \l_pdfmeta_tmpb_tl [10](#), [481](#), [482](#), [484](#), [489](#), [494](#), [844](#), [848](#), [852](#), [1466](#), [1470](#), [1473](#), [1477](#), [1575](#), [1577](#)
 _pdfmeta_verify_pdfa_annot_- flags: [90](#), [105](#)
 _pdfmeta_write_outputintent:nn [407](#), [422](#), [456](#), [488](#)
 _pdfmeta_xmp_add_packet_- chunk:n [741](#), [741](#), [748](#), [759](#), [766](#), [773](#), [781](#), [801](#), [871](#), [903](#), [905](#), [1545](#)
 _pdfmeta_xmp_add_packet_- chunk:nN [749](#), [749](#), [756](#), [791](#)
 _pdfmeta_xmp_add_packet_- close:nn ... [770](#), [770](#), [830](#), [831](#), [855](#), [856](#), [884](#), [885](#), [899](#), [900](#), [901](#), [960](#), [961](#), [963](#), [983](#), [998](#), [1185](#), [1186](#), [1187](#), [1188](#), [1189](#), [1225](#), [1226](#), [1227](#), [1241](#), [1242](#), [1243](#), [1244](#), [1245](#), [1307](#), [1308](#), [1320](#), [1321](#), [1323](#), [1455](#), [1573](#)
 _pdfmeta_xmp_add_packet_- field:nnn [991](#), [991](#), [1169](#), [1171](#), [1173](#), [1175](#), [1177](#), [1179](#), [1181](#), [1183](#), [1217](#), [1219](#), [1221](#), [1223](#), [1237](#), [1239](#)
 _pdfmeta_xmp_add_packet_- line:nnn [775](#), [775](#), [784](#), [815](#), [827](#), [954](#), [955](#), [956](#), [979](#), [980](#), [981](#), [982](#), [995](#), [996](#), [997](#), [1161](#), [1162](#), [1164](#), [1165](#), [1209](#), [1210](#), [1212](#), [1213](#), [1229](#), [1230](#), [1232](#), [1233](#), [1255](#), [1268](#), [1272](#), [1276](#), [1280](#), [1281](#), [1284](#), [1285](#), [1286](#), [1290](#), [1292](#), [1314](#), [1327](#), [1329](#), [1341](#), [1350](#), [1352](#), [1354](#), [1383](#), [1388](#), [1398](#), [1402](#), [1461](#), [1478](#), [1481](#), [1484](#), [1487](#), [1490](#), [1493](#), [1496](#), [1499](#), [1502](#), [1506](#), [1569](#), [1570](#), [1571](#), [1572](#)
 _pdfmeta_xmp_add_packet_- line:nnnN . [785](#), [785](#), [794](#), [1418](#), [1422](#), [1426](#), [1430](#), [1434](#), [1438](#), [1442](#)
 _pdfmeta_xmp_add_packet_line_- attr:nnnn [795](#), [795](#), [804](#), [847](#), [851](#), [1467](#), [1474](#)
 _pdfmeta_xmp_add_packet_line_- default:nnnn [805](#), [805](#), [817](#), [1251](#), [1259](#), [1263](#), [1389](#)
 _pdfmeta_xmp_add_packet_- list:nnnn [835](#), [859](#), [1359](#), [1363](#), [1365](#), [1376](#), [1378](#), [1393](#)
 _pdfmeta_xmp_add_packet_list_- simple:nnnn [818](#), [834](#), [1371](#), [1374](#), [1381](#), [1386](#)
 _pdfmeta_xmp_add_packet_- open:nn [757](#), [757](#), [762](#), [823](#), [824](#), [840](#), [841](#), [873](#), [874](#), [878](#), [879](#), [957](#), [958](#), [978](#), [1158](#), [1159](#), [1167](#), [1168](#), [1206](#), [1207](#), [1215](#), [1216](#), [1235](#), [1236](#), [1301](#), [1302](#), [1317](#), [1318](#)
 _pdfmeta_xmp_add_packet_open_- attr:nnn [763](#), [763](#), [769](#), [876](#), [953](#), [994](#), [1160](#), [1208](#), [1228](#), [1313](#), [1452](#), [1568](#)
 \g_pdfmeta_xmp_bool [500](#), [556](#), [557](#), [1519](#)
 _pdfmeta_xmp_build_dc: [891](#), [1357](#), [1357](#)
 _pdfmeta_xmp_build_iptc: [896](#), [1448](#), [1448](#)
 _pdfmeta_xmp_build_iptc_data:N [866](#), [1414](#), [1414](#)
 _pdfmeta_xmp_build_packet: ... [860](#), [860](#), [1529](#)
 _pdfmeta_xmp_build_pdf: [887](#), [1249](#), [1249](#)
 _pdfmeta_xmp_build_pdfd: [890](#), [1297](#), [1297](#)
 _pdfmeta_xmp_build_pdfd_- claim:nn [1305](#), [1311](#), [1311](#)
 _pdfmeta_xmp_build_photoshop: . [892](#), [1325](#), [1325](#)
 _pdfmeta_xmp_build_prism: [895](#), [1459](#), [1459](#)
 _pdfmeta_xmp_build_standards: . [889](#), [1278](#), [1278](#)
 _pdfmeta_xmp_build_user: [897](#), [1511](#), [1511](#)
 _pdfmeta_xmp_build_xmp: [893](#), [1257](#), [1257](#)
 _pdfmeta_xmp_build_xmpMM: [894](#), [1332](#), [1332](#)

_pdfmeta_xmp_build_xmpRights: .	888, 1396 , 1396
_pdfmeta_xmp_create_uuid:nN . . .	692, 692, 1337, 1346
\l_pdfmeta_xmp_currentdate_seq .	677, 685, 1528
\l_pdfmeta_xmp_currentdate_tl . .	677, 686, 1347, 1523, 1526, 1528
_pdfmeta_xmp_date_get:nNN	679, 679, 1266, 1270, 1274, 1385
\l_pdfmeta_xmp_date_regex .	641 , 646
_pdfmeta_xmp_date_split:nN	644 , 644, 648, 689, 1528
_pdfmeta_xmp_decr_indent:	620 , 637, 772, 1446
\l_pdfmeta_xmp_doclang_tl	721 , 862, 865, 1382
\g_pdfmeta_xmp_export_bool	579, 587, 592, 596, 1533
\g_pdfmeta_xmp_export_str	580, 588, 597, 1535
_pdfmeta_xmp_generate_bom:	604 , 608, 612, 872
_pdfmeta_xmp_incr_indent:	620 , 632, 760, 767, 1417
_pdfmeta_xmp_indent:	620 , 620, 745, 753
_pdfmeta_xmp_indent:n	620 , 626, 914
\l_pdfmeta_xmp_indent_int .	619 , 623, 634, 639, 902, 904, 1513, 1515
\l_pdfmeta_xmp_iptc_data_tl	866, 867, 1413 , 1450, 1454
_pdfmeta_xmp_iso_today:	1585, 1596, 1603
_pdfmeta_xmp_lang_get:nNN	725, 739, 844, 1464, 1471
\l_pdfmeta_xmp_lang_regex .	723 , 728
\l_pdfmeta_xmp_metalang_tl	721 , 731, 845, 863, 864, 865
\g_pdfmeta_xmp_packet_tl	740 , 743, 1454, 1531, 1536
\g_pdfmeta_xmp_pdfd_data_prop	1296 , 1299, 1303, 1559, 1575, 1579
_pdfmeta_xmp_print_date:N	649 , 649, 1268, 1272, 1276, 1387
_pdfmeta_xmp_property_new:nnnnn	967, 968, 1004, 1014, 1020, 1030, 1036, 1046, 1056, 1062, 1068, 1074, 1080, 1086, 1092, 1098, 1104, 1110, 1116, 1122, 1128, 1134, 1140, 1150, 1198, 1555
_pdfmeta_xmp_sanitize:nN	704 , 704, 720, 780, 790, 800
_pdfmeta_xmp_schema_enable_pdfd:	1191, 1247, 1558, 1565
_pdfmeta_xmp_schema_new:nnn	940 , 940, 1000, 1010, 1026, 1042, 1052, 1146, 1194, 1554
\g_pdfmeta_xmp_schema_property_prop	967, 973, 975
\l_pdfmeta_xmp_schema_seq	869, 880, 939 , 948
\g_pdfmeta_xmp_user_packet_str	1510
\g_pdfmeta_xmp_user_packet_tl	1510, 1514, 1543
_pdfmeta_xmp_wtpdf_accessibility_declaration:	545, 573, 576, 1585 , 1598
_pdfmeta_xmp_wtpdf_reuse_declaration:	546, 567, 570, 1585 , 1591
_pdfmeta_xmp_xmlns_new:nn	909 , 909, 917, 918, 919, 920, 921, 922, 923, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 1193, 1552
\g_pdfmeta_xmp_xmlns_prop	907, 911, 1550
\g_pdfmeta_xmp_xmlns_tl	877 , 907 , 912
pdfmetatmpa internal commands:	
\g_pdfmetatmpa_str	10
pdfuaid~(schema)	1026
PDFversion	1249
pdfxid~(schema)	1042
photoshop commands:	
photoshop:AuthorsPosition/pdfauthortitle	1357
photoshop:CaptionWriter/pdfcaptionwriter	1357
\PreviousTotalPages	1508
prg commands:	
\prg_do_nothing:	570, 576, 1247
\prg_new_conditional:Npnn	25
\prg_new_protected_conditional:Npnn	35
\prg_replicate:nn	623, 629, 903
\prg_return_false:	29, 48, 60, 68, 76, 82, 88
\prg_return_true:	32, 52, 61, 69, 75, 81, 87
prism commands:	
prism:subtitle/pdfsubtitle	1459
prism~(schema)	1052
Producer/pdfproducer	1249
prop commands:	
\prop_const_from_keyval:Nn	391, 398
\prop_get:NnN	23, 478

<code>\prop_get:NnNTF</code>	431, 973, 1575	<code>\str_new:N</code>	12, 13, 580, 907
<code>\prop_gput:Nnn</code>	199, 201, 203, 209, 211, 223, 225, 227, 235, 237, 239, 248, 250, 252, 263, 265, 267, 275, 277, 279, 287, 289, 291, 293, 295, 297, 310, 313, 350, 358, 366, 374, 383, 472, 542, 911, 975, 1559, 1579	<code>\str_range:Nnn</code>	697, 698, 699, 700, 701
<code>\prop_gremove:Nn</code>	206, 214, 255, 299, 301, 303	<code>\str_set:Nn</code>	694, 695, 1334, 1343
<code>\prop_gset_eq:NN</code>	196, 220, 232, 245, 260, 272, 284, 307	<code>\str_set_eq:NN</code>	717
<code>\prop_gset_from_keyval:Nn</code>	134	<code>\c_tilde_str</code>	709
<code>\prop_if_empty:NTF</code>	1299	sys commands:	
<code>\prop_if_exist:NTF</code>	452, 482	<code>\c_sys_day_int</code>	1589
<code>\prop_if_in:NnTF</code>	27, 37, 467, 1550	<code>\c_sys_engine_exec_str</code>	504, 1253
<code>\prop_item:Nn</code>	19, 45, 415	<code>\c_sys_engine_version_str</code>	504, 1253
<code>\prop_map_inline:Nn</code>	450, 485, 1303	<code>\sys_if_engine luatex_p:</code>	605
<code>\prop_new:N</code>	16, 133, 195, 219, 231, 244, 259, 271, 283, 306, 336, 908, 967, 1296	<code>\sys_if_engine xetex_p:</code>	606
<code>\ProvidesExplPackage</code>	3	<code>\c_sys_jobname_str</code>	588, 1391
R			
regex commands:			
<code>\regex_extract_once:NnN</code>	728	<code>\c_sys_month_int</code>	1588
<code>\regex_new:N</code>	641, 723	<code>\c_sys_timestamp_str</code>	5, 1521, 1523
<code>\regex_set:Nn</code>	642, 724	<code>\c_sys_year_int</code>	1587
<code>\regex_split:NnN</code>	646	T	
S			
seq commands:			
<code>\seq_if_empty:NTF</code>	729	tex commands:	
<code>\seq_item:Nn</code>	651, 653, 655, 657, 659, 660, 662, 663, 665, 666, 667, 668, 670, 671, 674, 735, 736	<code>\tex_mdffivesum:D</code>	694
<code>\seq_map_inline:Nn</code>	880	text commands:	
<code>\seq_new:N</code>	14, 15, 678, 939	<code>\text_declare_purify_equivalent:Nn</code>	708, 709
<code>\seq_put_right:Nn</code>	948	<code>\text_purify:n</code>	710
<code>\seq_remove_all:Nn</code>	869	<code>\texttilde</code>	709
<code>\seq_set_eq:NN</code>	685	tl commands:	
str commands:			
<code>\c_hash_str</code>	9, 875, 924, 932, 935, 936, 937, 938, 1594, 1601	<code>\c_space_tl</code>	414, 623, 629
<code>\str_case:nn</code>	1406	<code>\tl_clear:N</code>	1416
<code>\str_convert_pdfname:n</code>	425	<code>\tl_concat:NNN</code>	1577
<code>\str_greplace_all:Nnn</code>	712, 713, 714, 715	<code>\tl_gput_right:Nn</code>	319, 743, 912, 951, 976, 1454, 1543
<code>\str_gset:Nn</code>	597, 711	<code>\tl_if_blank:nTF</code>	348, 356, 364, 372, 380, 651, 658, 661, 664, 669, 683, 778, 788, 798, 808, 864, 1508
<code>\str_gset_eq:NN</code>	588	<code>\tl_if_empty:NTF</code>	867, 1450
<code>\str_if_empty:NTF</code>	1335, 1344	<code>\tl_if_empty:nTF</code>	1315
<code>\str_if_exist:NTF</code>	1521	<code>\tl_if_eq:nnTF</code>	86, 845
<code>\str_lowercase:n</code>	694	<code>\tl_if_exist:NTF</code>	943, 971
		<code>\tl_if_in:nnTF</code>	74, 80
		<code>\tl_new:N</code>	10, 11, 677, 721, 722, 740, 949, 950, 1413, 1510
		<code>\tl_put_right:Nn</code>	751
		<code>\tl_set:Nn</code>	682, 710, 731, 732, 735, 736, 810, 813, 862, 863, 1505, 1566
		<code>\tl_set_eq:NN</code>	686, 1523
		<code>\tl_to_str:N</code>	708, 711
		<code>\tl_use:N</code>	882, 959
U			
use commands:			
<code>\use:N</code>	42		
<code>\use_i:nn</code>	1291		
<code>\use_ii:nn</code>	1293		