

The `l3pdfmeta` module

PDF standards

L^AT_EX PDF management testphase bundle

The L^AT_EX Project*

Version 0.96i, released 2024-05-23

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 an `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_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)

`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
  }
}
```

`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.

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

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.

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.

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.

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

`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 datas 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/>.

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`.

⁵with a number of changes which are discussed in more details below

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

2.3.2 Declarations

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

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

2.3.3 Dates

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

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

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

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

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

```

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

```

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

2.4 Language

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

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

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

```

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

```

2.5 Rights

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

```

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

```

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.

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

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.

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.

3 l3pdfmeta implementation

```
1 <@@=pdfmeta>
2 <*header>
3 \ProvidesExplPackage{l3pdfmeta}{2024-05-23}{0.96i}
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:nTF` 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 to 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:nnF { min_pdf_version }
111   { \pdf_version: }
112   { \msg_warning:nneee {pdf}{wrong-pdfversion}
113     {\pdf_version:}{low}
114     {
115       \pdfmeta_standard_item:n{type}
116       -
117       \pdfmeta_standard_item:n{level}
118     }
119   }
120   \pdfmeta_standard_verify:nnF { max_pdf_version }
121   { \pdf_version: }
122   { \msg_warning:nneee {pdf}{wrong-pdfversion}
123     {\pdf_version:}{high}
124     {
125       \pdfmeta_standard_item:n{type}
126       -
127       \pdfmeta_standard_item:n{level}
128     }
129   }
130 }
```

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
131 \prop_new:c { g__pdfmeta_standard_pdf/A-1B_prop }
132 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/A-1B_prop }
133 {
134   ,name           = pdf/A-1B
135   ,type           = A
136   ,level          = 1
137   ,conformance    = B
138   ,year           = 2005
139   ,min_pdf_version = 1.4      %minimum
140   ,max_pdf_version = 1.4      %minimum
141   ,no_encryption  =
142   ,no_external_content = % no F, FFilter, or FDecodeParms in stream dicts
143   ,no_embed_content = % no EF key in filespec, no /Type/EmbeddedFiles
144   ,max_string_size = 65535
145   ,max_array_size  = 8191
146   ,max_dict_size   = 4095
147   ,max_obj_num     = 8388607
```

```

148     ,max_nest_qQ      = 28
149     ,named_actions   = {NextPage, PrevPage, FirstPage, LastPage}
150     ,annot_flags     =
151     %booleans. Only the existence of the key matter.
152     %If the entry is added it means a requirements is there
153     %(in most cases "don't use ...")
154     %
155     %=====
156     % Rule 6.1.13-1 CosDocument, isOptionalContentPresent == false
157     ,Catalog_no_OCProperties =
158     %=====
159     % Rule 6.6.1-1: PDAction, S == "GoTo" || S == "GoToR" || S == "Thread"
160     %                || S == "URI" || S == "Named" || S == "SubmitForm"
161     % means: no /S/Launch, /S/Sound, /S/Movie, /S/ResetForm, /S/ImportData,
162     %                /S/JavaScript, /S/Hide
163     ,annot_action_A   = {GoTo,GoToR,Thread,URI,Named,SubmitForm}
164     %=====
165     % Rule 6.6.2-1: PDAnnot, Subtype != "Widget" || AA_size == 0
166     % means: no AA dictionary
167     ,annot_widget_no_AA =
168     %=====
169     % Rule 6.9-2: PDAnnot, Subtype != "Widget" || (A_size == 0 && AA_size == 0)
170     % (looks like a tightening of the previous rule)
171     ,annot_widget_no_A_AA =
172     %=====
173     % Rule 6.9-1 PDAcroForm, NeedAppearances == null || NeedAppearances == false
174     ,form_no_NeedAppearances =
175     %=====
176     %Rule 6.9-3 PDFFormField, AA_size == 0
177     ,form_no_AA      =
178     %=====
179     % to be continued https://docs.verapdf.org/validation/pdfa-part1/
180     % - Outputintent/colorprofiles requirements
181     % an outputintent should be loaded and is unique.
182     ,outputintent_A   = {GTS_PDFA1}
183     % - no Alternates key in image dictionaries
184     % - no OPI, Ref, Subtype2 with PS key in xobjects
185     % - Interpolate = false in images
186     % - no TR, TR2 in ExtGstate
187 }
188
189 %A-2b =====
190 \prop_new:c { g__pdfmeta_standard_pdf/A-2B_prop }
191 \prop_gset_eq:cc
192   { g__pdfmeta_standard_pdf/A-2B_prop }
193   { g__pdfmeta_standard_pdf/A-1B_prop }
194 \prop_gput:cnn
195   { g__pdfmeta_standard_pdf/A-2B_prop }{name}{pdf/A-2B}
196 \prop_gput:cnn
197   { g__pdfmeta_standard_pdf/A-2B_prop }{year}{2011}
198 \prop_gput:cnn
199   { g__pdfmeta_standard_pdf/A-2B_prop }{level}{2}
200 % embedding files is allowed (with restrictions)
201 \prop_gremove:cn

```

```

202 { g__pdfmeta_standard_pdf/A-2B_prop }
203 { embed_content}
204 \prop_gput:cnn
205 { g__pdfmeta_standard_pdf/A-2B_prop }{max_pdf_version}{1.7}
206 %A-2u =====
207 \prop_new:c { g__pdfmeta_standard_pdf/A-2U_prop }
208 \prop_gset_eq:cc
209 { g__pdfmeta_standard_pdf/A-2U_prop }
210 { g__pdfmeta_standard_pdf/A-2B_prop }
211 \prop_gput:cnn
212 { g__pdfmeta_standard_pdf/A-2U_prop }{name}{pdf/A-2U}
213 \prop_gput:cnn
214 { g__pdfmeta_standard_pdf/A-2U_prop }{conformance}{U}
215 \prop_gput:cnn
216 { g__pdfmeta_standard_pdf/A-2U_prop }{unicode}{f}
217
218 %A-2a =====
219 \prop_new:c { g__pdfmeta_standard_pdf/A-2A_prop }
220 \prop_gset_eq:cc
221 { g__pdfmeta_standard_pdf/A-2A_prop }
222 { g__pdfmeta_standard_pdf/A-2B_prop }
223 \prop_gput:cnn
224 { g__pdfmeta_standard_pdf/A-2A_prop }{name}{pdf/A-2A}
225 \prop_gput:cnn
226 { g__pdfmeta_standard_pdf/A-2A_prop }{conformance}{A}
227 \prop_gput:cnn
228 { g__pdfmeta_standard_pdf/A-2A_prop }{tagged}{f}
229
230
231 %A-3b =====
232 \prop_new:c { g__pdfmeta_standard_pdf/A-3B_prop }
233 \prop_gset_eq:cc
234 { g__pdfmeta_standard_pdf/A-3B_prop }
235 { g__pdfmeta_standard_pdf/A-2B_prop }
236 \prop_gput:cnn
237 { g__pdfmeta_standard_pdf/A-3B_prop }{name}{pdf/A-3B}
238 \prop_gput:cnn
239 { g__pdfmeta_standard_pdf/A-3B_prop }{year}{2012}
240 \prop_gput:cnn
241 { g__pdfmeta_standard_pdf/A-3B_prop }{level}{3}
242 % embedding files is allowed (with restrictions)
243 \prop_gremove:cn
244 { g__pdfmeta_standard_pdf/A-3B_prop }
245 { embed_content}
246 %A-3u =====
247 \prop_new:c { g__pdfmeta_standard_pdf/A-3U_prop }
248 \prop_gset_eq:cc
249 { g__pdfmeta_standard_pdf/A-3U_prop }
250 { g__pdfmeta_standard_pdf/A-3B_prop }
251 \prop_gput:cnn
252 { g__pdfmeta_standard_pdf/A-3U_prop }{name}{pdf/A-3U}
253 \prop_gput:cnn
254 { g__pdfmeta_standard_pdf/A-3U_prop }{conformance}{U}
255 \prop_gput:cnn

```

```

256 { g__pdfmeta_standard_pdf/A-3U_prop }{unicode}{ }
257
258 %A-3a =====
259 \prop_new:c { g__pdfmeta_standard_pdf/A-3A_prop }
260 \prop_gset_eq:cc
261 { g__pdfmeta_standard_pdf/A-3A_prop }
262 { g__pdfmeta_standard_pdf/A-3B_prop }
263 \prop_gput:cnn
264 { g__pdfmeta_standard_pdf/A-3A_prop }{name}{pdf/A-3A}
265 \prop_gput:cnn
266 { g__pdfmeta_standard_pdf/A-3A_prop }{conformance}{A}
267 \prop_gput:cnn
268 { g__pdfmeta_standard_pdf/A-3A_prop }{tagged}{ }
269
270 %A-4 =====
271 \prop_new:c { g__pdfmeta_standard_pdf/A-4_prop }
272 \prop_gset_eq:cc
273 { g__pdfmeta_standard_pdf/A-4_prop }
274 { g__pdfmeta_standard_pdf/A-3U_prop }
275 \prop_gput:cnn
276 { g__pdfmeta_standard_pdf/A-4_prop }{name}{pdf/A-4}
277 \prop_gput:cnn
278 { g__pdfmeta_standard_pdf/A-4_prop }{level}{4}
279 \prop_gput:cnn
280 { g__pdfmeta_standard_pdf/A-4_prop }{min_pdf_version}{2.0}
281 \prop_gput:cnn
282 { g__pdfmeta_standard_pdf/A-4_prop }{year}{2020}
283 \prop_gput:cnn
284 { g__pdfmeta_standard_pdf/A-4_prop }{no_CharSet}{ }
285 \prop_gput:cnn
286 { g__pdfmeta_standard_pdf/A-4_prop }{Trailer_no_Info}{ }
287 \prop_gremove:cn
288 { g__pdfmeta_standard_pdf/A-4_prop }{conformance}
289 \prop_gremove:cn
290 { g__pdfmeta_standard_pdf/A-4_prop }{max_pdf_version}
291
292 %A-4f =====
293 \prop_new:c { g__pdfmeta_standard_pdf/A-4F_prop }
294 \prop_gset_eq:cc
295 { g__pdfmeta_standard_pdf/A-4F_prop }
296 { g__pdfmeta_standard_pdf/A-4_prop }
297 \prop_gput:cnn
298 { g__pdfmeta_standard_pdf/A-4F_prop }{conformance}{F}
299 % containsEmbeddedFiles == true ISO 19005-4:2020, Clause: 6.9, Test number: 5
300 \prop_gput:cnn
301 { 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


```

302 \AddToHook{begindocument/end}
303 {
304   \pdfmeta_standard_verify:nF{Catalog_EmbeddedFiles}
305   {
306     \tl_gput_right:Nn\g__kernel_pdfmanagement_end_run_code_tl
307     {
308       \bool_if:NT \g__pdfmanagement_active_bool
309       {
310         \pdfdict_if_empty:nT { g__pdf_Core/Catalog/Names/EmbeddedFiles }
311         {
312           \group_begin:
313           \pdfdict_put:nne {l_pdffile/Filespec} {Desc}{(note~about~PDF/A-4F)}
314           \pdfdict_put:nnn { l_pdffile/Filespec }{AFRelationship} { /Unspecified }
315           \pdffile_embed_stream:nnN {PDF~standard~A-4F~requires~a~file}{readme.txt}\l__pdf
316           \exp_args:Nne \__pdf_backend_Names_gpush:nn{EmbeddedFiles}{(readme)~\l__pdfmeta_
317           \group_end:
318         }
319       }
320     }
321   }
322 }

```

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.

```
323 \prop_new:N \g__pdfmeta_outputintents_prop
```

(End of definition for `\g__pdfmeta_outputintents_prop`.)

Some keys to fill the property.

```
324 \keys_define:nn { document / metadata }
325 {
326   colorprofiles .code:n =
327   {
328     \keys_set:nn { document / metadata / colorprofiles }{#1}
329   }
330 }
331 \keys_define:nn { document / metadata / colorprofiles }
332 {
333   ,A .code:n =
334   {
335     \tl_if_blank:nF {#1}
336     {
337       \prop_gput:Nnn \g__pdfmeta_outputintents_prop
338       { GTS_PDFA1 } {#1}
339     }
340   }
341   ,a .code:n =
342   {
343     \tl_if_blank:nF {#1}
344     {
345       \prop_gput:Nnn \g__pdfmeta_outputintents_prop
346       { GTS_PDFA1 } {#1}
347     }
348   }
349   ,X .code:n =
350   {
351     \tl_if_blank:nF {#1}
352     {
353       \prop_gput:Nnn \g__pdfmeta_outputintents_prop
354       { GTS_PDFX } {#1}
355     }
356   }
357   ,x .code:n =
358   {
359     \tl_if_blank:nF {#1}
360     {
361       \prop_gput:Nnn \g__pdfmeta_outputintents_prop
362       { GTS_PDFX } {#1}
363     }
364   }
365   ,unknown .code:n =
366   {
367     \tl_if_blank:nF {#1}
368     {
369       \exp_args:NNo
370       \prop_gput:Nnn \g__pdfmeta_outputintents_prop
371       { \l_keys_key_str } {#1}
372     }
373   }
374 }
```

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

```

375 \pdfdict_new:n {l_pdfmeta/outputintent}
376 \pdfdict_put:nnn {l_pdfmeta/outputintent}
377 {Type}{/OutputIntent}
378 \prop_const_from_keyval:cn { c__pdfmeta_colorprofile_sRGB.icc}
379 {
380 ,OutputConditionIdentifier=IEC~sRGB
381 ,Info=IEC-61966-2.1-Default~RGB~colour~space~~sRGB
382 ,RegistryName=http://www.iec.ch
383 ,N = 3
384 }
385 \prop_const_from_keyval:cn { c__pdfmeta_colorprofile_FOGRA39L_coated.icc}
386 {
387 ,OutputConditionIdentifier=FOGRA39L~Coated
388 ,Info={Offset~printing,~according~to~ISO~12647-2:2004/Amd-1,~OFCOM,~ %
389 paper~type~1~or~2~==~coated~art,~115~g/m2,~tone~value~increase~
390 curves~A~(CMY)~and~B~(K)}
391 ,RegistryName=http://www.fogra.org
392 ,N = 4
393 }

```

__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 full proof if pathes are used.

```

394 \cs_new_protected:Npn \__pdfmeta_embed_colorprofile:n #1%#1 file name
395 {
396   \pdf_object_if_exist:nF { __color_icc_ #1 }
397   {
398     \pdf_object_new:n { __color_icc_ #1 }
399     \pdf_object_write:nne { __color_icc_ #1 } { fstream }
400     {
401       {/N\c_space_tl
402         \prop_item:cn{c__pdfmeta_colorprofile_#1}{N}
403       }
404       {#1}
405     }
406   }
407 }
408
409 \cs_new_protected:Npn \__pdfmeta_write_outputintent:nn #1 #2 %#1 file name, #2 subtype
410 {
411   \group_begin:
412   \pdfdict_put:nne {l_pdfmeta/outputintent}{S}{/\str_convert_pdfname:n{#2}}
413   \pdfdict_put:nne {l_pdfmeta/outputintent}
414   {DestOutputProfile}
415   {\pdf_object_ref:n{ __color_icc_ #1 }}
416   \clist_map_inline:nn { OutputConditionIdentifier, Info, RegistryName }
417   {
418     \prop_get:cnNT
419     { c__pdfmeta_colorprofile_#1}
420     { ##1 }

```

```

421     \l__pdfmeta_tmpa_tl
422     {
423       \pdf_string_from_unicode:nVN {utf8/string}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_st
424       \pdfdict_put:nne
425       {l__pdfmeta/outputintent}{##1}{\l__pdfmeta_tmpa_str}
426     }
427   }
428   \pdf_object_unnamed_write:ne {dict}{\pdfdict_use:n {l__pdfmeta/outputintent} }
429   \pdfmanagement_add:nne {Catalog}{OutputIntents}{\pdf_object_ref_last:}
430 \group_end:
431 }

```

(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

```

432
433 \AddToHook{begindocument/end}
434 {
435   \pdfmeta_standard_verify:nTF {outputintent_A}
436   {
437     \prop_map_inline:Nn \g__pdfmeta_outputintents_prop
438     {
439       \prop_if_exist:cTF {c__pdfmeta_colorprofile_#2}
440       {
441         \__pdfmeta_embed_colorprofile:n
442         {#2}
443         \__pdfmeta_write_outputintent:nn
444         {#2}
445         {#1}
446       }
447       {
448         \msg_warning:nnn{pdfmeta}{colorprofile-undefined}{#2}
449       }
450     }
451   }

```

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.

```

452   {
453     \exp_args:NNe
454     \prop_if_in:NnF
455     \g__pdfmeta_outputintents_prop
456     { \pdfmeta_standard_item:n { outputintent_A } }
457     {
458       \exp_args:NNe
459       \prop_gput:Nnn
460       \g__pdfmeta_outputintents_prop
461       { \pdfmeta_standard_item:n { outputintent_A } }
462       { sRGB.icc }
463     }
464     \exp_args:NNe
465     \prop_get:NnN
466     \g__pdfmeta_outputintents_prop
467     { \pdfmeta_standard_item:n { outputintent_A } }

```

```

468     \l__pdfmeta_tmpb_tl
469     \prop_if_exist:cTF {c__pdfmeta_colorprofile_\l__pdfmeta_tmpb_tl}
470     {
471         \exp_args:NV \__pdfmeta_embed_colorprofile:n \l__pdfmeta_tmpb_tl
472         \prop_map_inline:Nn \g__pdfmeta_outputintents_prop
473         {
474             \exp_args:NV
475             \__pdfmeta_write_outputintent:nn
476             \l__pdfmeta_tmpb_tl
477             { #1 }
478         }
479     }
480     {
481         \msg_warning:nne{pdfmeta}{colorprofile-undefined}{\l__pdfmeta_tmpb_tl}
482     }
483 }
484 }

```

3.2 Regression test

This is simply a copy of the backend function.

```

485 \cs_new_protected:Npn \pdfmeta_set_regression_data:
486 { \__pdf_backend_set_regression_data: }

```

4 XMP-Metadata implementation

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

```

487 \bool_new:N\g__pdfmeta_xmp_bool
488 \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.

```

489 \hook_gput_code:nnn{pdfmanagement/add}{pdfmanagement}
490 {
491     \pdfmanagement_add:nne {Info}{Producer}{(\c_sys_engine_exec_str-\c_sys_engine_version_str)}
492     \pdfmanagement_add:nne {Info}{Creator}{(LaTeX)}
493 }

```

4.1 New document keys

```

494 \keys_define:nn { document / metadata }
495 {
496     _pdfstandard / X-4 .code:n =
497     {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-4}},
498     _pdfstandard / X-4p .code:n =
499     {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-4p}},
500     _pdfstandard / X-5g .code:n =
501     {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5g}},
502     _pdfstandard / X-5n .code:n =
503     {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5n}},
504     _pdfstandard / X-5pg .code:n =
505     {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5pg}},

```

```

506 _pdfstandard / X-6 .code:n =
507   {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6p}},
508 _pdfstandard / X-6n .code:n =
509   {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6n}},
510 _pdfstandard / X-6p .code:n =
511   {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6p}},
512 _pdfstandard / UA-1 .code:n =
513   {
514     \AddToDocumentProperties [document]{pdfstandard-UA}{{1}}
515   },

```

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.

```

516 _pdfstandard / UA-2 .code:n =
517   {
518     \AddToDocumentProperties [document]{pdfstandard-UA}{{2}}{2024}
519     \AddToHook{begindocument/before}
520     {\prop_gput:Nnn \g__pdfmeta_standard_prop {Trailer_no_Info}}
521     \AddToHook{begindocument/before}
522     {
523       \__pdfmeta_xmp_wtpdf_accessibility_declaration:
524       \__pdfmeta_xmp_wtpdf_reuse_declaration:
525     }
526   },
527 xmp .choice:,
528 xmp / true .code:n = { \bool_gset_true:N \g__pdfmeta_xmp_bool },
529 xmp / false .code:n = { \bool_gset_false:N \g__pdfmeta_xmp_bool},
530 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.

```

531 xmp / wtpdf .code:n =
532   {
533     \keys_set:nn {__pdfmeta/xmp}{{#1}}
534   },
535 }
536 \keys_define:nn {__pdfmeta/xmp}
537 {
538   reuse .choice:,
539   reuse / true .code:n = \__pdfmeta_xmp_wtpdf_reuse_declaration:,
540   reuse / false .code:n =
541     {
542       \cs_set_eq:NN \__pdfmeta_xmp_wtpdf_reuse_declaration: \prg_do_nothing:
543     },
544   accessibility .choice:,
545   accessibility / true .code:n = \__pdfmeta_xmp_wtpdf_accessibility_declaration:,
546   accessibility /false .code:n =
547     {
548       \cs_set_eq:NN \__pdfmeta_xmp_wtpdf_accessibility_declaration: \prg_do_nothing:
549     },
550 }

```

XMP debugging option

```

551 \bool_new:N \g__pdfmeta_xmp_export_bool
552 \str_new:N \g__pdfmeta_xmp_export_str

```

```

553
554 \keys_define:nn { document / metadata }
555 {
556 ,debug / xmp-export .choice:
557 ,debug / xmp-export / true .code:n=
558 {
559   \bool_gset_true:N \g__pdfmeta_xmp_export_bool
560   \str_gset_eq:NN \g__pdfmeta_xmp_export_str \c_sys_jobname_str
561 }
562 ,debug / xmp-export / false .code:n =
563 {
564   \bool_gset_false:N \g__pdfmeta_xmp_export_bool
565 }
566 ,debug / xmp-export /unknown .code:n =
567 {
568   \bool_gset_true:N \g__pdfmeta_xmp_export_bool
569   \str_gset:Nn \g__pdfmeta_xmp_export_str { #1 }
570 }
571 ,debug / xmp-export .default:n = true
572 }

```

4.2 Messages

```

573 \msg_new:nnn{pdfmeta}{namespace-defined}{The~xmlns~namespace~‘#1’~is~already~declared}
574 \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:`

```

575 \bool_lazy_or:nnTF
576 { \sys_if_engine luatex_p: }
577 { \sys_if_engine xetex_p: }
578 {
579   \cs_new:Npn \__pdfmeta_xmp_generate_bom:
580     { \char_generate:nn {"FEFF"}{12} }
581 }
582 {
583   \cs_new:Npn \__pdfmeta_xmp_generate_bom:
584     {
585       \char_generate:nn {"EF"}{12}
586       \char_generate:nn {"BB"}{12}
587       \char_generate:nn {"BF"}{12}
588     }
589 }

```

(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`

```

590 \int_new:N \l__pdfmeta_xmp_indent_int

```

(End of definition for \l__pdfmeta_xmp_indent_int.)

```
\__pdfmeta_xmp_indent:
\__pdfmeta_xmp_indent:n 591 \cs_new:Npn \__pdfmeta_xmp_indent:
\__pdfmeta_xmp_incr_indent: 592 {
\__pdfmeta_xmp_decr_indent: 593   \iow_newline:
594   \prg_replicate:nn {\l__pdfmeta_xmp_indent_int}{\c_space_tl}
595 }
596
597 \cs_new:Npn \__pdfmeta_xmp_indent:n #1
598 {
599   \iow_newline:
600   \prg_replicate:nn {#1}{\c_space_tl}
601 }
602
603 \cs_new_protected:Npn \__pdfmeta_xmp_incr_indent:
604 {
605   \int_incr:N \l__pdfmeta_xmp_indent_int
606 }
607
608 \cs_new_protected:Npn \__pdfmeta_xmp_decr_indent:
609 {
610   \int_decr:N \l__pdfmeta_xmp_indent_int
611 }
```

(End of definition for __pdfmeta_xmp_indent: and others.)

4.3.3 Date and time handling

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

```
\l__pdfmeta_xmp_date_regex
612 \regex_new:N \l__pdfmeta_xmp_date_regex
613 \regex_set:Nn \l__pdfmeta_xmp_date_regex
614 {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.

```
615 \cs_new_protected:Npn \__pdfmeta_xmp_date_split:nN #1 #2 %#1 date, #2 seq
616 {
617   \regex_split:NnN \l__pdfmeta_xmp_date_regex {#1} #2
618 }
619 \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.

```
620 \cs_new:Npn \__pdfmeta_xmp_print_date:N #1 % seq
621 {
622   \tl_if_blank:eTF { \seq_item:Nn #1 {1} }
```



```

623     {
624       \seq_item:Nn #1 {2} %year
625       -
626       \seq_item:Nn #1 {3} %month
627       -
628       \seq_item:Nn #1 {4} % day
629       \tl_if_blank:eF
630         { \seq_item:Nn #1 {5} }
631         { T \seq_item:Nn #1 {5} } %hour
632       \tl_if_blank:eF
633         { \seq_item:Nn #1 {6} }
634         { : \seq_item:Nn #1 {6} } %minutes
635       \tl_if_blank:eF
636         { \seq_item:Nn #1 {7} }
637         { : \seq_item:Nn #1 {7} } %seconds
638       \seq_item:Nn #1 {8} %Z,+,-
639       \seq_item:Nn #1 {9}
640       \tl_if_blank:eF
641         { \seq_item:Nn #1 {10} }
642         { : \seq_item:Nn #1 {10} }
643     }
644     {
645       \seq_item:Nn #1 {1}
646     }
647 }

```

(End of definition for __pdfmeta_xmp_print_date:N.)

\l__pdfmeta_xmp_currentdate_tl
\l__pdfmeta_xmp_currentdate_seq

The tl var contains the date of the log-file in PDF format, the seq the result splitted with the regex.

```

648 \tl_new:N \l__pdfmeta_xmp_currentdate_tl
649 \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.

```

650 \cs_new_protected:Npn \__pdfmeta_xmp_date_get:nNN #1 #2 #3
651   %#1 property, #2 tl var with PDF date, #3 seq for splitted date
652   {
653     \tl_set:Ne #2 { \GetDocumentProperties{#1} }
654     \tl_if_blank:VTF #2
655     {
656       \seq_set_eq:NN #3 \l__pdfmeta_xmp_currentdate_seq
657       \tl_set_eq:NN #2 \l__pdfmeta_xmp_currentdate_tl
658     }
659     {
660       \__pdfmeta_xmp_date_split:VN #2 #3
661     }
662   }

```

(End of definition for __pdfmeta_xmp_date_get:nNN.)

4.3.4 UUID

We need a command to generate an uuid

`_pdfmeta_xmp_create_uuid:nN`

```
663 \cs_new_protected:Npn \_pdfmeta_xmp_create_uuid:nN #1 #2
664 {
665   \str_set:Ne#2 {\str_lowercase:f{\tex_mdffivesum:D{#1}}}
666   \str_set:Ne#2
667     { uuid:
668       \str_range:Nnn #2{1}{8}
669       -\str_range:Nnn#2{9}{12}
670       -4\str_range:Nnn#2{13}{15}
671       -8\str_range:Nnn#2{16}{18}
672       -\str_range:Nnn#2{19}{30}
673     }
674 }
```

(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.

```
675 \cs_new_protected:Npn \_pdfmeta_xmp_sanitize:nN #1 #2
676 %#1 input string, #2 str with the output
677 {
678   \group_begin:
679   \text_declare_purify_equivalent:Nn \& {\tl_to_str:N & }
680   \text_declare_purify_equivalent:Nn \texttilde {\c_tilde_str}
681   \tl_set:Ne \l__pdfmeta_tmpa_tl { \text_purify:n {#1} }
682   \str_gset:Ne \g__pdfmeta_tmpa_str { \tl_to_str:N \l__pdfmeta_tmpa_tl }
683   \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {\&}{\&#;}
684   \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {<}{\<#;}
685   \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {>}{\>#;}
686   \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {"}{\"#;}
687   \group_end:
688   \str_set_eq:NN #2 \g__pdfmeta_tmpa_str
689 }
690
691 \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`

```
692 \tl_new:N \l__pdfmeta_xmp_doclang_tl
693 \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`

```
694 \regex_new:N\l__pdfmeta_xmp_lang_regex
695 \regex_set:Nn\l__pdfmeta_xmp_lang_regex {\A\[[A-Za-z\-\]]+\)(.*)}
(End of definition for \l__pdfmeta_xmp_lang_regex.)
696 \cs_new_protected:Npn \__pdfmeta_xmp_lang_get:nNN #1 #2 #3
697 % #1 text, #2 t1 var for lang match (or default), #3 t1 var for text
698 {
699   \regex_extract_once:NnN \l__pdfmeta_xmp_lang_regex {#1}\l__pdfmeta_tmpa_seq
700   \seq_if_empty:NTF \l__pdfmeta_tmpa_seq
701     {
702       \tl_set:Nn #2 \l__pdfmeta_xmp_metalang_tl
703       \tl_set:Nn #3 {#1}
704     }
705     {
706       \tl_set:Ne #2 {\seq_item:Nn\l__pdfmeta_tmpa_seq{2}}
707       \tl_set:Ne #3 {\seq_item:Nn\l__pdfmeta_tmpa_seq{3}}
708     }
709   }
710 \cs_generate_variant:Nn \__pdfmeta_xmp_lang_get:nNN {eNN,VNN}
```

4.5 Filling the packet

This t1 var that holds the whole packet

`\g__pdfmeta_xmp_packet_t1`

```
711 \tl_new:N \g__pdfmeta_xmp_packet_t1
(End of definition for \g__pdfmeta_xmp_packet_t1.)
```

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.

```
712 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_chunk:n #1
713 {
714   \tl_gput_right:Ne\g__pdfmeta_xmp_packet_t1
715   {
716     \__pdfmeta_xmp_indent: \exp_not:n{#1}
717   }
718 }
719 \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.

```
720 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_chunk:nN #1 #2
721 {
722   \tl_put_right:Ne#2
723   {
724     \__pdfmeta_xmp_indent: \exp_not:n{#1}
725   }
726 }
727 \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.

```
728 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_open:nn #1 #2 % #1 prefix #2 name
729   {
730     \_pdfmeta_xmp_add_packet_chunk:n {<#1:#2>}
731     \_pdfmeta_xmp_incr_indent:
732   }
733 \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.

```
734 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_open_attr:nnn #1 #2 #3
735   % #1 prefix #2 name #3 attr
736   {
737     \_pdfmeta_xmp_add_packet_chunk:n {<#1:#2~#3>}
738     \_pdfmeta_xmp_incr_indent:
739   }
740 \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.

```
741 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_close:nn #1 #2 % #1 prefix #2:name
742   {
743     \_pdfmeta_xmp_decr_indent:
744     \_pdfmeta_xmp_add_packet_chunk:n {</#1:#2>}
745   }
```

(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.

```
746 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_line:nnn #1 #2 #3
747   % #1 prefix #2 name #3 content
748   {
749     \tl_if_blank:nF {#3}
750     {
751       \_pdfmeta_xmp_sanitizate:nN {#3}\l__pdfmeta_tmpa_str
752       \_pdfmeta_xmp_add_packet_chunk:e {<#1:#2>\l__pdfmeta_tmpa_str</#1:#2>}
753     }
754   }
755 \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.

```
756 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_line:nnnN #1 #2 #3 #4
757   % #1 prefix #2 name #3 content #4 tl_var to prebuilt.
758   {
759     \tl_if_blank:nF {#3}
760     {
```

```

761     \_pdfmeta_xmp_sanitize:nN {#3}\l__pdfmeta_tmpa_str
762     \_pdfmeta_xmp_add_packet_chunk:eN {<#1:#2>\l__pdfmeta_tmpa_str</#1:#2>} #4
763   }
764 }
765 \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

```

766 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_line_attr:nnnn #1 #2 #3 #4
767   % #1 prefix #2 name #3 attribute #4 content
768   {
769     \tl_if_blank:nF {#4}
770     {
771       \_pdfmeta_xmp_sanitize:nN {#4}\l__pdfmeta_tmpa_str
772       \_pdfmeta_xmp_add_packet_chunk:e {<#1:#2-#3>\l__pdfmeta_tmpa_str</#1:#2>}
773     }
774   }
775 \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

```

776 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_line_default:nnnn #1 #2 #3 #4
777   % #1 prefix #2 name #3 default #4 content
778   {
779     \tl_if_blank:nTF { #4 }
780     {
781       \tl_set:Nn \l__pdfmeta_tmpa_tl {#3}
782     }
783     {
784       \tl_set:Nn \l__pdfmeta_tmpa_tl {#4}
785     }
786     \_pdfmeta_xmp_add_packet_line:nnV {#1}{#2}\l__pdfmeta_tmpa_tl
787   }
788 \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:

```

789 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_list_simple:nnnn #1 #2 #3 #4
790   % #1 prefix, #2 name, #3 type (Seq/Bag/Alt) #4 a clist
791   {
792     \clist_if_empty:nF { #4 }
793     {
794       \_pdfmeta_xmp_add_packet_open:nn {#1}{#2}
795       \_pdfmeta_xmp_add_packet_open:nn {rdf}{#3}
796       \clist_map_inline:nn {#4}
797       {
798         \_pdfmeta_xmp_add_packet_line:nnn
799         {rdf}{li}{##1}
800       }
801       \_pdfmeta_xmp_add_packet_close:nn{rdf}{#3}
802       \_pdfmeta_xmp_add_packet_close:nn {#1}{#2}

```

```

803     }
804   }
805 \cs_generate_variant:Nn \_pdfmeta_xmp_add_packet_list_simple:nnnn {nnnV,nnne}

```

Here we check also for the language.

```

806 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_list:nnnn #1 #2 #3 #4
807   % #1 prefix, #2 name, #3 type (Seq/Bag/Alt) #4 a clist
808   {
809     \clist_if_empty:nF { #4 }
810     {
811       \_pdfmeta_xmp_add_packet_open:nn {#1}{#2}
812       \_pdfmeta_xmp_add_packet_open:nn {rdf}{#3}
813       \clist_map_inline:nn {#4}
814       {
815         \_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!

```

816         \tl_if_eq:eeTF{\l__pdfmeta_tmpa_tl}{\l__pdfmeta_xmp_metalang_tl}
817         {
818           \_pdfmeta_xmp_add_packet_line_attr:nneV
819           {rdf}{li}{xml:lang="x-default" }\l__pdfmeta_tmpb_tl
820         }
821         {
822           \_pdfmeta_xmp_add_packet_line_attr:nneV
823           {rdf}{li}{xml:lang="\l__pdfmeta_tmpa_tl" }\l__pdfmeta_tmpb_tl
824         }
825       }
826     \_pdfmeta_xmp_add_packet_close:nn{rdf}{#3}
827     \_pdfmeta_xmp_add_packet_close:nn {#1}{#2}
828   }
829 }
830 \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.

```

831 \cs_new_protected:Npn \_pdfmeta_xmp_build_packet:
832   {

```

Get the main languages

```

833   \tl_set:Ne \l__pdfmeta_xmp_doclang_tl {\GetDocumentProperties{document/lang}}
834   \tl_set:Ne \l__pdfmeta_xmp_metalang_tl {\GetDocumentProperties{hyperref/pdfmetalang}}
835   \tl_if_blank:VT \l__pdfmeta_xmp_metalang_tl
836   { \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

```

837   \_pdfmeta_xmp_build_iptc_data:N \l__pdfmeta_xmp_iptc_data_tl
838   \tl_if_empty:NT \l__pdfmeta_xmp_iptc_data_tl
839   {
840     \seq_remove_all:Nn \l__pdfmeta_xmp_schema_seq { Iptc4xmpCore }
841   }

```

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

```

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

```

The rdf namespaces

```

847 \_pdfmeta_xmp_add_packet_open_attr:nne
848 {rdf}{Description}{rdf:about="" \g__pdfmeta_xmp_xmlns_tl}

```

The extensions

```

849 \_pdfmeta_xmp_add_packet_open:nn{pdfaExtension}{schemas}
850 \_pdfmeta_xmp_add_packet_open:nn {rdf}{Bag}
851 \seq_map_inline:Nn \l__pdfmeta_xmp_schema_seq
852 {
853 \tl_use:c { g__pdfmeta_xmp_schema_##1_tl }
854 }
855 \_pdfmeta_xmp_add_packet_close:nn {rdf}{Bag}
856 \_pdfmeta_xmp_add_packet_close:nn {pdfaExtension}{schemas}

```

Now starts the part with the data.

```

857 % data
858 \_pdfmeta_xmp_build_pdf:
859 \_pdfmeta_xmp_build_xmpRights:
860 \_pdfmeta_xmp_build_standards: %pdfaid,pdfxid,pdfuaid
861 \_pdfmeta_xmp_build_pdfd:
862 \_pdfmeta_xmp_build_dc:
863 \_pdfmeta_xmp_build_photoshop:
864 \_pdfmeta_xmp_build_xmp:
865 \_pdfmeta_xmp_build_xmpMM:
866 \_pdfmeta_xmp_build_prism:
867 \_pdfmeta_xmp_build_iptc:
868 \_pdfmeta_xmp_build_user: %user additions
869 % end
870 \_pdfmeta_xmp_add_packet_close:nn {rdf}{Description}
871 \_pdfmeta_xmp_add_packet_close:nn {rdf}{RDF}
872 \_pdfmeta_xmp_add_packet_close:nn {x}{xmpmeta}
873 \int_set:Nn \l__pdfmeta_xmp_indent_int{20}
874 \prg_replicate:nn{10}{\_pdfmeta_xmp_add_packet_chunk:n {}}
875 \int_zero:N \l__pdfmeta_xmp_indent_int
876 \_pdfmeta_xmp_add_packet_chunk:n {<?xpacket~end="w"?>}
877 }

```

(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.

```

878 \str_new:N \g__pdfmeta_xmp_xmlns_tl
879 \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
\_pdfmeta_xmp_xmlns_new:ne 880 \cs_new_protected:Npn \_pdfmeta_xmp_xmlns_new:nn #1 #2
881 {
882   \prop_gput:Nnn \g_pdfmeta_xmp_xmlns_prop {#1}{#2}
883   \tl_gput_right:Ne \g_pdfmeta_xmp_xmlns_tl
884   {
885     \_pdfmeta_xmp_indent:n{4} xmlns:\exp_not:n{#1="#2"}
886   }
887 }
888 \cs_generate_variant:Nn \_pdfmeta_xmp_xmlns_new:nn {ne}
```

(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

```
889 \_pdfmeta_xmp_xmlns_new:nn {pdf}      {http://ns.adobe.com/pdf/1.3/}
890 \_pdfmeta_xmp_xmlns_new:nn {xmpRights}{http://ns.adobe.com/xap/1.0/rights/}
891 \_pdfmeta_xmp_xmlns_new:nn {dc}      {http://purl.org/dc/elements/1.1/}
892 \_pdfmeta_xmp_xmlns_new:nn {photoshop}{http://ns.adobe.com/photoshop/1.0/}
893 \_pdfmeta_xmp_xmlns_new:nn {xmp}     {http://ns.adobe.com/xap/1.0/}
894 \_pdfmeta_xmp_xmlns_new:nn {xmpMM}   {http://ns.adobe.com/xap/1.0/mm/}
895 \_pdfmeta_xmp_xmlns_new:ne {stEvt}
896   {http://ns.adobe.com/xap/1.0/sType/ResourceEvent\c_hash_str}
897 \_pdfmeta_xmp_xmlns_new:nn {pdfaid}   {http://www.aiim.org/pdfa/ns/id/}
898 \_pdfmeta_xmp_xmlns_new:nn {pdfuaid}  {http://www.aiim.org/pdfua/ns/id/}
899 \_pdfmeta_xmp_xmlns_new:nn {pdfx}     {http://ns.adobe.com/pdfx/1.3/}
900 \_pdfmeta_xmp_xmlns_new:nn {pdfxid}   {http://www.npes.org/pdfx/ns/id/}
901 \_pdfmeta_xmp_xmlns_new:nn {prism}    {http://prismstandard.org/namespaces/basic/3.0/}
902 %\_pdfmeta_xmp_xmlns_new:nn {jav}     {http://www.niso.org/schemas/jav/1.0/}
903 %\_pdfmeta_xmp_xmlns_new:nn {xmpTPg}  {http://ns.adobe.com/xap/1.0/t/pg/}
904 \_pdfmeta_xmp_xmlns_new:ne {stFnt}    {http://ns.adobe.com/xap/1.0/sType/Font\c_hash_str}
905 \_pdfmeta_xmp_xmlns_new:nn {Iptc4xmpCore}{http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
906 \_pdfmeta_xmp_xmlns_new:nn {pdfaExtension}{http://www.aiim.org/pdfa/ns/extension/}
907 \_pdfmeta_xmp_xmlns_new:ne {pdfaSchema}{http://www.aiim.org/pdfa/ns/schema\c_hash_str}
908 \_pdfmeta_xmp_xmlns_new:ne {pdfaProperty}{http://www.aiim.org/pdfa/ns/property\c_hash_str}
909 \_pdfmeta_xmp_xmlns_new:ne {pdfaType} {http://www.aiim.org/pdfa/ns/type\c_hash_str}
910 \_pdfmeta_xmp_xmlns_new:ne {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
911 \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.

```
912 \cs_new_protected:Npn \_pdfmeta_xmp_schema_new:nnn #1 #2 #3
913   % #1 name #2 prefix, #3 text
914   {
915     \seq_put_right:Nn \l__pdfmeta_xmp_schema_seq { #2 }
916     \tl_new:c { g__pdfmeta_xmp_schema_#2_tl }
917     \tl_new:c { g__pdfmeta_xmp_schema_#2_properties_tl }
918     \tl_gput_right:cn { g__pdfmeta_xmp_schema_#2_tl }
919     {
920       \_pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
921       \_pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{schema}{#1}
922       \_pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{prefix}{#2}
923       \_pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{namespaceURI}{#3}
924       \_pdfmeta_xmp_add_packet_open:nn {pdfaSchema}{property}
925       \_pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
926       \tl_use:c { g__pdfmeta_xmp_schema_#2_properties_tl }
927       \_pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
928       \_pdfmeta_xmp_add_packet_close:nn {pdfaSchema}{property}
929       \cs_if_exist_use:c {__pdfmeta_xmp_schema_#2_additions:}
930       \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
931     }
932   }
```

(End of definition for _pdfmeta_xmp_schema_new:nnn.)

_pdfmeta_xmp_property_new:nnn

This adds a property to a schema.

```
933 \cs_new_protected:Npn \_pdfmeta_xmp_property_new:nnnnn #1 #2 #3 #4 #5 %
934   % #1 schema #2 name, #3 type, #4 category #5 description
935   {
936     \tl_gput_right:cn { g__pdfmeta_xmp_schema_#1_properties_tl }
937     {
938       \_pdfmeta_xmp_add_packet_open:nn {rdf}{li~rdf:parseType="Resource"}
939       \_pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{name}{#2}
940       \_pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{valueType}{#3}
941       \_pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{category}{#4}
942       \_pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{description}{#5}
943       \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
944     }
945   }
```

(End of definition for _pdfmeta_xmp_property_new:nnn.)

_pdfmeta_xmp_add_packet_field:nnn

This adds a field to a schema.

```
946 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_field:nnn #1 #2 #3 %
947   % #1 name #2 valuetype #3 description
948   {
949     \_pdfmeta_xmp_add_packet_open_attr:nnn {rdf}{li}{rdf:parseType="Resource"}
950     \_pdfmeta_xmp_add_packet_line:nnn {pdfaField}{name}{#1}
951     \_pdfmeta_xmp_add_packet_line:nnn {pdfaField}{valueType}{#2}
952     \_pdfmeta_xmp_add_packet_line:nnn {pdfaField}{description}{#3}
953     \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
954   }
```

(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.

```
955     \_pdfmeta_xmp_schema_new:nnn
956         {XMP~Media~Management~Schema}
957         {xmpMM}
958         {http://ns.adobe.com/xap/1.0/mm/}
959     \_pdfmeta_xmp_property_new:nnnnn
960         {xmpMM}
961         {OriginalDocumentID}
962         {URI}
963         {internal}
964         {The~common~identifier~for~all~versions~and~renditions~of~a~document.}
```

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

pdfaid-(schema)

```
965     \_pdfmeta_xmp_schema_new:nnn
966         {PDF/A~Identification~Schema}
967         {pdfaid}
968         {http://www.aiim.org/pdfa/ns/id/}
969     \_pdfmeta_xmp_property_new:nnnnn
970         {pdfaid}
971         {year}
972         {Integer}
973         {internal}
974         {Year~of~standard}
```

(End of definition for pdfaid-(schema). This function is documented on page ??.)

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

pdfuaid-(schema)

```
975     \_pdfmeta_xmp_schema_new:nnn
976         {PDF/UA~Universal~Accessibility~Schema}
```

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.

```

977     {pdfuaid}
978     {http://www.aiim.org/pdfua/ns/id/}
979   \_pdfmeta_xmp_property_new:nnnnn
980     {pdfuaid}
981     {part}
982     {Integer}
983     {internal}
984     {Part-of-ISO-14289-standard}
985   \_pdfmeta_xmp_property_new:nnnnn
986     {pdfuaid}
987     {rev}
988     {Integer}
989     {internal}
990     {Revision-of-ISO-14289-standard}

```

(End of definition for pdfuaid-(schema). This function is documented on page ??.)

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)

```

991   \_pdfmeta_xmp_schema_new:nnn
992     {PDF/X-ID-Schema}
993     {pdfxid}
994     {http://www.npes.org/pdfx/ns/id/}
995   \_pdfmeta_xmp_property_new:nnnnn
996     {pdfxid}
997     {GTS_PDFXVersion}
998     {Text}
999     {internal}
1000    {ID-of-PDF/X-standard}

```

(End of definition for pdfxid-(schema). This function is documented on page ??.)

prism-(schema)

```

1001   \_pdfmeta_xmp_schema_new:nnn
1002     {PRISM-Basic-Metadata}
1003     {prism}
1004     {http://prismstandard.org/namespaces/basic/3.0/}
1005   \_pdfmeta_xmp_property_new:nnnnn
1006     {prism}
1007     {complianceProfile}
1008     {Text}
1009     {internal}
1010     {PRISM-specification-compliance-profile-to-which-this-document-adheres}
1011   \_pdfmeta_xmp_property_new:nnnnn
1012     {prism}
1013     {publicationName}

```

```

1014     {Text}
1015     {external}
1016     {Publication~name}
1017 \_pdfmeta_xmp_property_new:nnnnn
1018     {prism}
1019     {aggregationType}
1020     {Text}
1021     {external}
1022     {Publication~type}
1023 \_pdfmeta_xmp_property_new:nnnnn
1024     {prism}
1025     {bookEdition}
1026     {Text}
1027     {external}
1028     {Edition~of~the~book~in~which~the~document~was~published}
1029 \_pdfmeta_xmp_property_new:nnnnn
1030     {prism}
1031     {volume}
1032     {Text}
1033     {external}
1034     {Publication~volume~number}
1035 \_pdfmeta_xmp_property_new:nnnnn
1036     {prism}
1037     {number}
1038     {Text}
1039     {external}
1040     {Publication~issue~number~within~a~volume}
1041 \_pdfmeta_xmp_property_new:nnnnn
1042     {prism}
1043     {pageRange}
1044     {Text}
1045     {external}
1046     {Page~range~for~the~document~within~the~print~version~of~its~publication}
1047 \_pdfmeta_xmp_property_new:nnnnn
1048     {prism}
1049     {issn}
1050     {Text}
1051     {external}
1052     {ISSN~for~the~printed~publication~in~which~the~document~was~published}
1053 \_pdfmeta_xmp_property_new:nnnnn
1054     {prism}
1055     {eIssn}
1056     {Text}
1057     {external}
1058     {ISSN~for~the~electronic~publication~in~which~the~document~was~published}
1059 \_pdfmeta_xmp_property_new:nnnnn
1060     {prism}
1061     {isbn}
1062     {Text}
1063     {external}
1064     {ISBN~for~the~publication~in~which~the~document~was~published}
1065 \_pdfmeta_xmp_property_new:nnnnn
1066     {prism}
1067     {doi}

```

```

1068     {Text}
1069     {external}
1070     {Digital-Object-Identifier-for-the-document}
1071 \_pdfmeta_xmp_property_new:nnnnn
1072     {prism}
1073     {url}
1074     {URL}
1075     {external}
1076     {URL-at-which-the-document-can-be-found}
1077 \_pdfmeta_xmp_property_new:nnnnn
1078     {prism}
1079     {byteCount}
1080     {Integer}
1081     {internal}
1082     {Approximate-file-size-in-octets}
1083 \_pdfmeta_xmp_property_new:nnnnn
1084     {prism}
1085     {pageCount}
1086     {Integer}
1087     {internal}
1088     {Number-of-pages-in-the-print-version-of-the-document}
1089 \_pdfmeta_xmp_property_new:nnnnn
1090     {prism}
1091     {subtitle}
1092     {Text}
1093     {external}
1094     {Document's-subtitle}

```

(End of definition for prism-(schema). This function is documented on page ??.)

iptc

```

1095 \_pdfmeta_xmp_schema_new:nnn
1096     {IPTC-Core-Schema}
1097     {Iptc4xmpCore}
1098     {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
1099 \_pdfmeta_xmp_property_new:nnnnn
1100     {Iptc4xmpCore}
1101     {CreatorContactInfo}
1102     {ContactInfo}
1103     {external}
1104     {Document-creator's-contact-information}
1105 \cs_new_protected:cpn { __pdfmeta_xmp_schema_Iptc4xmpCore_additions: }
1106     {
1107     \_pdfmeta_xmp_add_packet_open:nn{pdfaSchema}{valueType}
1108     \_pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1109     \_pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1110     \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{ContactInfo}
1111     \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
1112     {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
1113     \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{Iptc4xmpCore}
1114     \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
1115     {Basic-set-of-information-to-get-in-contact-with-a-person}
1116     \_pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
1117     \_pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}

```

```

1118         \_pdfmeta_xmp_add_packet_field:nnn{CiAdrCity}{Text}
1119         {Contact~information~city}
1120         \_pdfmeta_xmp_add_packet_field:nnn{CiAdrCtry}{Text}
1121         {Contact~information~country}
1122         \_pdfmeta_xmp_add_packet_field:nnn{CiAdrExtadr}{Text}
1123         {Contact~information~address}
1124         \_pdfmeta_xmp_add_packet_field:nnn{CiAdrPcode}{Text}
1125         {Contact~information~local~postal~code}
1126         \_pdfmeta_xmp_add_packet_field:nnn{CiAdrRegion}{Text}
1127         {Contact~information~regional~information~such~as~state~or~province}
1128         \_pdfmeta_xmp_add_packet_field:nnn{CiEmailWork}{Text}
1129         {Contact~information~email~address(es)}
1130         \_pdfmeta_xmp_add_packet_field:nnn{CiTelWork}{Text}
1131         {Contact~information~telephone~number(s)}
1132         \_pdfmeta_xmp_add_packet_field:nnn{CiUrlWork}{Text}
1133         {Contact~information~Web~URL(s)}
1134         \_pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1135         \_pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
1136         \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1137         \_pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1138         \_pdfmeta_xmp_add_packet_close:nn{pdfaSchema}{valueType}
1139     }

```

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 compliancy) 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.)

```

1140     \cs_new_protected:Npn \_pdfmeta_xmp_schema_enable_pdfd:
1141     {
1142         \_pdfmeta_xmp_xmlns_new:ne {pdfd}{http://pdfa.org/declarations/}
1143         \_pdfmeta_xmp_schema_new:nnn
1144             {PDF~Declarations~Schema}
1145             {pdfd}
1146             {http://pdfa.org/declarations/}
1147         \_pdfmeta_xmp_property_new:nnnnn
1148             {pdfd}
1149             {declarations}
1150             {Bag~declaration}
1151             {external}
1152             {An~unordered~array~of~PDF~Declaration~entries,~where~each~PDF~Declaration~represen

```

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

```

1153     \cs_new_protected:cpn { _pdfmeta_xmp_schema_pdfd_additions: }
1154     {

```

```

1155     \_pdfmeta_xmp_add_packet_open:nn{pdfaSchema}{valueType}
1156     \_pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1157     \_pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1158     \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{claim}
1159     \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
1160     {http://pdfa.org/declarations/}
1161     \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{pdfd}
1162     \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
1163     {A~structure~describing~properties~of~an~individual~claim.}
1164     \_pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
1165     \_pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1166     \_pdfmeta_xmp_add_packet_field:nnn{claimReport}{Text}
1167     {A~URL~to~a~report~containing~details~of~the~specific~conformance~claim}
1168     \_pdfmeta_xmp_add_packet_field:nnn{claimCredentials}{Text}
1169     {The~claimant's~credentials.}
1170     \_pdfmeta_xmp_add_packet_field:nnn{claimDate}{Text}
1171     {A~date~identifying~when~the~claim~was~made.}
1172     \_pdfmeta_xmp_add_packet_field:nnn{claimBy}{Text}
1173     {The~name~of~the~organization~and/or~individual~and/or~software~making}
1174     \_pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1175     \_pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
1176     \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1177     \_pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1178     \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{declaration}
1179     \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
1180     {http://pdfa.org/declarations/}
1181     \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{pdfd}
1182     \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
1183     {A~structure~describing~a~single~PDF~Declaration~asserting~conformance~v}
1184     \_pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
1185     \_pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1186     \_pdfmeta_xmp_add_packet_field:nnn{conformsTo}{Text}
1187     {A~property~containing~a~URI~specifying~the~standard~or~profile~by~the}
1188     \_pdfmeta_xmp_add_packet_field:nnn{claimData}{Bag~claim}
1189     {An~unordered~array~of~claim~data,~where~each~claim~identifies~the~natu}
1190     \_pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1191     \_pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
1192     \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1193     \_pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1194     \_pdfmeta_xmp_add_packet_close:nn{pdfaSchema}{valueType}
1195 }

```

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

```

1196     \cs_gset_eq:NN \_pdfmeta_xmp_schema_enable_pdfd: \prg_do_nothing:
1197 }

```

4.8 The actual user / document data

4.8.1 pdf

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

```

\_pdfmeta_xmp_build_pdf:
  Producer/pdfproducer
  PDFversion

```

```

1198 \cs_new_protected:Npn \__pdfmeta_xmp_build_pdf:
1199 {

```

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

```

1200 \__pdfmeta_xmp_add_packet_line_default:nnee
1201 {pdf}{Producer}
1202 {\c_sys_engine_exec_str-\c_sys_engine_version_str}
1203 {\GetDocumentProperties{hyperref/pdfproducer}}

```

Now the PDF version

```

1204 \__pdfmeta_xmp_add_packet_line:nne{pdf}{PDFVersion}{\pdf_version:}
1205 }

```

(End of definition for __pdfmeta_xmp_build_pdf:, Producer/pdfproducer, and PDFversion. These functions are documented on page ??.)

4.8.2 xmp

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

```

\__pdfmeta_xmp_build_xmp:
  CreatorTool/pdfcreator 1206 \cs_new_protected:Npn \__pdfmeta_xmp_build_xmp:
    BaseUrl/baseurl      1207 {

```

The creator

```

1208 \__pdfmeta_xmp_add_packet_line_default:nnee
1209 {xmp}{CreatorTool}
1210 {LaTeX}
1211 { \GetDocumentProperties{hyperref/pdfcreator} }

```

The baseurl

```

1212 \__pdfmeta_xmp_add_packet_line_default:nnee
1213 {xmp}{BaseUrl}{}
1214 { \GetDocumentProperties{hyperref/baseurl} }

```

CreationDate

```

1215 \__pdfmeta_xmp_date_get:nNN
1216 {document/creationdate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1217 \__pdfmeta_xmp_add_packet_line:nne{xmp}{CreateDate}{\__pdfmeta_xmp_print_date:N\l__pdfme
1218 \pdfmanagement_add:nne{Info}{CreateDate}{(\l__pdfmeta_tmpa_tl)}

```

ModifyDate

```

1219 \__pdfmeta_xmp_date_get:nNN
1220 {document/moddate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1221 \__pdfmeta_xmp_add_packet_line:nne{xmp}{ModifyDate}{\__pdfmeta_xmp_print_date:N\l__pdfme
1222 \pdfmanagement_add:nne{Info}{ModDate}{(\l__pdfmeta_tmpa_tl)}

```

MetadataDate

```

1223 \__pdfmeta_xmp_date_get:nNN
1224 {hyperref/pdfmetadate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1225 \__pdfmeta_xmp_add_packet_line:nne{xmp}{MetadataDate}{\__pdfmeta_xmp_print_date:N\l__pdfme
1226 }

```

(End of definition for __pdfmeta_xmp_build_xmp:, CreatorTool/pdfcreator, and BaseUrl/baseurl. These functions are documented on page ??.)

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:`

```
1227 \cs_new_protected:Npn \_pdfmeta_xmp_build_standards:
1228 {
1229   \_pdfmeta_xmp_add_packet_line:nne {pdfaid}{part}{\pdfmeta_standard_item:n{level}}
1230   \_pdfmeta_xmp_add_packet_line:nne
1231     {pdfaid}{conformance}{\pdfmeta_standard_item:n{conformance}}
1232   \int_compare:nNnTF {0\pdfmeta_standard_item:n{level}}<{4}
1233     {\_pdfmeta_xmp_add_packet_line:nne {pdfaid}{year} {\pdfmeta_standard_item:n{year}}}
1234     {\_pdfmeta_xmp_add_packet_line:nne {pdfaid}{rev} {\pdfmeta_standard_item:n{year}}}
1235   \_pdfmeta_xmp_add_packet_line:nne
1236     {pdfxid}{GTS_PDFXVersion}{\GetDocumentProperties{document/pdfstandard-X}}
1237   \pdfmanagement_get_documentproperties:nNT {document/pdfstandard-UA}\l__pdfmeta_tmpa_tl
1238   {
1239     \_pdfmeta_xmp_add_packet_line:nne
1240       {pdfuaid}{part}{\exp_last_unbraced:No\use_i:nn \l__pdfmeta_tmpa_tl}
1241     \_pdfmeta_xmp_add_packet_line:nne
1242       {pdfuaid}{rev}{\exp_last_unbraced:No\use_ii:nn \l__pdfmeta_tmpa_tl}
1243   }
1244 }
```

(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.

```
1245 \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:`

```
1246 \cs_new_protected:Npn \_pdfmeta_xmp_build_pdfd:
1247 {
1248   \prop_if_empty:NF\g_pdfmeta_xmp_pdfd_data_prop
1249   {
1250     \_pdfmeta_xmp_add_packet_open:nn{pdfd}{declarations}
1251     \_pdfmeta_xmp_add_packet_open:nn{rdf}{Bag}
1252     \prop_map_inline:Nn \g_pdfmeta_xmp_pdfd_data_prop
1253     {
1254       \_pdfmeta_xmp_build_pdfd_claim:nn{##1}{##2}
1255     }
1256     \_pdfmeta_xmp_add_packet_close:nn{rdf}{Bag}
1257     \_pdfmeta_xmp_add_packet_close:nn{pdfd}{declarations}
1258   }
1259 }
```

(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.

```
1260 \cs_new_protected:Npn \_pdfmeta_xmp_build_pdfd_claim:nn #1#2
1261 {
1262   \_pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1263   \_pdfmeta_xmp_add_packet_line:nnn{pdfd}{conformsTo}{#1}
1264   \tl_if_empty:nF {#2}
1265   {
1266     \_pdfmeta_xmp_add_packet_open:nn{pdfd}{claimData}
1267     \_pdfmeta_xmp_add_packet_open:nn{rdf}{Bag}
1268     #2
1269     \_pdfmeta_xmp_add_packet_close:nn{rdf}{Bag}
1270     \_pdfmeta_xmp_add_packet_close:nn{pdfd}{claimData}
1271   }
1272   \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1273 }
```

(End of definition for _pdfmeta_xmp_build_pdfd_claim:nn.)

4.10 Photoshop

_pdfmeta_xmp_build_photoshop:

```
1274 \cs_new_protected:Npn \_pdfmeta_xmp_build_photoshop:
1275 {
pdfauthortitle/photoshop:AuthorsPosition
1276   \_pdfmeta_xmp_add_packet_line:nne{photoshop}{AuthorsPosition}
1277   { \GetDocumentProperties{hyperref/pdfauthortitle} }
pdfcaptionwriter/photoshop:CaptionWriter
1278   \_pdfmeta_xmp_add_packet_line:nne{photoshop}{CaptionWriter}
1279   { \GetDocumentProperties{hyperref/pdfcaptionwriter} }
1280 }
```

(End of definition for _pdfmeta_xmp_build_photoshop:.)

4.11 XMP Media Management

_pdfmeta_xmp_build_xmpMM:

```
1281 \cs_new_protected:Npn \_pdfmeta_xmp_build_xmpMM:
1282 {
pdfdocumentid / xmpMM:DocumentID
1283   \str_set:Ne\l__pdfmeta_tmpa_str {\GetDocumentProperties{hyperref/pdfdocumentid}}
1284   \str_if_empty:NT \l__pdfmeta_tmpa_str
1285   {
1286     \_pdfmeta_xmp_create_uuid:nN
1287     {\jobname\GetDocumentProperties{hyperref/pdftitle}}
1288     \l__pdfmeta_tmpa_str
1289   }
1290   \_pdfmeta_xmp_add_packet_line:nnV{xmpMM}{DocumentID}
1291   \l__pdfmeta_tmpa_str
```

pdfinstanceid / xmpMM:InstanceID

```
1292     \str_set:Ne\l__pdfmeta_tmpa_str {\GetDocumentProperties{hyperref/pdfinstanceid}}
1293     \str_if_empty:NT \l__pdfmeta_tmpa_str
1294     {
1295         \__pdfmeta_xmp_create_uuid:nN
1296         {\jobname\l__pdfmeta_xmp_currentdate_tl}
1297         \l__pdfmeta_tmpa_str
1298     }
1299     \__pdfmeta_xmp_add_packet_line:nnV{xmpMM}{InstanceID}
1300     \l__pdfmeta_tmpa_str
```

pdfversionid/xmpMM:VersionID

```
1301     \__pdfmeta_xmp_add_packet_line:nne{xmpMM}{VersionID}
1302     { \GetDocumentProperties{hyperref/pdfversionid} }
```

pdfrendition/xmpMM:RenditionClass

```
1303     \__pdfmeta_xmp_add_packet_line:nne{xmpMM}{RenditionClass}
1304     { \GetDocumentProperties{hyperref/pdfrendition} }
1305 }
```

(End of definition for __pdfmeta_xmp_build_xmpMM:.)

4.12 Rest of dublin Core data

__pdfmeta_xmp_build_dc:

```
dc:creator/pdfauthor 1306 \cs_new_protected:Npn \__pdfmeta_xmp_build_dc:
dc:subject/pdfkeywords 1307 {
```

```
dc:type/pdftype pdfauthor/dc:creator
```

```
dc:publisher/pdfpublisher 1308     \__pdfmeta_xmp_add_packet_list:nnne {dc}{creator}{Seq}
```

```
dc:description/pdfsubject 1309     { \GetDocumentProperties{hyperref/pdfauthor} }
```

```
dc:language/lang/pdflang 1310     \int_compare:nNnT {0\pdfmeta_standard_item:n{level}}={1}
```

```
dc:identifier/pdfidentifier 1311     { \pdfmanagement_remove:nn{Info}{Author} }
```

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

```
1312     \__pdfmeta_xmp_add_packet_list:nnne {dc}{title}{Alt}
1313     { \GetDocumentProperties{hyperref/pdftitle} }
```

pdfkeywords/dc:subject

```
1314     \__pdfmeta_xmp_add_packet_list:nnne {dc}{subject}{Bag}
```

```
1315     { \GetDocumentProperties{hyperref/pdfkeywords} }
```

```
1316     \int_compare:nNnT {0\pdfmeta_standard_item:n{level}}={1}
```

```
1317     { \pdfmanagement_remove:nn{Info}{Keywords} }
```

pdftype/dc:type

```
1318     \pdfmanagement_get_documentproperties:nNTF { hyperref/pdftype } \l__pdfmeta_tmpa_tl
```

```
1319     {
```

```
1320         \__pdfmeta_xmp_add_packet_list_simple:nnnV {dc}{type}{Bag}\l__pdfmeta_tmpa_tl
```

```
1321     }
```

```
1322     {
```

```
1323         \__pdfmeta_xmp_add_packet_list_simple:nnnn {dc}{type}{Bag}{Text}
```

```
1324     }
```

```

pdfpublisher/dc:publisher
1325     \_pdfmeta_xmp_add_packet_list:nne {dc}{publisher}{Bag}
1326     { \GetDocumentProperties{hyperref/pdfpublisher} }
pdfsubject/dc:description
1327     \_pdfmeta_xmp_add_packet_list:nne
1328     {dc}{description}{Alt}
1329     {\GetDocumentProperties{hyperref/pdfsubject}}
lang/pdflang/dc:language
1330     \_pdfmeta_xmp_add_packet_list_simple:nnnV
1331     {dc}{language}{Bag}\l__pdfmeta_xmp_doclang_tl
pdfidentifier/dc:identifier
1332     \_pdfmeta_xmp_add_packet_line:nne{dc}{identifier}
1333     { \GetDocumentProperties{hyperref/pdfidentifier} }
pdfdate/dc:date
1334     \_pdfmeta_xmp_date_get:nNN {hyperref/pdfdate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1335     \_pdfmeta_xmp_add_packet_list_simple:nne
1336     {dc}{date}{Seq}{\_pdfmeta_xmp_print_date:N\l__pdfmeta_tmpa_seq}

```

The file format

```

1337     \_pdfmeta_xmp_add_packet_line:nnn{dc}{format}{application/pdf}

```

The source

```

1338     \_pdfmeta_xmp_add_packet_line_default:nnee
1339     {dc}{source}
1340     { \c_sys_jobname_str.tex }
1341     { \GetDocumentProperties{hyperref/pdfsource} }
1342     \_pdfmeta_xmp_add_packet_list:nne{dc}{rights}{Alt}
1343     {\GetDocumentProperties{hyperref/pdfcopyright}}
1344     }

```

(End of definition for _pdfmeta_xmp_build_dc: and others. These functions are documented on page ??.)

4.13 xmpRights

_pdfmeta_xmp_build_xmpRights:

```

1345 \cs_new_protected:Npn \_pdfmeta_xmp_build_xmpRights:
1346 {
1347     \_pdfmeta_xmp_add_packet_line:nne
1348     {xmpRights}
1349     {WebStatement}
1350     {\GetDocumentProperties{hyperref/pdflicenseurl}}
1351     \_pdfmeta_xmp_add_packet_line:nne
1352     {xmpRights}
1353     {Marked}
1354     {
1355     \str_case:en {\GetDocumentProperties{document/copyright}}
1356     {
1357         {true}{True}
1358         {false}{False}
1359     }
1360     }
1361 }

```

(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
```

```
1362 \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
```

```
1363 \cs_new_protected:Npn \_pdfmeta_xmp_build_iptc_data:N #1
```

```
1364 {
```

```
1365   \tl_clear:N #1
```

```
1366   \_pdfmeta_xmp_incr_indent:\_pdfmeta_xmp_incr_indent:\_pdfmeta_xmp_incr_indent:\_pdf
```

```
1367   \_pdfmeta_xmp_add_packet_line:nneN
```

```
1368     {Iptc4xmpCore}{CiAdrExtadr}
```

```
1369     {\GetDocumentProperties{hyperref/pdfcontactaddress}}
```

```
1370     #1
```

```
1371   \_pdfmeta_xmp_add_packet_line:nneN
```

```
1372     {Iptc4xmpCore}{CiAdrCity}
```

```
1373     {\GetDocumentProperties{hyperref/pdfcontactcity}}
```

```
1374     #1
```

```
1375   \_pdfmeta_xmp_add_packet_line:nneN
```

```
1376     {Iptc4xmpCore}{CiAdrPcode}
```

```
1377     {\GetDocumentProperties{hyperref/pdfcontactpostcode}}
```

```
1378     #1
```

```
1379   \_pdfmeta_xmp_add_packet_line:nneN
```

```
1380     {Iptc4xmpCore}{CiAdrCtry}
```

```
1381     {\GetDocumentProperties{hyperref/pdfcontactcountry}}
```

```
1382     #1
```

```
1383   \_pdfmeta_xmp_add_packet_line:nneN
```

```
1384     {Iptc4xmpCore}{CiTelWork}
```

```
1385     {\GetDocumentProperties{hyperref/pdfcontactphone}}
```

```
1386     #1
```

```
1387   \_pdfmeta_xmp_add_packet_line:nneN
```

```
1388     {Iptc4xmpCore}{CiEmailWork}
```

```
1389     {\GetDocumentProperties{hyperref/pdfcontactemail}}
```

```
1390     #1
```

```
1391   \_pdfmeta_xmp_add_packet_line:nneN
```

```
1392     {Iptc4xmpCore}{CiUrlWork}
```

```
1393     {\GetDocumentProperties{hyperref/pdfcontacturl}}
```

```
1394     #1
```

```
1395   \_pdfmeta_xmp_decr_indent:\_pdfmeta_xmp_decr_indent:\_pdfmeta_xmp_decr_indent:\_pdf
```

```
1396 }
```

(End of definition for _pdfmeta_xmp_build_iptc_data:N.)

```
\_pdfmeta_xmp_build_iptc:
```

```
1397 \cs_new_protected:Npn \_pdfmeta_xmp_build_iptc:
```

```
1398 {
```

```
1399   \tl_if_empty:NF\l_pdfmeta_xmp_iptc_data_tl
```

```

1400     {
1401         \__pdfmeta_xmp_add_packet_open_attr:nnn
1402         {Iptc4xmpCore}{CreatorContactInfo}{rdf:parseType="Resource"}
1403         \tl_gput_right:Ne\g__pdfmeta_xmp_packet_tl { \l__pdfmeta_xmp_iptc_data_tl }
1404         \__pdfmeta_xmp_add_packet_close:nn
1405         {Iptc4xmpCore}{CreatorContactInfo}
1406     }
1407 }

```

(End of definition for __pdfmeta_xmp_build_iptc:.)

4.15 Prism

```

\__pdfmeta_xmp_build_prism:
    complianceProfile
prism:subtitle/pdfsubtitle

```

```

1408 \cs_new_protected:Npn \__pdfmeta_xmp_build_prism:
1409 {

```

The compliance profile is a fix value taken from hyperxmp

```

1410     \__pdfmeta_xmp_add_packet_line:nnn
1411     {prism}{complianceProfile}
1412     {three}

```

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

```

1413     \__pdfmeta_xmp_lang_get:eNN
1414     {\GetDocumentProperties{hyperref/pdfsubtitle}}
1415     \l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl
1416     \__pdfmeta_xmp_add_packet_line_attr:nneV
1417     {prism}{subtitle}
1418     {xml:lang="\l__pdfmeta_tmpa_tl"}
1419     \l__pdfmeta_tmpb_tl

```

Then publicationName

```

1420     \__pdfmeta_xmp_lang_get:eNN
1421     {\GetDocumentProperties{hyperref/pdfpublication}}
1422     \l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl
1423     \__pdfmeta_xmp_add_packet_line_attr:nneV
1424     {prism}{publicationName}
1425     {xml:lang="\l__pdfmeta_tmpa_tl"}
1426     \l__pdfmeta_tmpb_tl

```

Now the rest

```

1427     \__pdfmeta_xmp_add_packet_line:nne
1428     {prism}{bookEdition}
1429     {\GetDocumentProperties{hyperref/pdfbookedition}}
1430     \__pdfmeta_xmp_add_packet_line:nne
1431     {prism}{aggregationType}
1432     {\GetDocumentProperties{hyperref/pdfpubtype}}
1433     \__pdfmeta_xmp_add_packet_line:nne
1434     {prism}{volume}
1435     {\GetDocumentProperties{hyperref/pdfvolumenum}}
1436     \__pdfmeta_xmp_add_packet_line:nne
1437     {prism}{number}
1438     {\GetDocumentProperties{hyperref/pdfissuenum}}
1439     \__pdfmeta_xmp_add_packet_line:nne
1440     {prism}{pageRange}

```

```

1441     {\GetDocumentProperties{hyperref/pdfpagerange}}
1442 \__pdfmeta_xmp_add_packet_line:nne
1443   {prism}{issn}
1444   {\GetDocumentProperties{hyperref/pdfissn}}
1445 \__pdfmeta_xmp_add_packet_line:nne
1446   {prism}{eIssn}
1447   {\GetDocumentProperties{hyperref/pdfeissn}}
1448 \__pdfmeta_xmp_add_packet_line:nne
1449   {prism}{doi}
1450   {\GetDocumentProperties{hyperref/pdfdoi}}
1451 \__pdfmeta_xmp_add_packet_line:nne
1452   {prism}{url}
1453   {\GetDocumentProperties{hyperref/pdfurl}}

```

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

```

1454 \tl_set:Nc \l__pdfmeta_tmpa_tl {\GetDocumentProperties{hyperref/pdfnumpages} }
1455 \__pdfmeta_xmp_add_packet_line:nne
1456   {prism}{pageCount}
1457   {\tl_if_blank:VTF \l__pdfmeta_tmpa_tl {\PreviousTotalPages}{\l__pdfmeta_tmpa_tl}}
1458 }

```

(End of definition for __pdfmeta_xmp_build_prism:, complianceProfile, and prism:subtitle/pdfsubtitle. These functions are documented on page ??.)

4.15.1 User additions

\g_pdfmeta_xmp_user_packet_str

```

1459 \tl_new:N \g__pdfmeta_xmp_user_packet_tl

```

(End of definition for \g_pdfmeta_xmp_user_packet_str.)

__pdfmeta_xmp_build_user:

```

1460 \cs_new_protected:Npn \__pdfmeta_xmp_build_user:
1461 {
1462   \int_zero:N \l__pdfmeta_xmp_indent_int
1463   \g__pdfmeta_xmp_user_packet_tl
1464   \int_set:Nn \l__pdfmeta_xmp_indent_int {3}
1465 }

```

(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

```

1466 \AddToHook{shipout/lastpage}
1467 {
1468   \bool_if:NT\g__pdfmeta_xmp_bool
1469   {
1470     \str_if_exist:NTF\c_sys_timestamp_str
1471     {
1472       \tl_set_eq:NN \l__pdfmeta_xmp_currentdate_tl \c_sys_timestamp_str
1473     }
1474   }

```

```

1475         \file_get_timestamp:nN{\jobname.log}\l__pdfmeta_xmp_currentdate_tl
1476     }
1477     \__pdfmeta_xmp_date_split:VN\l__pdfmeta_xmp_currentdate_tl\l__pdfmeta_xmp_currentdate_tl
1478     \__pdfmeta_xmp_build_packet:
1479     \exp_args:No
1480     \__pdf_backend_metadata_stream:n {\g__pdfmeta_xmp_packet_tl}
1481     \pdfmanagement_add:nne {Catalog} {Metadata}{\pdf_object_ref_last:}
1482     \bool_if:NT \g__pdfmeta_xmp_export_bool
1483     {
1484         \iow_open:Nn\g_tmpa_iow{\g__pdfmeta_xmp_export_str.xmpi}
1485         \exp_args:NNo\iow_now:Nn\g_tmpa_iow{\g__pdfmeta_xmp_packet_tl}
1486         \iow_close:N\g_tmpa_iow
1487     }
1488 }
1489 }

```

4.17 User commands

`\pdfmeta_xmp_add:n`

```

1490 \cs_new_protected:Npn \pdfmeta_xmp_add:n #1
1491 {
1492     \tl_gput_right:Nn \g__pdfmeta_xmp_user_packet_tl
1493     {
1494         \__pdfmeta_xmp_add_packet_chunk:n { #1 }
1495     }
1496 }

```

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

`\pdfmeta_xmp_xmlns_new:nn`

```

1497 \cs_new_protected:Npn \pdfmeta_xmp_xmlns_new:nn #1 #2
1498 {
1499     \prop_if_in:NnTF \g__pdfmeta_xmp_xmlns_prop {#1}
1500     {\msg_warning:nnn{pdfmeta}{namespace-defined}{#1}}
1501     {\__pdfmeta_xmp_xmlns_new:nn {#1}{#2}}
1502 }

```

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

`\pdfmeta_xmp_add_declaration:n`

`\pdfmeta_xmp_add_declaration:e`

```

1503 \cs_new_protected:Npn \pdfmeta_xmp_add_declaration:n #1 %conformsTo uri
1504 {
1505     \__pdfmeta_xmp_schema_enable_pdfd:
1506     \prop_gput:Nnn\g__pdfmeta_xmp_pdfd_data_prop{#1}{}
1507 }
1508 \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:enmn`

```

1509 \cs_new_protected:Npn \pdfmeta_xmp_add_declaration:nnnn #1#2#3#4#5
1510 %#1=conformsTo uri, #2 claimBy, #3 claimDate #4 claimCredentials #4 claimReport
1511 {
1512     \__pdfmeta_xmp_schema_enable_pdfd:

```



```

1513 \tl_set:Nn \l__pdfmeta_tmpa_tl
1514 {
1515   \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1516   \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimBy}{#2}
1517   \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimDate}{#3}
1518   \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimCredentials}{#4}
1519   \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimReport}{#5}
1520   \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1521 }
1522 \prop_get:NnNT \g__pdfmeta_xmp_pdfd_data_prop {#1}\l__pdfmeta_tmpb_tl
1523 {
1524   \tl_concat:NNN \l__pdfmeta_tmpa_tl \l__pdfmeta_tmpa_tl \l__pdfmeta_tmpb_tl
1525 }
1526 \prop_gput:Nno\g__pdfmeta_xmp_pdfd_data_prop{#1}
1527 {
1528   \l__pdfmeta_tmpa_tl
1529 }
1530 }
1531 \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:
1532 \cs_new:Npn \__pdfmeta_xmp_iso_today:
1533 {
1534   \int_use:N\c_sys_year_int-
1535   \int_compare:nNnT {\c_sys_month_int} < {10}{0} \int_use:N\c_sys_month_int -
1536   \int_compare:nNnT {\c_sys_day_int} < {10}{0} \int_use:N\c_sys_day_int
1537 }
1538 \cs_new_protected:Npn \__pdfmeta_xmp_wtpdf_reuse_declaration:
1539 {
1540   \pdfmeta_xmp_add_declaration:eeenn
1541   {http://pdfa.org/declarations/wtpdf\c_hash_str reuse1.0}
1542   {LaTeX-Project}
1543   {\__pdfmeta_xmp_iso_today:}{}{}
1544 }
1545 \cs_new_protected:Npn \__pdfmeta_xmp_wtpdf_accessibility_declaration:
1546 {
1547   \pdfmeta_xmp_add_declaration:ennnn
1548   {http://pdfa.org/declarations/wtpdf\c_hash_str accessibility1.0}
1549   {LaTeX-Project}
1550   {\__pdfmeta_xmp_iso_today:}{}{}
1551 }

```

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

```

1552 </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>	679
<code>\'</code>	614
<code>\+</code>	614
<code>\-</code>	614, 695
<code>\[</code>	695
<code>\]</code>	695
A	
<code>\A</code>	695
<code>\AddToDocumentProperties</code> ...	497, 499, 501, 503, 505, 507, 509, 511, 514, 518
<code>\AddToHook</code>	302, 433, 519, 521, 1466
B	
<code>BaseUrl/baseurl</code>	<u>1206</u>
bitset commands:	
<code>\bitset_set_false:Nn</code>	93, 94, 95
<code>\bitset_set_true:Nn</code>	92
<code>\bitset_to_arabic:N</code>	96, 97, 98, 99, 100
bool commands:	
<code>\bool_gset_false:N</code>	529, 564
<code>\bool_gset_true:N</code> ..	488, 528, 559, 568
<code>\bool_if:NTF</code>	308, 1468, 1482
<code>\bool_lazy_or:nnTF</code>	575
<code>\bool_new:N</code>	487, 551
C	
char commands:	
<code>\char_generate:nn</code> ..	580, 585, 586, 587
clist commands:	
<code>\clist_if_empty:nTF</code>	792, 809
<code>\clist_map_inline:nn</code> ..	416, 796, 813
<code>complianceProfile</code>	<u>1408</u>
<code>CreatorTool/pdfcreator</code>	<u>1206</u>
cs commands:	
<code>\cs_generate_variant:Nn</code>	619, 691, 710, 719, 727, 733, 740, 755, 765, 775, 788, 805, 830, 888, 1508, 1531
<code>\cs_gset_eq:NN</code>	1196
<code>\cs_if_exist:NTF</code>	39
<code>\cs_if_exist_use:N</code>	929
<code>\cs_new:Npn</code>	17, 579, 583, 591, 597, 620, 1532
<code>\cs_new_protected:Npn</code>	21, 56, 64, 72, 78, 84, 90, 394, 409, 485, 603, 608, 615, 650, 663, 675, 696, 712, 720, 728, 734, 741, 746, 756, 766, 776, 789, 806, 831, 880, 912, 933, 946, 1105, 1140, 1153, 1198, 1206, 1227, 1246, 1260, 1274, 1281, 1306, 1345, 1363, 1397, 1408, 1460, 1490, 1497, 1503, 1509, 1538, 1545
<code>\cs_set_eq:NN</code>	542, 548, 836
D	
<code>\d</code>	614
dc commands:	
<code>dc:description/pdfsubject</code>	<u>1306</u>
<code>dc:identifier/pdfidentifier</code> ..	<u>1306</u>
<code>dc:language/lang/pdflang</code>	<u>1306</u>
<code>dc:Ncreator/pdfauthor</code>	<u>1306</u>
<code>dc:publisher/pdfpublisher</code>	<u>1306</u>
<code>dc:subject/pdfkeywords</code>	<u>1306</u>
<code>dc:type/pdftype</code>	<u>1306</u>
<code>\DocumentMetadata</code>	2-4
E	
exp commands:	
<code>\exp_args:NNe</code>	453, 458, 464
<code>\exp_args:Nne</code>	316
<code>\exp_args:Nnne</code>	41
<code>\exp_args:NNno</code>	369, 1485
<code>\exp_args:No</code>	1479
<code>\exp_args:NV</code>	471, 474
<code>\exp_last_unbraced:No</code> ...	1240, 1242
<code>\exp_not:n</code>	716, 724, 885
F	
file commands:	
<code>\file_get_timestamp:nN</code>	1475
G	
<code>\GetDocumentProperties</code>	653, 833, 834, 1203, 1211, 1214, 1236, 1277, 1279, 1283, 1287, 1292, 1302, 1304, 1309, 1313, 1315, 1326, 1329, 1333, 1341, 1343, 1350, 1355, 1369, 1373, 1377, 1381, 1385, 1389, 1393, 1414, 1421, 1429, 1432, 1435, 1438, 1441, 1444, 1447, 1450, 1453, 1454
group commands:	
<code>\group_begin:</code>	312, 411, 678
<code>\group_end:</code>	317, 430, 687
H	
hook commands:	
<code>\hook_gput_code:nnn</code>	102, 489

I	
int commands:	
\ <code>\int_compare:nNnTF</code>	1232, 1310, 1316, 1535, 1536
\ <code>\int_decr:N</code>	610
\ <code>\int_incr:N</code>	605
\ <code>\int_new:N</code>	590
\ <code>\int_set:Nn</code>	873, 1464
\ <code>\int_use:N</code>	1534, 1535, 1536
\ <code>\int_zero:N</code>	875, 1462
iow commands:	
\ <code>\iow_close:N</code>	1486
\ <code>\iow_newline:</code>	593, 599
\ <code>\iow_now:Nn</code>	1485
\ <code>\iow_open:Nn</code>	1484
\ <code>\g_tmpa_iow</code>	1484, 1485, 1486
J	
\ <code>\jobname</code>	1287, 1296, 1475
K	
kernel internal commands:	
\ <code>\g_kernel_pdfmanagement_end_-run_code_tl</code>	306
keys commands:	
\ <code>\keys_define:nn</code>	324, 331, 494, 536, 554
\ <code>\l_keys_key_str</code>	371
\ <code>\keys_set:nn</code>	328, 533
M	
msg commands:	
\ <code>\msg_new:nnn</code>	7, 8, 573, 574
\ <code>\msg_warning:nnn</code>	448, 481, 1500
\ <code>\msg_warning:nnnn</code>	112, 122
P	
pdf commands:	
\ <code>\pdf_object_if_exist:nTF</code>	396
\ <code>\pdf_object_new:n</code>	398
\ <code>\pdf_object_ref:n</code>	415
\ <code>\pdf_object_ref_last:</code>	429, 1481
\ <code>\pdf_object_unnamed_write:nn</code> ..	428
\ <code>\pdf_object_write:nnn</code>	399
\ <code>\pdf_string_from_unicode:nnN</code> ..	423
\ <code>\pdf_version:</code>	3, 4, 111, 113, 121, 123, 1204
\ <code>\pdf_version_compare:NnTF</code>	58, 66
pdf internal commands:	
\ <code>__pdf_backend_metadata_stream:n</code>	1480
\ <code>__pdf_backend_Names_gpush:nn</code> ..	316
\ <code>__pdf_backend_omit_charset:n</code> ..	109
\ <code>__pdf_backend_omit_info:n</code>	107
\ <code>__pdf_backend_set_regression_-data:</code>	486
\ <code>pdfaid~(schema)</code>	965
pdfannot commands:	
\ <code>\pdfannot_dict_put:nnn</code>	96, 97, 98, 99, 100
\ <code>\l_pdfannot_F_bitset</code>	92, 93, 94, 95, 96, 97, 98, 99, 100
pdfdict commands:	
\ <code>\pdfdict_if_empty:nTF</code>	310
\ <code>\pdfdict_new:n</code>	375
\ <code>\pdfdict_put:nnn</code>	313, 314, 376, 412, 413, 424
\ <code>\pdfdict_use:n</code>	428
pdffile commands:	
\ <code>\pdffile_embed_stream:nnN</code>	315
pdfmanagement commands:	
\ <code>\pdfmanagement_add:nnn</code>	429, 491, 492, 1218, 1222, 1481
\ <code>\pdfmanagement_get_documentproperties:nNTF</code>	1237, 1318
\ <code>\pdfmanagement_remove:nn</code> ..	1311, 1317
pdfmanagement internal commands:	
\ <code>\g_pdfmanagement_active_bool</code> ..	308
pdfmeta commands:	
\ <code>\pdfmeta_set_regression_data:</code> ..	5, 485
\ <code>\pdfmeta_standard_get:nN</code>	2, 21, 21
\ <code>\pdfmeta_standard_item:n</code>	2, 17, 17, 115, 117, 125, 127, 456, 461, 467, 1229, 1231, 1232, 1233, 1234, 1310, 1316
\ <code>\pdfmeta_standard_verify:n</code>	2, 25
\ <code>\pdfmeta_standard_verify:nn</code> ..	2, 35
\ <code>\pdfmeta_standard_verify:nnN</code>	2
\ <code>\pdfmeta_standard_verify:nTF</code>	2, 35, 110, 120
\ <code>\pdfmeta_standard_verify:nTF</code>	2, 25, 104, 106, 108, 304, 435
\ <code>\pdfmeta_standard_verify_p:n</code> ..	2, 25
\ <code>\pdfmeta_xmp_add:n</code>	9, 1490, 1490
\ <code>\pdfmeta_xmp_add_declaration:n</code> ..	9, 1503, 1503, 1508
\ <code>\pdfmeta_xmp_add_declaration:nnnn</code>	9, 1509, 1509, 1531, 1540, 1547
\ <code>\pdfmeta_xmp_xmlns_new:nn</code>	9, 1497, 1497
pdfmeta internal commands:	
\ <code>__pdfmeta_embed_colorprofile:n</code> ..	394, 394, 441, 471
\ <code>\g_pdfmeta_outputintents_prop</code> ..	323, 337, 345, 353, 361, 370, 437, 455, 460, 466, 472
\ <code>\g_pdfmeta_standard_pdf/A-1B_-prop</code>	131
\ <code>\g_pdfmeta_standard_pdf/A-2A_-prop</code>	131

\g__pdfmeta_standard_pdf/A-2B_-
 prop [131](#)
 \g__pdfmeta_standard_pdf/A-2U_-
 prop [131](#)
 \g__pdfmeta_standard_pdf/A-3A_-
 prop [131](#)
 \g__pdfmeta_standard_pdf/A-3B_-
 prop [131](#)
 \g__pdfmeta_standard_pdf/A-3U_-
 prop [131](#)
 \g__pdfmeta_standard_pdf/A-4_-
 prop [131](#)
 \g__pdfmeta_standard_prop
 [16](#), [19](#), [23](#), [27](#), [37](#), [45](#), [520](#)
 __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](#), [699](#), [700](#), [706](#), [707](#), [1216](#), [1217](#),
 [1220](#), [1221](#), [1224](#), [1225](#), [1334](#), [1336](#)
 \g__pdfmeta_tmpa_str
 [13](#), [682](#), [683](#), [684](#), [685](#), [686](#), [688](#)
 \l__pdfmeta_tmpa_str
 [10](#), [423](#), [425](#), [751](#),
 [752](#), [761](#), [762](#), [771](#), [772](#), [1283](#), [1284](#),
 [1288](#), [1291](#), [1292](#), [1293](#), [1297](#), [1300](#)
 \l__pdfmeta_tmpa_tl [10](#),
 [315](#), [316](#), [421](#), [423](#), [681](#), [682](#), [781](#),
 [784](#), [786](#), [815](#), [816](#), [823](#), [1216](#), [1218](#),
 [1220](#), [1222](#), [1224](#), [1237](#), [1240](#), [1242](#),
 [1318](#), [1320](#), [1334](#), [1415](#), [1418](#), [1422](#),
 [1425](#), [1454](#), [1457](#), [1513](#), [1524](#), [1528](#)
 \l__pdfmeta_tmpb_seq [10](#)
 \l__pdfmeta_tmpb_tl [10](#),
 [468](#), [469](#), [471](#), [476](#), [481](#), [815](#), [819](#),
 [823](#), [1415](#), [1419](#), [1422](#), [1426](#), [1522](#), [1524](#)
 __pdfmeta_verify_pdfa_annot_-
 flags: [90](#), [105](#)
 __pdfmeta_write_outputintent:nn
 [394](#), [409](#), [443](#), [475](#)
 __pdfmeta_xmp_add_packet_-
 chunk:n [712](#), [712](#), [719](#), [730](#),
 [737](#), [744](#), [752](#), [772](#), [842](#), [874](#), [876](#), [1494](#)
 __pdfmeta_xmp_add_packet_-
 chunk:nN [720](#), [720](#), [727](#), [762](#)
 __pdfmeta_xmp_add_packet_-
 close:nn [741](#), [741](#), [801](#), [802](#),
 [826](#), [827](#), [855](#), [856](#), [870](#), [871](#), [872](#),
 [927](#), [928](#), [930](#), [943](#), [953](#), [1134](#), [1135](#),
 [1136](#), [1137](#), [1138](#), [1174](#), [1175](#), [1176](#),
 [1190](#), [1191](#), [1192](#), [1193](#), [1194](#), [1256](#),
 [1257](#), [1269](#), [1270](#), [1272](#), [1404](#), [1520](#)
 __pdfmeta_xmp_add_packet_-
 field:nnn [946](#), [946](#), [1118](#), [1120](#),
 [1122](#), [1124](#), [1126](#), [1128](#), [1130](#), [1132](#),
 [1166](#), [1168](#), [1170](#), [1172](#), [1186](#), [1188](#)
 __pdfmeta_xmp_add_packet_-
 line:nnn [746](#), [746](#),
 [755](#), [786](#), [798](#), [921](#), [922](#), [923](#), [939](#),
 [940](#), [941](#), [942](#), [950](#), [951](#), [952](#), [1110](#),
 [1111](#), [1113](#), [1114](#), [1158](#), [1159](#), [1161](#),
 [1162](#), [1178](#), [1179](#), [1181](#), [1182](#), [1204](#),
 [1217](#), [1221](#), [1225](#), [1229](#), [1230](#), [1233](#),
 [1234](#), [1235](#), [1239](#), [1241](#), [1263](#), [1276](#),
 [1278](#), [1290](#), [1299](#), [1301](#), [1303](#), [1332](#),
 [1337](#), [1347](#), [1351](#), [1410](#), [1427](#), [1430](#),
 [1433](#), [1436](#), [1439](#), [1442](#), [1445](#), [1448](#),
 [1451](#), [1455](#), [1516](#), [1517](#), [1518](#), [1519](#)
 __pdfmeta_xmp_add_packet_-
 line:nnnN . [756](#), [756](#), [765](#), [1367](#),
 [1371](#), [1375](#), [1379](#), [1383](#), [1387](#), [1391](#)
 __pdfmeta_xmp_add_packet_line_-
 attr:nnnn
 .. [766](#), [766](#), [775](#), [818](#), [822](#), [1416](#), [1423](#)
 __pdfmeta_xmp_add_packet_line_-
 default:nnnn [776](#),
 [776](#), [788](#), [1200](#), [1208](#), [1212](#), [1338](#)
 __pdfmeta_xmp_add_packet_-
 list:nnnn [806](#),
 [830](#), [1308](#), [1312](#), [1314](#), [1325](#), [1327](#), [1342](#)
 __pdfmeta_xmp_add_packet_list_-
 simple:nnnn
 [789](#), [805](#), [1320](#), [1323](#), [1330](#), [1335](#)
 __pdfmeta_xmp_add_packet_-
 open:nn [728](#), [728](#),
 [733](#), [794](#), [795](#), [811](#), [812](#), [844](#), [845](#),
 [849](#), [850](#), [924](#), [925](#), [938](#), [1107](#), [1108](#),
 [1116](#), [1117](#), [1155](#), [1156](#), [1164](#), [1165](#),
 [1184](#), [1185](#), [1250](#), [1251](#), [1266](#), [1267](#)
 __pdfmeta_xmp_add_packet_open_-
 attr:nnn [734](#), [734](#), [740](#), [847](#), [920](#),
 [949](#), [1109](#), [1157](#), [1177](#), [1262](#), [1401](#), [1515](#)
 \g__pdfmeta_xmp_bool
 [487](#), [528](#), [529](#), [1468](#)
 __pdfmeta_xmp_build_dc:
 [862](#), [1306](#), [1306](#)
 __pdfmeta_xmp_build iptc:
 [867](#), [1397](#), [1397](#)

_pdfmeta_xmp_build_iptc_data:N	837, 1363 , 1363	_pdfmeta_xmp_iso_today:	1532, 1543, 1550
_pdfmeta_xmp_build_packet: . . .	831, 831, 1478	_pdfmeta_xmp_lang_get:nNN	696, 710, 815, 1413, 1420
_pdfmeta_xmp_build_pdf:	858, 1198 , 1198	\l_pdfmeta_xmp_lang_regex	694, 699
_pdfmeta_xmp_build_pdfd:	861, 1246 , 1246	\l_pdfmeta_xmp_metalang_tl	692, 702, 816, 834, 835, 836
_pdfmeta_xmp_build_pdfd_-		\g_pdfmeta_xmp_packet_tl	711, 714, 1403, 1480, 1485
claim:nn	1254, 1260 , 1260	\g_pdfmeta_xmp_pdfd_data_prop	1245, 1248, 1252, 1506, 1522, 1526
_pdfmeta_xmp_build_photoshop:	863, 1274 , 1274	_pdfmeta_xmp_print_date:N	620, 620, 1217, 1221, 1225, 1336
_pdfmeta_xmp_build_prism:	866, 1408 , 1408	_pdfmeta_xmp_property_new:nnn 933	
_pdfmeta_xmp_build_standards:	860, 1227 , 1227	_pdfmeta_xmp_property_new:nnnnn	933, 959, 969, 979, 985,
_pdfmeta_xmp_build_user:	868, 1460 , 1460	995, 1005, 1011, 1017, 1023, 1029,
_pdfmeta_xmp_build_xmp:	864, 1206 , 1206	1035, 1041, 1047, 1053, 1059, 1065,
_pdfmeta_xmp_build_xmpMM:	865, 1281 , 1281	_pdfmeta_xmp_sanitize:nN	1071, 1077, 1083, 1089, 1099, 1147
_pdfmeta_xmp_build_xmpRights:	859, 1345 , 1345	675, 675, 691, 751, 761, 771
_pdfmeta_xmp_create_uuid:nN	663, 663, 1286, 1295	_pdfmeta_xmp_schema_enable_-	
\l_pdfmeta_xmp_currentdate_seq	648, 656, 1477	pdfd:	1140, 1196, 1505, 1512
\l_pdfmeta_xmp_currentdate_tl	648, 657, 1296, 1472, 1475, 1477	_pdfmeta_xmp_schema_new:nnn	912, 912,
_pdfmeta_xmp_date_get:nNN	650, 650, 1215, 1219, 1223, 1334	955, 965, 975, 991, 1001, 1095, 1143
\l_pdfmeta_xmp_date_regex	612, 617	\l_pdfmeta_xmp_schema_seq	840, 851, 911 , 915
_pdfmeta_xmp_date_split:nN	615, 615, 619, 660, 1477	\g_pdfmeta_xmp_user_packet_str 1459	
_pdfmeta_xmp_decr_indent:	591, 608, 743, 1395	\g_pdfmeta_xmp_user_packet_tl	1459, 1463, 1492
\l_pdfmeta_xmp_doclang_tl	692, 833, 836, 1331	_pdfmeta_xmp_wtpdf_accessibility_-	
\g_pdfmeta_xmp_export_bool	551, 559, 564, 568, 1482	declaration:	523, 545, 548, 1532 , 1545
\g_pdfmeta_xmp_export_str	552, 560, 569, 1484	_pdfmeta_xmp_wtpdf_reuse_-	
_pdfmeta_xmp_generate_bom:	575, 579, 583, 843	declaration:	524, 539, 542, 1532 , 1538
_pdfmeta_xmp_incr_indent:	591, 603, 731, 738, 1366	_pdfmeta_xmp_xmlns_new:nn	880, 880, 888, 889,
_pdfmeta_xmp_indent:	591, 591, 716, 724	890, 891, 892, 893, 894, 895, 897,
_pdfmeta_xmp_indent:n 591 , 597, 885		898, 899, 900, 901, 902, 903, 904,
\l_pdfmeta_xmp_indent_int	590, 594, 605, 610, 873, 875, 1462, 1464	\g_pdfmeta_xmp_xmlns_prop	905, 906, 907, 908, 909, 910, 1142, 1501
\l_pdfmeta_xmp_iptc_data_tl	837, 838, 1362 , 1399, 1403	\g_pdfmeta_xmp_xmlns_tl 848 , 878 , 883	
		pdfmetatmpa internal commands:	
		\g_pdfmetatmpa_str	10
		pdfuaid~(schema)	975
		PDFversion	1198
		pdfxid~(schema)	991
		photoshop commands:	
		photoshop:AuthorsPosition/pdfauthortitle	1306

<code>\tl_new:N</code>	10, 11,	<code>\tl_use:N</code>	853, 926
	648, 692, 693, 711, 916, 917, 1362, 1459		
<code>\tl_put_right:Nn</code>	722		
<code>\tl_set:Nn</code>	653, 681, 702, 703,		
	706, 707, 781, 784, 833, 834, 1454, 1513	use commands:	
<code>\tl_set_eq:NN</code>	657, 1472	<code>\use:N</code>	42
<code>\tl_to_str:N</code>	679, 682	<code>\use_i:nn</code>	1240
		<code>\use_ii:nn</code>	1242

U