

The **l3pdfmeta** module

PDF standards

L^AT_EX PDF management testphase bundle

The L^AT_EX Project*

Version 0.96b, released 2023-11-05

1 **l3pdfmeta** documentation

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

1.1 Verifying requirements of PDF standards

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

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

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

Verifying against a PDF-standard involves two different tasks:

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

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

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

*E-mail: latex-team@latex-project.org

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

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

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

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

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

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

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

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

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

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

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

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

1.1.1 Simple tests without handler

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

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

`no_encryption` don't encrypt

no_external_content no /F, /FFilter, or /FDecodeParms in stream dictionaries

no_embed_content no /EF key in filespec, no /Type/EmbeddedFiles. *This will be checked in future by l3pdffiles for the files it embeds.* The restriction is set 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*

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:nnn_{Catalog}{OutputIntents}{\langle object reference \rangle}` 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.

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

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.

```

\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.

`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`

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

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

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.
```

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 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 }}
```

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

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 should be added manually.

2.6 PDF related data

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

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

2.7 Document data

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

2.8 User commands

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

<code>\pdfmeta_xmp_add:n</code>	<code>\pdfmeta_xmp_add:n{<XML>}</code>
---------------------------------	--

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

<code>\pdfmeta_xmp_xmlns_new:nn</code>	<code>\pdfmeta_xmp_xmlns_new:nn{<prefix>}{<uri>}</code>
--	---

With this command a xmlns name space can be added.

3 l3pdfmeta implementation

```

1 <@@=pdfmeta>
2 <*header>
3 \ProvidesExplPackage{l3pdfmeta}{2023-11-05}{0.96b}
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_tmppb_tl
\l__pdfmeta_tmpa_str
\g__pdfmetatmpa_str
\l__pdfmeta_tmpa_seq
\l__pdfmeta_tmppb_seq
10 \tl_new:N \l__pdfmeta_tmpa_tl
11 \tl_new:N \l__pdfmeta_tmppb_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

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

\pdfmeta_standard_verify:nTF

```

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

```

(End of definition for \pdfmeta_standard_verify:nTF. This function is documented on page 2.)

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

```

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

```

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

Now we setup a number of handlers.

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

`_standard_verify_handler_min_pdf_version:nn`

```

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

```

(End of definition for __pdfmeta_standard_verify_handler_min_pdf_version:nn.)

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

`_standard_verify_handler_max_pdf_version:nn`

```

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

```

(End of definition for __pdfmeta_standard_verify_handler_max_pdf_version:nn.)

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

ta_standard_verify_handler_named_actions:nn

```

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

```

(End of definition for __pdfmeta_standard_verify_handler_named_actions:nn.)

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

a_standard_verify_handler_annot_action_A:nn

```

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

```

(End of definition for __pdfmeta_standard_verify_handler_annot_action_A:nn.)

This check is probably not needed, but for completeness

ard_verify_handler_outputintent_subtype:nn

```

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

```

(End of definition for __pdfmeta_standard_verify_handler_outputintent_subtype:nn.)

3.1.3 Enforcing requirements

A number of requirements can sensibly be enforced by us.

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

```

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

```

At begin document this should be checked:

```

102 \hook_gput_code:nnn {begindocument} {pdf}
103 {
104   \pdfmeta_standard_verify:nF { annot_flags }
105   { \__pdfmeta_verify_pdfa_annot_flags: }
106   \pdfmeta_standard_verify:nF { Trailer_no_Info }
107   { \__pdf_backend_omit_info:n {1} }
108   \pdfmeta_standard_verify:nF { no_CharSet }
109   { \__pdf_backend_omit_charset:n {1} }
110   \pdfmeta_standard_verify: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: PDAAction, 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}{ }
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}{ }
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}

```

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

3.1.5 Colorprofiles and Outputintents

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

Adding a profile and an intent is technically easy:

1. Embed the profile as stream with

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

2. Write a `/OutputIntent` dictionary for this

```

\pdf_object_unnamed_write:ne {dict}
{
  /Type /OutputIntent
  /S /GTS_PDFA1 % or GTS_PDFX or ISO_PDFE1 or ...

```

```

        /DestOutputProfile \pdf_object_ref_last: % ref the color profile
        /OutputConditionIdentifier ...
        ... %more info
    }

```

3. Reference the dictionary in the catalog:

```

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

```

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

\g_pdfmeta_outputintents_prop

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

```

291 \prop_new:N \g__pdfmeta_outputintents_prop

```

(End of definition for \g__pdfmeta_outputintents_prop.)

Some keys to fill the property.

```

292 \keys_define:nn { document / metadata }
293 {
294     colorprofiles .code:n =
295     {
296         \keys_set:nn { document / metadata / colorprofiles }{#1}
297     }
298 }
299 \keys_define:nn { document / metadata / colorprofiles }
300 {
301     ,A .code:n =
302     {
303         \tl_if_blank:nF {#1}
304         {
305             \prop_gput:Nnn \g__pdfmeta_outputintents_prop
306             { GTS_PDFA1 } {#1}
307         }
308     }
309     ,a .code:n =
310     {
311         \tl_if_blank:nF {#1}
312         {
313             \prop_gput:Nnn \g__pdfmeta_outputintents_prop
314             { GTS_PDFA1 } {#1}
315         }
316     }
317     ,X .code:n =
318     {
319         \tl_if_blank:nF {#1}
320         {
321             \prop_gput:Nnn \g__pdfmeta_outputintents_prop
322             { GTS_PDFX } {#1}
323         }
324     }
325     ,x .code:n =
326     {

```



```

327         \tl_if_blank:nF {#1}
328         {
329             \prop_gput:Nnn \g__pdfmeta_outputintents_prop
330             { GTS_PDFX } {#1}
331         }
332     }
333     ,unknown .code:n =
334     {
335         \tl_if_blank:nF {#1}
336         {
337             \exp_args:NNo
338             \prop_gput:Nnn \g__pdfmeta_outputintents_prop
339             { \l_keys_key_str } {#1}
340         }
341     }
342 }

```

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

```

343 \pdfdict_new:n {l_pdfmeta/outputintent}
344 \pdfdict_put:nnn {l_pdfmeta/outputintent}
345 {Type}{/OutputIntent}
346 \prop_const_from_keyval:cn { c__pdfmeta_colorprofile_sRGB.icc}
347 {
348     ,OutputConditionIdentifier=IEC~sRGB
349     ,Info=IEC-61966-2.1~Default~RGB~colour~space~~~sRGB
350     ,RegistryName=http://www.iec.ch
351     ,N = 3
352 }
353 \prop_const_from_keyval:cn { c__pdfmeta_colorprofile_FOGRA39L_coated.icc}
354 {
355     ,OutputConditionIdentifier=FOGRA39L~Coated
356     ,Info={Offset~printing,~according~to~ISO~12647-2:2004/Amd~1,~OFCOM,~ %
357         paper~type~1~or~2~==~coated~art,~115~g/m2,~tone~value~increase~
358         curves~A~(CMY)~and~B~(K)}
359     ,RegistryName=http://www.fogra.org
360     ,N = 4
361 }

```

_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.

```

362 \cs_new_protected:Npn \_pdfmeta_embed_colorprofile:n #1%#1 file name
363 {
364     \pdf_object_if_exist:nF { __color_icc_ #1 }
365     {
366         \pdf_object_new:n { __color_icc_ #1 }
367         \pdf_object_write:nne { __color_icc_ #1 } { fstream }
368         {
369             {/N\c_space_tl
370                 \prop_item:cn{c__pdfmeta_colorprofile_#1}{N}
371             }
372             {#1}

```

```

373     }
374   }
375 }
376
377 \cs_new_protected:Npn \__pdfmeta_write_outputintent:nn #1 #2 %#1 file name, #2 subtype
378 {
379   \group_begin:
380   \pdfdict_put:nne {l_pdfmeta/outputintent}{S}{/\str_convert_pdfname:n{#2}}
381   \pdfdict_put:nne {l_pdfmeta/outputintent}
382     {DestOutputProfile}
383     {\pdf_object_ref:n{ __color_icc_ #1 }}
384   \clist_map_inline:nn { OutputConditionIdentifier, Info, RegistryName }
385     {
386       \prop_get:cnNT
387       { c__pdfmeta_colorprofile_#1}
388       { ##1 }
389       \l__pdfmeta_tmpa_tl
390       {
391         \pdf_string_from_unicode:nVN {utf8/string}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_st
392         \pdfdict_put:nne
393           {l_pdfmeta/outputintent}{##1}{\l__pdfmeta_tmpa_str}
394       }
395     }
396   \pdf_object_unnamed_write:ne {dict}{\pdfdict_use:n {l_pdfmeta/outputintent} }
397   \pdfmanagement_add:nne {Catalog}{OutputIntents}{\pdf_object_ref_last:}
398   \group_end:
399 }

```

(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

```

400
401 \AddToHook{begindocument/end}
402 {
403   \pdfmeta_standard_verify:nTF {outputintent_A}
404   {
405     \prop_map_inline:Nn \g__pdfmeta_outputintents_prop
406     {
407       \__pdfmeta_embed_colorprofile:n
408       {#2}
409       \__pdfmeta_write_outputintent:nn
410       {#2}
411       {#1}
412     }
413   }

```

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.

```

414   {
415     \exp_args:NNe
416     \prop_if_in:NnF
417       \g__pdfmeta_outputintents_prop
418       { \pdfmeta_standard_item:n { outputintent_A } }
419     {

```

```

420         \exp_args:NNe
421         \prop_gput:Nnn
422         \g__pdfmeta_outputintents_prop
423         { \pdfmeta_standard_item:n { outputintent_A } }
424         { sRGB.icc }
425     }
426     \exp_args:NNe
427     \prop_get:NnN
428     \g__pdfmeta_outputintents_prop
429     { \pdfmeta_standard_item:n { outputintent_A } }
430     \l__pdfmeta_tmpb_tl
431     \exp_args:NV \__pdfmeta_embed_colorprofile:n \l__pdfmeta_tmpb_tl
432     \prop_map_inline:Nn \g__pdfmeta_outputintents_prop
433     {
434         \exp_args:NV
435         \__pdfmeta_write_outputintent:nn
436         \l__pdfmeta_tmpb_tl
437         { #1 }
438     }
439 }
440 }

```

3.2 Regression test

This is simply a copy of the backend function.

```

441 \cs_new_protected:Npn \pdfmeta_set_regression_data:
442 { \__pdf_backend_set_regression_data: }

```

4 XMP-Metadata implementation

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

```

443 \bool_new:N\g__pdfmeta_xmp_bool
444 \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.

```

445 \hook_gput_code:nnn{pdfmanagement/add}{pdfmanagement}
446 {
447     \pdfmanagement_add:nne {Info}{Producer}{(\c_sys_engine_exec_str-\c_sys_engine_version_str)}
448     \pdfmanagement_add:nne {Info}{Creator}{(LaTeX)}
449 }

```

4.1 New document keys

```

450 \keys_define:nn { document / metadata }
451 {
452     _pdfstandard / X-4 .code:n =
453     {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-4}},
454     _pdfstandard / X-4p .code:n =
455     {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-4p}},
456     _pdfstandard / X-5g .code:n =
457     {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5g}},

```

```

458 _pdfstandard / X-5n .code:n =
459   {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5n}},
460 _pdfstandard / X-5pg .code:n =
461   {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5pg}},
462 _pdfstandard / X-6 .code:n =
463   {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6p}},
464 _pdfstandard / X-6n .code:n =
465   {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6n}},
466 _pdfstandard / X-6p .code:n =
467   {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6p}},
468 _pdfstandard / UA-1 .code:n =
469   {
470     \AddToDocumentProperties [document]{pdfstandard-UA}{{1}{2016}}
471   },
472 _pdfstandard / UA-2 .code:n =
473   {\AddToDocumentProperties [document]{pdfstandard-UA}{{2}{2023}}},
474 xmp .bool_gset:N = \g__pdfmeta_xmp_bool
475 }

```

XMP debugging option

```

476 \bool_new:N \g__pdfmeta_xmp_export_bool
477 \str_new:N \g__pdfmeta_xmp_export_str
478
479 \keys_define:nn { document / metadata }
480 {
481   ,debug / xmp-export .choice:
482   ,debug / xmp-export / true .code:n=
483     {
484       \bool_gset_true:N \g__pdfmeta_xmp_export_bool
485       \str_gset_eq:NN \g__pdfmeta_xmp_export_str \c_sys_jobname_str
486     }
487   ,debug / xmp-export / false .code:n =
488     {
489       \bool_gset_false:N \g__pdfmeta_xmp_export_bool
490     }
491   ,debug / xmp-export / unknown .code:n =
492     {
493       \bool_gset_true:N \g__pdfmeta_xmp_export_bool
494       \str_gset:Nn \g__pdfmeta_xmp_export_str { #1 }
495     }
496   ,debug / xmp-export .default:n = true
497 }

```

4.2 Messages

```

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

```

4.3 Some helper commands

4.3.1 Generate a BOM

__pdfmeta_xmp_generate_bom:

```

499 \bool_lazy_or:nnTF
500 { \sys_if_engine luatex_p: }
501 { \sys_if_engine xetex_p: }
502 {

```

```

503     \cs_new:Npn \__pdfmeta_xmp_generate_bom:
504       { \char_generate:nn {"FEFF">{12} }
505     }
506     {
507     \cs_new:Npn \__pdfmeta_xmp_generate_bom:
508       {
509         \char_generate:nn {"EF">{12}
510         \char_generate:nn {"BB">{12}
511         \char_generate:nn {"BF">{12}
512       }
513     }

```

(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
```

```
514 \int_new:N \l__pdfmeta_xmp_indent_int
```

(End of definition for \l__pdfmeta_xmp_indent_int.)

```

\__pdfmeta_xmp_indent:
\__pdfmeta_xmp_indent:n
\__pdfmeta_xmp_incr_indent:
\__pdfmeta_xmp_decr_indent:
515 \cs_new:Npn \__pdfmeta_xmp_indent:
516   {
517     \iow_newline:
518     \prg_replicate:nn {\l__pdfmeta_xmp_indent_int}{\c_space_tl}
519   }
520
521 \cs_new:Npn \__pdfmeta_xmp_indent:n #1
522   {
523     \iow_newline:
524     \prg_replicate:nn {#1}{\c_space_tl}
525   }
526
527 \cs_new_protected:Npn \__pdfmeta_xmp_incr_indent:
528   {
529     \int_incr:N \l__pdfmeta_xmp_indent_int
530   }
531
532 \cs_new_protected:Npn \__pdfmeta_xmp_decr_indent:
533   {
534     \int_decr:N \l__pdfmeta_xmp_indent_int
535   }

```

(End of definition for __pdfmeta_xmp_indent: and others.)

4.3.3 Date and time handling

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

\l__pdfmeta_xmp_date_regex

```
536 \regex_new:N \l__pdfmeta_xmp_date_regex
537 \regex_set:Nn \l__pdfmeta_xmp_date_regex
538 {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.

```
539 \cs_new_protected:Npn \__pdfmeta_xmp_date_split:nN #1 #2 %#1 date, #2 seq
540 {
541   \regex_split:NnN \l__pdfmeta_xmp_date_regex {#1} #2
542 }
543 \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.

```
544 \cs_new:Npn \__pdfmeta_xmp_print_date:N #1 % seq
545 {
546   \tl_if_blank:eTF { \seq_item:Nn #1 {1} }
547   {
548     \seq_item:Nn #1 {2} %year
549     -
550     \seq_item:Nn #1 {3} %month
551     -
552     \seq_item:Nn #1 {4} % day
553     \tl_if_blank:eF
554     { \seq_item:Nn #1 {5} }
555     { T \seq_item:Nn #1 {5} } %hour
556     \tl_if_blank:eF
557     { \seq_item:Nn #1 {6} }
558     { : \seq_item:Nn #1 {6} } %minutes
559     \tl_if_blank:eF
560     { \seq_item:Nn #1 {7} }
561     { : \seq_item:Nn #1 {7} } %seconds
562     \seq_item:Nn #1 {8} %Z,+,-
563     \seq_item:Nn #1 {9}
564     \tl_if_blank:eF
565     { \seq_item:Nn #1 {10} }
566     { : \seq_item:Nn #1 {10} }
567   }
568   {
569     \seq_item:Nn #1 {1}
570   }
571 }
```

(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.

```
572 \tl_new:N \l__pdfmeta_xmp_currentdate_tl
573 \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.

```

574 \cs_new_protected:Npn \__pdfmeta_xmp_date_get:nNN #1 #2 #3
575   %#1 property, #2 tl var with PDF date, #3 seq for splitted date
576   {
577     \tl_set:Nx #2 { \GetDocumentProperties{#1} }
578     \tl_if_blank:VTF #2
579     {
580       \seq_set_eq:NN #3 \l__pdfmeta_xmp_currentdate_seq
581       \tl_set_eq:NN #2 \l__pdfmeta_xmp_currentdate_tl
582     }
583     {
584       \__pdfmeta_xmp_date_split:VN #2 #3
585     }
586   }

```

(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`

```

587 \cs_new_protected:Npn \__pdfmeta_xmp_create_uuid:nN #1 #2
588   {
589     \str_set:Nx#2 {\str_lowercase:f{\tex_mdffivesum:D{#1}}}
590     \str_set:Nx#2
591     {
592       \str_range:Nnn #2{1}{8}
593       -\str_range:Nnn#2{9}{12}
594       -4\str_range:Nnn#2{13}{15}
595       -8\str_range:Nnn#2{16}{18}
596       -\str_range:Nnn#2{19}{30}
597     }
598   }

```

(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.

```

599 \cs_new_protected:Npn \__pdfmeta_xmp_sanitize:nN #1 #2
600   %#1 input string, #2 str with the output
601   {
602     \group_begin:
603     \text_declare_purify_equivalent:Nn \& {\tl_to_str:N & }
604     \text_declare_purify_equivalent:Nn \texttilde {\c_tilde_str}
605     \tl_set:Nx \l__pdfmeta_tmpa_tl { \text_purify:n {#1} }
606     \str_gset:Nx \g__pdfmeta_tmpa_str { \tl_to_str:N \l__pdfmeta_tmpa_tl }
607     \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {\&}{\&}
608     \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {\<}{\<t;}
609     \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {\>}{\>t;}
610     \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {" "}{\"}
611     \group_end:

```

```

612     \str_set_eq:NN #2 \g__pdfmeta_tmpa_str
613   }
614
615   \cs_generate_variant:Nn\__pdfmeta_xmp_sanitiz:nN {VN}

```

(End of definition for __pdfmeta_xmp_sanitiz: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
616 \tl_new:N \l__pdfmeta_xmp_doclang_tl
617 \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
618 \regex_new:N\l__pdfmeta_xmp_lang_regex
619 \regex_set:Nn\l__pdfmeta_xmp_lang_regex {\A\([A-Za-z-]+\)\(.*\)}

620 \cs_new_protected:Npn \__pdfmeta_xmp_lang_get:nNN #1 #2 #3
621 % #1 text, #2 tl var for lang match (or default), #3 tl var for text
622 {
623   \regex_extract_once:NnN \l__pdfmeta_xmp_lang_regex {#1}\l__pdfmeta_tmpa_seq
624   \seq_if_empty:NTF \l__pdfmeta_tmpa_seq
625   {
626     \tl_set:Nn #2 \l__pdfmeta_xmp_metalang_tl
627     \tl_set:Nn #3 {#1}
628   }
629   {
630     \tl_set:Ne #2 {\seq_item:Nn\l__pdfmeta_tmpa_seq{2}}
631     \tl_set:Ne #3 {\seq_item:Nn\l__pdfmeta_tmpa_seq{3}}
632   }
633 }
634 \cs_generate_variant:Nn \__pdfmeta_xmp_lang_get:nNN {eNN,VNN}

```

4.5 Filling the packet

This tl var that holds the whole packet

```

\g__pdfmeta_xmp_packet_tl
635 \tl_new:N \g__pdfmeta_xmp_packet_tl

```

(End of definition for \g__pdfmeta_xmp_packet_tl.)

4.5.1 Helper commands to add lines and lists

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

```

636 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_chunk:n #1
637 {
638   \tl_gput_right:N\g__pdfmeta_xmp_packet_tl
639   {
640     \_pdfmeta_xmp_indent: \exp_not:n{#1}
641   }
642 }
643 \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.

```

644 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_chunk:nN #1 #2
645 {
646   \tl_put_right:N\g__pdfmeta_xmp_packet_tl
647   {
648     \_pdfmeta_xmp_indent: \exp_not:n{#1}
649   }
650 }
651 \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.

```

652 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_open:nn #1 #2 % #1 prefix #2 name
653 {
654   \_pdfmeta_xmp_add_packet_chunk:n {<#1:#2>}
655   \_pdfmeta_xmp_incr_indent:
656 }
657 \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.

```

658 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_open_attr:nnn #1 #2 #3
659 % #1 prefix #2 name #3 attr
660 {
661   \_pdfmeta_xmp_add_packet_chunk:n {<#1:#2~#3>}
662   \_pdfmeta_xmp_incr_indent:
663 }
664 \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.

```

665 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_close:nn #1 #2 % #1 prefix #2: name
666 {
667   \_pdfmeta_xmp_decr_indent:
668   \_pdfmeta_xmp_add_packet_chunk:n {</#1:#2>}
669 }

```

(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.

```

670 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_line:nnn #1 #2 #3
671   {%#1 prefix #2 name #3 content
672   {
673     \tl_if_blank:nF {#3}
674     {
675       \_pdfmeta_xmp_sanitizize:nN {#3}\l__pdfmeta_tmpa_str
676       \_pdfmeta_xmp_add_packet_chunk:e {<#1:#2>\l__pdfmeta_tmpa_str</#1:#2>}
677     }
678   }
679 \cs_generate_variant:Nn \_pdfmeta_xmp_add_packet_line:nnn {nne,nnV,nnee}

```

(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.

```

680 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_line:nnnN #1 #2 #3 #4
681   {%#1 prefix #2 name #3 content #4 tl_var to prebuilt.
682   {
683     \tl_if_blank:nF {#3}
684     {
685       \_pdfmeta_xmp_sanitizize:nN {#3}\l__pdfmeta_tmpa_str
686       \_pdfmeta_xmp_add_packet_chunk:eN {<#1:#2>\l__pdfmeta_tmpa_str</#1:#2>} #4
687     }
688   }
689 \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

```

690 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_line_attr:nnnn #1 #2 #3 #4
691   {%#1 prefix #2 name #3 attribute #4 content
692   {
693     \tl_if_blank:nF {#4}
694     {
695       \_pdfmeta_xmp_sanitizize:nN {#4}\l__pdfmeta_tmpa_str
696       \_pdfmeta_xmp_add_packet_chunk:e {<#1:#2-#3>\l__pdfmeta_tmpa_str</#1:#2>}
697     }
698   }
699 \cs_generate_variant:Nn \_pdfmeta_xmp_add_packet_line_attr:nnnn {nnee,nnneV}

```

(End of definition for _pdfmeta_xmp_add_packet_line_attr:nnnn.)

_pdfmeta_xmp_add_packet_line_default:nnnn

```

700 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_line_default:nnnn #1 #2 #3 #4
701   {% #1 prefix #2 name #3 default #4 content
702   {
703     \tl_if_blank:nTF { #4 }
704     {
705       \tl_set:Nn \l__pdfmeta_tmpa_tl {#3}

```

```

706     }
707     {
708         \tl_set:Nn \l__pdfmeta_tmpa_tl {#4}
709     }
710     \__pdfmeta_xmp_add_packet_line:nnV {#1}{#2}\l__pdfmeta_tmpa_tl
711 }
712 \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:

```

713 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_list_simple:nnnn #1 #2 #3 #4
714 % #1 prefix, #2 name, #3 type (Seq/Bag/Alt) #4 a clist
715 {
716     \clist_if_empty:nF { #4 }
717     {
718         \__pdfmeta_xmp_add_packet_open:nn {#1}{#2}
719         \__pdfmeta_xmp_add_packet_open:nn {rdf}{#3}
720         \clist_map_inline:nn {#4}
721         {
722             \__pdfmeta_xmp_add_packet_line:nnn
723             {rdf}{li}{##1}
724         }
725         \__pdfmeta_xmp_add_packet_close:nn{rdf}{#3}
726         \__pdfmeta_xmp_add_packet_close:nn {#1}{#2}
727     }
728 }
729 \cs_generate_variant:Nn \__pdfmeta_xmp_add_packet_list_simple:nnnn {nnnV,nnne}

```

Here we check also for the language.

```

730 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_list:nnnn #1 #2 #3 #4
731 % #1 prefix, #2 name, #3 type (Seq/Bag/Alt) #4 a clist
732 {
733     \clist_if_empty:nF { #4 }
734     {
735         \__pdfmeta_xmp_add_packet_open:nn {#1}{#2}
736         \__pdfmeta_xmp_add_packet_open:nn {rdf}{#3}
737         \clist_map_inline:nn {#4}
738         {
739             \__pdfmeta_xmp_lang_get:nNN {##1}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl
740             \__pdfmeta_xmp_add_packet_line_attr:nneV
741             {rdf}{li}{xml:lang="\l__pdfmeta_tmpa_tl" }\l__pdfmeta_tmpb_tl
742         }
743         \__pdfmeta_xmp_add_packet_close:nn{rdf}{#3}
744         \__pdfmeta_xmp_add_packet_close:nn {#1}{#2}
745     }
746 }
747 \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.

```

748 \cs_new_protected:Npn \__pdfmeta_xmp_build_packet:

```

```
749 {
```

Get the main languages

```
750 \tl_set:Nx \l__pdfmeta_xmp_doclang_tl {\GetDocumentProperties{document/lang}}
751 \tl_set:Nx \l__pdfmeta_xmp_metalang_tl {\GetDocumentProperties{hyperref/pdfmetalang}}
752 \tl_if_blank:VT \l__pdfmeta_xmp_metalang_tl
753 { \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

```
754 \__pdfmeta_xmp_build_iptc_data:N \l__pdfmeta_xmp_iptc_data_tl
755 \tl_if_empty:NT \l__pdfmeta_xmp_iptc_data_tl
756 {
757   \seq_remove_all:Nn \l__pdfmeta_xmp_schema_seq { Iptc4xmpCore }
758 }
```

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

```
759 \__pdfmeta_xmp_add_packet_chunk:e
760 {<?xpacket~begin="\__pdfmeta_xmp_generate_bom:"~id="W5MOMpCehiHzreSzNTczkc9d"?>}
761 \__pdfmeta_xmp_add_packet_open:nn{x}{xmpmeta~xmlns:x="adobe:ns:meta/"}
762 \__pdfmeta_xmp_add_packet_open:ne{rdf}
763 {RDF~xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns~c_hash_str"}
```

The rdf namespaces

```
764 \__pdfmeta_xmp_add_packet_open_attr:nne
765 {rdf}{Description}{rdf:about="" \g__pdfmeta_xmp_xmlns_tl}
```

The extensions

```
766 \__pdfmeta_xmp_add_packet_open:nn{pdfaExtension}{schemas}
767 \__pdfmeta_xmp_add_packet_open:nn {rdf}{Bag}
768 \seq_map_inline:Nn \l__pdfmeta_xmp_schema_seq
769 {
770   \tl_use:c { g__pdfmeta_xmp_schema_##1_tl }
771 }
772 \__pdfmeta_xmp_add_packet_close:nn {rdf}{Bag}
773 \__pdfmeta_xmp_add_packet_close:nn {pdfaExtension}{schemas}
```

Now starts the part with the data.

```
774 % data
775 \__pdfmeta_xmp_build_pdf:
776 \__pdfmeta_xmp_build_xmpRights:
777 \__pdfmeta_xmp_build_standards: %pdfaid,pdfxid,pdfuaid
778 \__pdfmeta_xmp_build_dc:
779 \__pdfmeta_xmp_build_photoshop:
780 \__pdfmeta_xmp_build_xmp:
781 \__pdfmeta_xmp_build_xmpMM:
782 \__pdfmeta_xmp_build_prism:
783 \__pdfmeta_xmp_build_iptc:
784 \__pdfmeta_xmp_build_user: %user additions
785 % end
786 \__pdfmeta_xmp_add_packet_close:nn {rdf}{Description}
787 \__pdfmeta_xmp_add_packet_close:nn {rdf}{RDF}
788 \__pdfmeta_xmp_add_packet_close:nn {x}{xmpmeta}
789 \int_set:Nn \l__pdfmeta_xmp_indent_int{20}
790 \prg_replicate:nn{10}{\__pdfmeta_xmp_add_packet_chunk:n {}}
791 \int_zero:N \l__pdfmeta_xmp_indent_int
```

```

792     \__pdfmeta_xmp_add_packet_chunk:n {<?xpacket~end="w"?>}
793 }

```

(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.

```

794 \str_new:N \g__pdfmeta_xmp_xmlns_tl
795 \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

```

```

796 \cs_new_protected:Npn \__pdfmeta_xmp_xmlns_new:nn #1 #2
797 {
798     \prop_gput:Nnn \g__pdfmeta_xmp_xmlns_prop {#1}{#2}
799     \tl_gput_right:Ne \g__pdfmeta_xmp_xmlns_tl
800     {
801         \__pdfmeta_xmp_indent:n{4} xmlns:\exp_not:n{#1="#2"}
802     }
803 }
804 \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

```

805 \__pdfmeta_xmp_xmlns_new:nn {pdf}      {http://ns.adobe.com/pdf/1.3/}
806 \__pdfmeta_xmp_xmlns_new:nn {xmpRights}{http://ns.adobe.com/xap/1.0/rights/}
807 \__pdfmeta_xmp_xmlns_new:nn {dc}       {http://purl.org/dc/elements/1.1/}
808 \__pdfmeta_xmp_xmlns_new:nn {photoshop}{http://ns.adobe.com/photoshop/1.0/}
809 \__pdfmeta_xmp_xmlns_new:nn {xmp}      {http://ns.adobe.com/xap/1.0/}
810 \__pdfmeta_xmp_xmlns_new:nn {xmpMM}    {http://ns.adobe.com/xap/1.0/mm/}
811 \__pdfmeta_xmp_xmlns_new:ne {stEvt}    {http://ns.adobe.com/xap/1.0/sType/ResourceEvent\c_hash_str}
812 \__pdfmeta_xmp_xmlns_new:nn {pdfaid}   {http://www.aiim.org/pdfa/ns/id/}
813 \__pdfmeta_xmp_xmlns_new:nn {pdfuaid}  {http://www.aiim.org/pdfua/ns/id/}
814 \__pdfmeta_xmp_xmlns_new:nn {pdfx}     {http://ns.adobe.com/pdfx/1.3/}
815 \__pdfmeta_xmp_xmlns_new:nn {pdfxid}   {http://www.npes.org/pdfx/ns/id/}
816 \__pdfmeta_xmp_xmlns_new:nn {prism}    {http://prismstandard.org/namespaces/basic/3.0/}
817 \__pdfmeta_xmp_xmlns_new:nn {jav}      {http://www.niso.org/schemas/jav/1.0/}
818 \__pdfmeta_xmp_xmlns_new:nn {xmpIPg}   {http://ns.adobe.com/xap/1.0/t/pg/}
819 \__pdfmeta_xmp_xmlns_new:ne {stFnt}    {http://ns.adobe.com/xap/1.0/sType/Font\c_hash_str}
820 \__pdfmeta_xmp_xmlns_new:nn {Iptc4xmpCore}{http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
821 \__pdfmeta_xmp_xmlns_new:nn {pdfaExtension}{http://www.aiim.org/pdfa/ns/extension/}
822 \__pdfmeta_xmp_xmlns_new:ne {pdfaSchema}{http://www.aiim.org/pdfa/ns/schema\c_hash_str}
823 \__pdfmeta_xmp_xmlns_new:ne {pdfaProperty}{http://www.aiim.org/pdfa/ns/property\c_hash_str}
824 \__pdfmeta_xmp_xmlns_new:ne {pdfaType} {http://www.aiim.org/pdfa/ns/type\c_hash_str}
825 \__pdfmeta_xmp_xmlns_new:ne {pdfaField}{http://www.aiim.org/pdfa/ns/field\c_hash_str}
826 \__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

```
827 \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.

```
828 \cs_new_protected:Npn \__pdfmeta_xmp_schema_new:nnn #1 #2 #3
829   {%#1 name #2 prefix, #3 text
830   {
831     \seq_put_right:Nn \l__pdfmeta_xmp_schema_seq { #2 }
832     \tl_new:c { g__pdfmeta_xmp_schema_#2_tl }
833     \tl_new:c { g__pdfmeta_xmp_schema_#2_properties_tl }
834     \tl_gput_right:cn { g__pdfmeta_xmp_schema_#2_tl }
835     {
836       \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
837       \__pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{schema}{#1}
838       \__pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{prefix}{#2}
839       \__pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{namespaceURI}{#3}
840       \__pdfmeta_xmp_add_packet_open:nn {pdfaSchema}{property}
841       \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
842       \tl_use:c { g__pdfmeta_xmp_schema_#2_properties_tl }
843       \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
844       \__pdfmeta_xmp_add_packet_close:nn {pdfaSchema}{property}
845       \cs_if_exist_use:c {__pdfmeta_xmp_schema_#2_additions:}
846       \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
847     }
848   }
```

(End of definition for __pdfmeta_xmp_schema_new:nnn.)

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

```
849 \cs_new_protected:Npn \__pdfmeta_xmp_property_new:nnnnn #1 #2 #3 #4 #5 %
850   {%#1 schema #2 name, #3 type, #4 category #5 description
851   {
852     \tl_gput_right:cn { g__pdfmeta_xmp_schema_#1_properties_tl }
853     {
854       \__pdfmeta_xmp_add_packet_open:nn {rdf}{li~rdf:parseType="Resource"}
855       \__pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{name}{#2}
856       \__pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{valueType}{#3}
857       \__pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{category}{#4}
858       \__pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{description}{#5}
859       \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
860     }
861   }
```

(End of definition for `__pdfmeta_xmp_property_new:nnn`.)

`__pdfmeta_xmp_add_packet_field:nnn`

This adds a field to a schema.

```
862 \cs_new_protected:Npn __pdfmeta_xmp_add_packet_field:nnn #1 #2 #3 %
863   %#1 name #2 valuetype #3 description
864   {
865     __pdfmeta_xmp_add_packet_open_attr:nnn {rdf}{li}{rdf:parseType="Resource"}
866     __pdfmeta_xmp_add_packet_line:nnn {pdfaField}{name}{#1}
867     __pdfmeta_xmp_add_packet_line:nnn {pdfaField}{valueType}{#2}
868     __pdfmeta_xmp_add_packet_line:nnn {pdfaField}{description}{#3}
869     __pdfmeta_xmp_add_packet_close:nn{rdf}{li}
870   }
```

(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.

```
871 __pdfmeta_xmp_schema_new:nnn
872   {XMP~Media~Management~Schema}
873   {xmpMM}
874   {http://ns.adobe.com/xap/1.0/mm/}
875 __pdfmeta_xmp_property_new:nnnnn
876   {xmpMM}
877   {OriginalDocumentID}
878   {URI}
879   {internal}
880   {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)

```
881 __pdfmeta_xmp_schema_new:nnn
882   {PDF/A~Identification~Schema}
883   {pdfaid}
884   {http://www.aiim.org/pdfa/ns/id/}
```

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.

```

885     \_pdfmeta_xmp_property_new:nnnnn
886         {pdfaid}
887         {year}
888         {Integer}
889         {internal}
890         {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)

```

891     \_pdfmeta_xmp_schema_new:nnn
892         {PDF/UA~Universal~Accessibility~Schema}
893         {pdfuaid}
894         {http://www.aiim.org/pdfua/ns/id/}
895     \_pdfmeta_xmp_property_new:nnnnn
896         {pdfuaid}
897         {part}
898         {Integer}
899         {internal}
900         {Part~of~ISO~14289~standard}
901     \_pdfmeta_xmp_property_new:nnnnn
902         {pdfuaid}
903         {rev}
904         {Integer}
905         {internal}
906         {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)

```

907     \_pdfmeta_xmp_schema_new:nnn
908         {PDF/X~ID~Schema}
909         {pdfxid}
910         {http://www.npes.org/pdfx/ns/id/}
911     \_pdfmeta_xmp_property_new:nnnnn
912         {pdfxid}
913         {GTS_PDFXVersion}
914         {Text}
915         {internal}
916         {ID~of~PDF/X~standard}

```

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

prism~(schPrism)

```
917 \_pdfmeta_xmp_schema_new:nnn
918 {PRISM~Basic~Metadata}
919 {prism}
920 {http://prismstandard.org/namespaces/basic/3.0/}
921 \_pdfmeta_xmp_property_new:nnnnn
922 {prism}
923 {complianceProfile}
924 {Text}
925 {internal}
926 {PRISM~specification~compliance~profile~to~which~this~document~adheres}
927 \_pdfmeta_xmp_property_new:nnnnn
928 {prism}
929 {publicationName}
930 {Text}
931 {external}
932 {Publication~name}
933 \_pdfmeta_xmp_property_new:nnnnn
934 {prism}
935 {aggregationType}
936 {Text}
937 {external}
938 {Publication~type}
939 \_pdfmeta_xmp_property_new:nnnnn
940 {prism}
941 {bookEdition}
942 {Text}
943 {external}
944 {Edition~of~the~book~in~which~the~document~was~published}
945 \_pdfmeta_xmp_property_new:nnnnn
946 {prism}
947 {volume}
948 {Text}
949 {external}
950 {Publication~volume~number}
951 \_pdfmeta_xmp_property_new:nnnnn
952 {prism}
953 {number}
954 {Text}
955 {external}
956 {Publication~issue~number~within~a~volume}
957 \_pdfmeta_xmp_property_new:nnnnn
958 {prism}
959 {pageRange}
960 {Text}
961 {external}
962 {Page~range~for~the~document~within~the~print~version~of~its~publication}
963 \_pdfmeta_xmp_property_new:nnnnn
964 {prism}
965 {issn}
966 {Text}
967 {external}
968 {ISSN~for~the~printed~publication~in~which~the~document~was~published}
```

```

969 \_pdfmeta_xmp_property_new:nnnnn
970 {prism}
971 {eIssn}
972 {Text}
973 {external}
974 {ISSN~for~the~electronic~publication~in~which~the~document~was~published}
975 \_pdfmeta_xmp_property_new:nnnnn
976 {prism}
977 {isbn}
978 {Text}
979 {external}
980 {ISBN~for~the~publication~in~which~the~document~was~published}
981 \_pdfmeta_xmp_property_new:nnnnn
982 {prism}
983 {doi}
984 {Text}
985 {external}
986 {Digital~Object~Identifier~for~the~document}
987 \_pdfmeta_xmp_property_new:nnnnn
988 {prism}
989 {url}
990 {URL}
991 {external}
992 {URL~at~which~the~document~can~be~found}
993 \_pdfmeta_xmp_property_new:nnnnn
994 {prism}
995 {byteCount}
996 {Integer}
997 {internal}
998 {Approximate~file~size~in~octets}
999 \_pdfmeta_xmp_property_new:nnnnn
1000 {prism}
1001 {pageCount}
1002 {Integer}
1003 {internal}
1004 {Number~of~pages~in~the~print~version~of~the~document}
1005 \_pdfmeta_xmp_property_new:nnnnn
1006 {prism}
1007 {subtitle}
1008 {Text}
1009 {external}
1010 {Document's~subtitle}

```

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

iptc

```

1011 \_pdfmeta_xmp_schema_new:nnn
1012 {IPTC~Core~Schema}
1013 {Iptc4xmpCore}
1014 {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
1015 \_pdfmeta_xmp_property_new:nnnnn
1016 {Iptc4xmpCore}
1017 {CreatorContactInfo}
1018 {ContactInfo}

```

```

1019     {external}
1020     {Document~creator's~contact~information}
1021 \cs_new_protected:cpn { __pdfmeta_xmp_schema_Iptc4xmpCore_additions: }
1022     {
1023     \__pdfmeta_xmp_add_packet_open:nn{pdfaSchema}{valueType}
1024     \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1025     \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1026     \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{ContactInfo}
1027     \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
1028     {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
1029     \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{Iptc4xmpCore}
1030     \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
1031     {Basic~set~of~information~to~get~in~contact~with~a~person}
1032     \__pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
1033     \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1034     \__pdfmeta_xmp_add_packet_field:nnn{CiAdrCity}{Text}
1035     {Contact~information~city}
1036     \__pdfmeta_xmp_add_packet_field:nnn{CiAdrCtry}{Text}
1037     {Contact~information~country}
1038     \__pdfmeta_xmp_add_packet_field:nnn{CiAdrExtadr}{Text}
1039     {Contact~information~address}
1040     \__pdfmeta_xmp_add_packet_field:nnn{CiAdrPcode}{Text}
1041     {Contact~information~local~postal~code}
1042     \__pdfmeta_xmp_add_packet_field:nnn{CiAdrRegion}{Text}
1043     {Contact~information~regional~information~such~as~state~or~province}
1044     \__pdfmeta_xmp_add_packet_field:nnn{CiEmailWork}{Text}
1045     {Contact~information~email~address(es)}
1046     \__pdfmeta_xmp_add_packet_field:nnn{CiTelWork}{Text}
1047     {Contact~information~telephone~number(s)}
1048     \__pdfmeta_xmp_add_packet_field:nnn{CiUrlWork}{Text}
1049     {Contact~information~Web~URL(s)}
1050     \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1051     \__pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
1052     \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1053     \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1054     \__pdfmeta_xmp_add_packet_close:nn{pdfaSchema}{valueType}
1055     }

```

jav : currently ignored

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 1056 \cs_new_protected:Npn \__pdfmeta_xmp_build_pdf:
    PDFversion         1057 {

```

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

```

1058   \__pdfmeta_xmp_add_packet_line_default:nnee
1059   {pdf}{Producer}

```

```

1060     {\c_sys_engine_exec_str-\c_sys_engine_version_str}
1061     {\GetDocumentProperties{hyperref/pdfproducer}}

```

Now the PDF version

```

1062     \__pdfmeta_xmp_add_packet_line:nne{pdf}{PDFVersion}{\pdf_version:}
1063 }

```

(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
  BaseUrl/baseurl
1064 \cs_new_protected:Npn \__pdfmeta_xmp_build_xmp:
1065 {

```

The creator

```

1066   \__pdfmeta_xmp_add_packet_line_default:nnee
1067   {xmp}{CreatorTool}
1068   {LaTeX}
1069   { \GetDocumentProperties{hyperref/pdfcreator} }

```

The baseurl

```

1070   \__pdfmeta_xmp_add_packet_line_default:nnee
1071   {xmp}{BaseUrl}{\}
1072   { \GetDocumentProperties{hyperref/baseurl} }

```

CreationDate

```

1073   \__pdfmeta_xmp_date_get:nNN
1074   {document/creationdate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1075   \__pdfmeta_xmp_add_packet_line:nne{xmp}{CreateDate}{\__pdfmeta_xmp_print_date:N\l__pdfme
1076   \pdfmanagement_add:nne{Info}{CreationDate}{(\l__pdfmeta_tmpa_tl)}

```

ModifyDate

```

1077   \__pdfmeta_xmp_date_get:nNN
1078   {document/moddate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1079   \__pdfmeta_xmp_add_packet_line:nne{xmp}{ModifyDate}{\__pdfmeta_xmp_print_date:N\l__pdfme
1080   \pdfmanagement_add:nne{Info}{ModDate}{(\l__pdfmeta_tmpa_tl)}

```

MetadataDate

```

1081   \__pdfmeta_xmp_date_get:nNN
1082   {hyperref/pdfmetadate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1083   \__pdfmeta_xmp_add_packet_line:nne{xmp}{MetadataDate}{\__pdfmeta_xmp_print_date:N\l__pdfme
1084   }

```

(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:`

```
1085 \cs_new_protected:Npn \__pdfmeta_xmp_build_standards:
1086 {
1087   \__pdfmeta_xmp_add_packet_line:nne {pdfaid}{part}{\pdfmeta_standard_item:n{level}}
1088   \__pdfmeta_xmp_add_packet_line:nne
1089     {pdfaid}{conformance}{\pdfmeta_standard_item:n{conformance}}
1090   \int_compare:nNnTF {0\pdfmeta_standard_item:n{level}}<{4}
1091     {\__pdfmeta_xmp_add_packet_line:nne {pdfaid}{year} {\pdfmeta_standard_item:n{year}}}
1092     {\__pdfmeta_xmp_add_packet_line:nne {pdfaid}{rev} {\pdfmeta_standard_item:n{year}}}
1093   \__pdfmeta_xmp_add_packet_line:nne
1094     {pdfxid}{GTS_PDFXVersion}{\GetDocumentProperties{document/pdfstandard-X}}
1095   \pdfmanagement_get_documentproperties:nNT {document/pdfstandard-UA}\l__pdfmeta_tmpa_tl
1096   {
1097     \__pdfmeta_xmp_add_packet_line:nne
1098       {pdfuaid}{part}{\exp_last_unbraced:No\use_i:nn \l__pdfmeta_tmpa_tl}
1099     \__pdfmeta_xmp_add_packet_line:nne
1100       {pdfuaid}{rev}{\exp_last_unbraced:No\use_ii:nn \l__pdfmeta_tmpa_tl}
1101   }
1102 }
```

(End of definition for __pdfmeta_xmp_build_standards:.)

4.8.4 Photoshop

`__pdfmeta_xmp_build_photoshop:`

```
1103 \cs_new_protected:Npn \__pdfmeta_xmp_build_photoshop:
1104 {
pdfauthortitle/photoshop:AuthorsPosition
1105   \__pdfmeta_xmp_add_packet_line:nne{photoshop}{AuthorsPosition}
1106     { \GetDocumentProperties{hyperref/pdfauthortitle} }
pdfcaptionwriter/photoshop:CaptionWriter
1107   \__pdfmeta_xmp_add_packet_line:nne{photoshop}{CaptionWriter}
1108     { \GetDocumentProperties{hyperref/pdfcaptionwriter} }
1109 }
```

(End of definition for __pdfmeta_xmp_build_photoshop:.)

4.9 XMP Media Management

`__pdfmeta_xmp_build_xmpMM:`

```
1110 \cs_new_protected:Npn \__pdfmeta_xmp_build_xmpMM:
1111 {
```

pdfdocumentid / xmpMM:DocumentID

```

1112     \str_set:Ne\l__pdfmeta_tmpa_str {\GetDocumentProperties{hyperref/pdfdocumentid}}
1113     \str_if_empty:NT \l__pdfmeta_tmpa_str
1114     {
1115         \__pdfmeta_xmp_create_uuid:nN
1116         {\jobname\GetDocumentProperties{hyperref/pdftitle}}
1117         \l__pdfmeta_tmpa_str
1118     }
1119     \__pdfmeta_xmp_add_packet_line:nnV{xmpMM}{DocumentID}
1120     \l__pdfmeta_tmpa_str

```

pdfinstanceid / xmpMM:InstanceID

```

1121     \str_set:Ne\l__pdfmeta_tmpa_str {\GetDocumentProperties{hyperref/pdfinstanceid}}
1122     \str_if_empty:NT \l__pdfmeta_tmpa_str
1123     {
1124         \__pdfmeta_xmp_create_uuid:nN
1125         {\jobname\l__pdfmeta_xmp_currentdate_tl}
1126         \l__pdfmeta_tmpa_str
1127     }
1128     \__pdfmeta_xmp_add_packet_line:nnV{xmpMM}{InstanceID}
1129     \l__pdfmeta_tmpa_str

```

pdfversionid/xmpMM:VersionID

```

1130     \__pdfmeta_xmp_add_packet_line:nne{xmpMM}{VersionID}
1131     { \GetDocumentProperties{hyperref/pdfversionid} }

```

pdfrendition/xmpMM:RenditionClass

```

1132     \__pdfmeta_xmp_add_packet_line:nne{xmpMM}{RenditionClass}
1133     { \GetDocumentProperties{hyperref/pdfrendition} }
1134 }

```

(End of definition for __pdfmeta_xmp_build_xmpMM:.)

4.10 Rest of dublin Core data

__pdfmeta_xmp_build_dc:

```

    dc:creator/pdfauthor 1135 \cs_new_protected:Npn \__pdfmeta_xmp_build_dc:
    dc:subject/pdfkeywords 1136 {

```

dc:type/pdfdtype

pdfauthor/dc:creator

dc:publisher/pdfpublisher

```

1137     \__pdfmeta_xmp_add_packet_list:nnne {dc}{creator}{Seq}
1138     { \GetDocumentProperties{hyperref/pdfauthor} }

```

dc:description/pdfsubject

dc:language/lang/pdflang

```

1139     \int_compare:nNtT {0\pdfmeta_standard_item:n{level}}={1}
1140     { \pdfmanagement_remove:nn{Info}{Author} }

```

dc:identifier/pdfidentifier

photoshop:AuthorsPosition/pdfauthoritle

photoshop:CaptionWriter/pdfcaptionwriter

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

```

1141     \__pdfmeta_xmp_add_packet_list:nnne {dc}{title}{Alt}
1142     { \GetDocumentProperties{hyperref/pdftitle} }

```

pdfkeywords/dc:subject

```

1143     \__pdfmeta_xmp_add_packet_list:nnne {dc}{subject}{Bag}
1144     { \GetDocumentProperties{hyperref/pdfkeywords} }
1145     \int_compare:nNtT {0\pdfmeta_standard_item:n{level}}={1}
1146     { \pdfmanagement_remove:nn{Info}{Keywords} }

```

pdftype/dc:type

```
1147 \pdfmanagement_get_documentproperties:nNTF { hyperref/pdftype } \l__pdfmeta_tmpa_tl
1148 {
1149   \__pdfmeta_xmp_add_packet_list_simple:nnnV {dc}{type}{Bag}\l__pdfmeta_tmpa_tl
1150 }
1151 {
1152   \__pdfmeta_xmp_add_packet_list_simple:nnnn {dc}{type}{Bag}{Text}
1153 }
```

pdfpublisher/dc:publisher

```
1154 \__pdfmeta_xmp_add_packet_list:nnne {dc}{publisher}{Bag}
1155 { \GetDocumentProperties{hyperref/pdfpublisher} }
```

pdfsubject/dc:description

```
1156 \__pdfmeta_xmp_add_packet_list:nnne
1157 {dc}{description}{Alt}
1158 { \GetDocumentProperties{hyperref/pdfsubject} }
```

lang/pdflang/dc:language

```
1159 \__pdfmeta_xmp_add_packet_list_simple:nnnV
1160 {dc}{language}{Bag}\l__pdfmeta_xmp_doclang_tl
```

pdfidentifier/dc:identifier

```
1161 \__pdfmeta_xmp_add_packet_line:nne{dc}{identifier}
1162 { \GetDocumentProperties{hyperref/pdfidentifier} }
```

pdfdate/dc:date

```
1163 \__pdfmeta_xmp_date_get:nnN {hyperref/pdfdate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1164 \__pdfmeta_xmp_add_packet_list_simple:nnne
1165 {dc}{date}{Seq}{\__pdfmeta_xmp_print_date:N\l__pdfmeta_tmpa_seq}
```

The file format

```
1166 \__pdfmeta_xmp_add_packet_line:nnn{dc}{format}{application/pdf}
```

The source

```
1167 \__pdfmeta_xmp_add_packet_line_default:nnee
1168 {dc}{source}
1169 { \c_sys_jobname_str.tex }
1170 { \GetDocumentProperties{hyperref/pdfsource} }
1171 \__pdfmeta_xmp_add_packet_list:nnne{dc}{rights}{Alt}
1172 { \GetDocumentProperties{hyperref/pdfcopyright} }
1173 }
```

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

4.11 xmpRights

__pdfmeta_xmp_build_xmpRights:

```
1174 \cs_new_protected:Npn \__pdfmeta_xmp_build_xmpRights:
1175 {
1176   \__pdfmeta_xmp_add_packet_line:nne
1177   {xmpRights}
1178   {WebStatement}
1179   { \GetDocumentProperties{hyperref/pdflicenseurl} }
1180 }
```

(End of definition for __pdfmeta_xmp_build_xmpRights:.)

4.12 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
```

```
1181 \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
```

```
1182 \cs_new_protected:Npn \__pdfmeta_xmp_build_iptc_data:N #1
```

```
1183 {
```

```
1184   \tl_clear:N #1
```

```
1185   \__pdfmeta_xmp_incr_indent:\__pdfmeta_xmp_incr_indent:\__pdfmeta_xmp_incr_indent:\__pdf
```

```
1186   \__pdfmeta_xmp_add_packet_line:nneN
```

```
1187     {Iptc4xmpCore}{CiAdrExtadr}
```

```
1188     {\GetDocumentProperties{hyperref/pdfcontactaddress}}
```

```
1189     #1
```

```
1190   \__pdfmeta_xmp_add_packet_line:nneN
```

```
1191     {Iptc4xmpCore}{CiAdrCity}
```

```
1192     {\GetDocumentProperties{hyperref/pdfcontactcity}}
```

```
1193     #1
```

```
1194   \__pdfmeta_xmp_add_packet_line:nneN
```

```
1195     {Iptc4xmpCore}{CiAdrPcode}
```

```
1196     {\GetDocumentProperties{hyperref/pdfcontactpostcode}}
```

```
1197     #1
```

```
1198   \__pdfmeta_xmp_add_packet_line:nneN
```

```
1199     {Iptc4xmpCore}{CiAdrCtry}
```

```
1200     {\GetDocumentProperties{hyperref/pdfcontactcountry}}
```

```
1201     #1
```

```
1202   \__pdfmeta_xmp_add_packet_line:nneN
```

```
1203     {Iptc4xmpCore}{CiTelWork}
```

```
1204     {\GetDocumentProperties{hyperref/pdfcontactphone}}
```

```
1205     #1
```

```
1206   \__pdfmeta_xmp_add_packet_line:nneN
```

```
1207     {Iptc4xmpCore}{CiEmailWork}
```

```
1208     {\GetDocumentProperties{hyperref/pdfcontactemail}}
```

```
1209     #1
```

```
1210   \__pdfmeta_xmp_add_packet_line:nneN
```

```
1211     {Iptc4xmpCore}{CiUrlWork}
```

```
1212     {\GetDocumentProperties{hyperref/pdfcontacturl}}
```

```
1213     #1
```

```
1214   \__pdfmeta_xmp_decr_indent:\__pdfmeta_xmp_decr_indent:\__pdfmeta_xmp_decr_indent:\__pdf
```

```
1215 }
```

(End of definition for __pdfmeta_xmp_build_iptc_data:N.)

```
\__pdfmeta_xmp_build_iptc:
```

```
1216 \cs_new_protected:Npn \__pdfmeta_xmp_build_iptc:
```

```
1217 {
```

```
1218   \tl_if_empty:NF\l__pdfmeta_xmp_iptc_data_tl
```

```
1219   {
```

```
1220     \__pdfmeta_xmp_add_packet_open_attr:nnn
```

```
1221     {Iptc4xmpCore}{CreatorContactInfo}{rdf:parseType="Resource"}
```



```

1222         \tl_gput_right:N\g__pdfmeta_xmp_packet_tl { \l__pdfmeta_xmp_iptc_data_tl }
1223         \__pdfmeta_xmp_add_packet_close:nn
1224         {Iptc4xmpCore}{CreatorContactInfo}
1225     }
1226 }

```

(End of definition for __pdfmeta_xmp_build_iptc:.)

4.13 Prism

```

\__pdfmeta_xmp_build_prism:
    complianceProfile 1227 \cs_new_protected:Npn \__pdfmeta_xmp_build_prism:
    prism:subtitle/pdfsubtitle 1228 {

```

The compliance profile is a fix value taken from hyperxmp

```

1229     \__pdfmeta_xmp_add_packet_line:nnn
1230     {prism}{complianceProfile}
1231     {three}

```

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

```

1232     \__pdfmeta_xmp_lang_get:eNN
1233     {\GetDocumentProperties{hyperref/pdfsubtitle}}
1234     \l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl
1235     \__pdfmeta_xmp_add_packet_line_attr:nneV
1236     {prism}{subtitle}
1237     {xml:lang="\l__pdfmeta_tmpa_tl"}
1238     \l__pdfmeta_tmpb_tl

```

Then publicationName

```

1239     \__pdfmeta_xmp_lang_get:eNN
1240     {\GetDocumentProperties{hyperref/pdfpublication}}
1241     \l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl
1242     \__pdfmeta_xmp_add_packet_line_attr:nneV
1243     {prism}{publicationName}
1244     {xml:lang="\l__pdfmeta_tmpa_tl"}
1245     \l__pdfmeta_tmpb_tl

```

Now the rest

```

1246     \__pdfmeta_xmp_add_packet_line:nne
1247     {prism}{bookEdition}
1248     {\GetDocumentProperties{hyperref/pdfbookedition}}
1249     \__pdfmeta_xmp_add_packet_line:nne
1250     {prism}{aggregationType}
1251     {\GetDocumentProperties{hyperref/pdfpubtype}}
1252     \__pdfmeta_xmp_add_packet_line:nne
1253     {prism}{volume}
1254     {\GetDocumentProperties{hyperref/pdfvolumenum}}
1255     \__pdfmeta_xmp_add_packet_line:nne
1256     {prism}{number}
1257     {\GetDocumentProperties{hyperref/pdfissuenum}}
1258     \__pdfmeta_xmp_add_packet_line:nne
1259     {prism}{pageRange}
1260     {\GetDocumentProperties{hyperref/pdfpagerange}}
1261     \__pdfmeta_xmp_add_packet_line:nne
1262     {prism}{issn}

```

```

1263     {\GetDocumentProperties{hyperref/pdfissn}}
1264 \__pdfmeta_xmp_add_packet_line:nne
1265     {prism}{eIssn}
1266     {\GetDocumentProperties{hyperref/pdfeissn}}
1267 \__pdfmeta_xmp_add_packet_line:nne
1268     {prism}{doi}
1269     {\GetDocumentProperties{hyperref/pdfdoi}}
1270 \__pdfmeta_xmp_add_packet_line:nne
1271     {prism}{url}
1272     {\GetDocumentProperties{hyperref/pdfurl}}

```

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

```

1273 \tl_set:N \l__pdfmeta_tmpa_tl { \GetDocumentProperties{hyperref/pdfnumpages} }
1274 \__pdfmeta_xmp_add_packet_line:nne
1275     {prism}{pageCount}
1276     {\tl_if_blank:VTF \l__pdfmeta_tmpa_tl {\PreviousTotalPages}{\l__pdfmeta_tmpa_tl}}
1277 }

```

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

4.13.1 User additions

\g__pdfmeta_xmp_user_packet_str

```

1278 \tl_new:N \g__pdfmeta_xmp_user_packet_tl

```

(End of definition for \g__pdfmeta_xmp_user_packet_str.)

__pdfmeta_xmp_build_user:

```

1279 \cs_new_protected:Npn \__pdfmeta_xmp_build_user:
1280 {
1281     \int_zero:N \l__pdfmeta_xmp_indent_int
1282     \g__pdfmeta_xmp_user_packet_tl
1283     \int_set:Nn \l__pdfmeta_xmp_indent_int {3}
1284 }

```

(End of definition for __pdfmeta_xmp_build_user:.)

4.14 Activating the metadata

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

```

1285 \AddToHook{shipout/lastpage}
1286 {
1287     \bool_if:NT\g__pdfmeta_xmp_bool
1288     {
1289         \str_if_exist:NTF\c_sys_timestamp_str
1290         {
1291             \tl_set_eq:NN \l__pdfmeta_xmp_currentdate_tl \c_sys_timestamp_str
1292         }
1293         {
1294             \file_get_timestamp:nN{\jobname.log}\l__pdfmeta_xmp_currentdate_tl
1295         }
1296         \__pdfmeta_xmp_date_split:VN\l__pdfmeta_xmp_currentdate_tl\l__pdfmeta_xmp_currentdate

```

```

1297     \__pdfmeta_xmp_build_packet:
1298     \exp_args:No
1299     \__pdf_backend_metadata_stream:n {\g__pdfmeta_xmp_packet_tl}
1300     \pdfmanagement_add:nne {Catalog} {Metadata}{\pdf_object_ref_last:}
1301     \bool_if:NT \g__pdfmeta_xmp_export_bool
1302     {
1303         \iow_open:Nn\g_tmpa_iow{\g__pdfmeta_xmp_export_str.xmpi}
1304         \exp_args:NNo\iow_now:Nn\g_tmpa_iow{\g__pdfmeta_xmp_packet_tl}
1305         \iow_close:N\g_tmpa_iow
1306     }
1307 }
1308 }

```

4.15 User commands

`\pdfmeta_xmp_add:n`

```

1309 \cs_new_protected:Npn \pdfmeta_xmp_add:n #1
1310 {
1311     \tl_gput_right:Nn \g__pdfmeta_xmp_user_packet_tl
1312     {
1313         \__pdfmeta_xmp_add_packet_chunk:n { #1 }
1314     }
1315 }

```

(End of definition for `\pdfmeta_xmp_add:n`. This function is documented on page 8.)

`\pdfmeta_xmp_xmlns_new:nn`

```

1316 \cs_new_protected:Npn \pdfmeta_xmp_xmlns_new:nn #1 #2
1317 {
1318     \prop_if_in:NnTF \g__pdfmeta_xmp_xmlns_prop {#1}
1319     {\msg_warning:nnn{pdfmeta}{namespace-defined}{#1}}
1320     {\__pdfmeta_xmp_xmlns_new:nn {#1}{#2}}
1321 }

```

(End of definition for `\pdfmeta_xmp_xmlns_new:nn`. This function is documented on page 8.)

```

1322 </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		A	
<code>\&</code>	603	<code>\A</code>	619
<code>\'</code>	538	<code>\AddToDocumentProperties</code>	453, 455, 457, 459, 461, 463, 465, 467, 470, 473
<code>\+</code>	538	<code>\AddToHook</code>	401, 1285
<code>\-</code>	538, 619		
<code>\[</code>	619	B	
<code>\]</code>	619	<code>BaseUrl/baseurl</code>	1064

bitset commands:

- \bitset_set_false:Nn 93, 94, 95
- \bitset_set_true:Nn 92
- \bitset_to_arabic:N 96, 97, 98, 99, 100

bool commands:

- \bool_gset_false:N 489
- \bool_gset_true:N 444, 484, 493
- \bool_if:NTF 1287, 1301
- \bool_lazy_or:nnTF 499
- \bool_new:N 443, 476

C

char commands:

- \char_generate:nn . . 504, 509, 510, 511

clist commands:

- \clist_if_empty:nTF 716, 733
- \clist_map_inline:nn . . 384, 720, 737

complianceProfile 1227

CreatorTool/pdfcreator 1064

cs commands:

- \cs_generate_variant:Nn
- 543, 615, 634, 643, 651, 657,
- 664, 679, 689, 699, 712, 729, 747, 804
- \cs_if_exist:NTF 39
- \cs_if_exist_use:N 845
- \cs_new:Npn 17, 503, 507, 515, 521, 544
- \cs_new_protected:Npn
- 21, 56, 64, 72, 78, 84, 90,
- 362, 377, 441, 527, 532, 539, 574,
- 587, 599, 620, 636, 644, 652, 658,
- 665, 670, 680, 690, 700, 713, 730,
- 748, 796, 828, 849, 862, 1021, 1056,
- 1064, 1085, 1103, 1110, 1135, 1174,
- 1182, 1216, 1227, 1279, 1309, 1316
- \cs_set_eq:NN 753

D

\d 538

dc commands:

- dc:description/pdfsubject 1135
- dc:identifier/pdfidentifier 1135
- dc:language/lang/pdflang 1135
- dc:Nreator/pdfauthor 1135
- dc:publisher/pdfpublisher 1135
- dc:subject/pdfkeywords 1135
- dc:type/pdftype 1135

\DocumentMetadata 2-4

E

exp commands:

- \exp_args:NNe 415, 420, 426
- \exp_args:Nnne 41
- \exp_args:NNo 337, 1304
- \exp_args:No 1298

- \exp_args:NV 431, 434
- \exp_last_unbraced:No 1098, 1100
- \exp_not:n 640, 648, 801

F

file commands:

- \file_get_timestamp:nN 1294

G

\GetDocumentProperties

- 577, 750, 751, 1061, 1069, 1072,
- 1094, 1106, 1108, 1112, 1116, 1121,
- 1131, 1133, 1138, 1142, 1144, 1155,
- 1158, 1162, 1170, 1172, 1179, 1188,
- 1192, 1196, 1200, 1204, 1208, 1212,
- 1233, 1240, 1248, 1251, 1254, 1257,
- 1260, 1263, 1266, 1269, 1272, 1273

group commands:

- \group_begin: 379, 602
- \group_end: 398, 611

H

hook commands:

- \hook_gput_code:nnn 102, 445

I

int commands:

- \int_compare:nNnTF 1090, 1139, 1145
- \int_decr:N 534
- \int_incr:N 529
- \int_new:N 514
- \int_set:Nn 789, 1283
- \int_zero:N 791, 1281

iow commands:

- \iow_close:N 1305
- \iow_newline: 517, 523
- \iow_now:Nn 1304
- \iow_open:Nn 1303
- \g_tmpa_iow 1303, 1304, 1305

J

\jobname 1116, 1125, 1294

K

keys commands:

- \keys_define:nn 292, 299, 450, 479
- \l_keys_key_str 339
- \keys_set:nn 296

M

msg commands:

- \msg_new:nnn 7, 8, 498
- \msg_warning:nnn 1319
- \msg_warning:nnnnn 112, 122

P

pdf commands:

[\pdf_object_if_exist:nTF](#) 364
[\pdf_object_new:n](#) 366
[\pdf_object_ref:n](#) 383
[\pdf_object_ref_last:](#) 397, 1300
[\pdf_object_unnamed_write:nn](#) .. 396
[\pdf_object_write:nnn](#) 367
[\pdf_string_from_unicode:nnN](#) .. 391
[\pdf_version:](#) [3](#), [111](#), [113](#), [121](#), [123](#), [1062](#)
[\pdf_version_compare:NnTF](#) ... 58, 66

pdf internal commands:

[__pdf_backend_metadata_stream:n](#)
 1299
[__pdf_backend_omit_charset:n](#) .. 109
[__pdf_backend_omit_info:n](#) 107
[__pdf_backend_set_regression_-
 data:](#) 442

[pdfaid~\(schema\)](#) [881](#)

pdfannot commands:

[\pdfannot_dict_put:nnn](#)
 96, 97, 98, 99, 100
[\l_pdfannot_F_bitset](#)
 ... 92, 93, 94, 95, 96, 97, 98, 99, 100

pdfdict commands:

[\pdfdict_new:n](#) 343
[\pdfdict_put:nnn](#) ... 344, 380, 381, 392
[\pdfdict_use:n](#) 396

pdfmanagement commands:

[\pdfmanagement_add:nnn](#)
 397, 447, 448, 1076, 1080, 1300
[\pdfmanagement_get_documentproperties:nNTF](#)
 1095, 1147
[\pdfmanagement_remove:nn](#) . 1140, 1146

pdfmeta commands:

[\pdfmeta_set_regression_data:](#) 5, 441
[\pdfmeta_standard_get:nN](#) ... [2](#), [21](#), [21](#)
[\pdfmeta_standard_item:n](#)
 [2](#), [17](#), [17](#), [115](#),
[117](#), [125](#), [127](#), [418](#), [423](#), [429](#), [1087](#),
[1089](#), [1090](#), [1091](#), [1092](#), [1139](#), [1145](#)
[\pdfmeta_standard_verify:n](#) ... [2](#), [25](#)
[\pdfmeta_standard_verify:nn](#) .. [2](#), [35](#)
[\pdfmeta_standard_verify:nnN](#) [2](#)
[\pdfmeta_standard_verify:nnTF](#) ...
 [2](#), [35](#), [110](#), [120](#)
[\pdfmeta_standard_verify:nTF](#) ...
 [2](#), [25](#), [104](#), [106](#), [108](#), [403](#)
[\pdfmeta_standard_verify_p:n](#) . [2](#), [25](#)
[\pdfmeta_xmp_add:n](#) [8](#), [1309](#), [1309](#)
[\pdfmeta_xmp_xmlns_new:nn](#)
 [8](#), [1316](#), [1316](#)

pdfmeta internal commands:

[__pdfmeta_embed_colorprofile:n](#) .
 362, 362, 407, 431
[\g__pdfmeta_outputintents_prop](#) ..
 [291](#), [305](#), [313](#),
[321](#), [329](#), [338](#), [405](#), [417](#), [422](#), [428](#), [432](#)
[\g__pdfmeta_standard_pdf/A-1B_-
 prop](#) [131](#)
[\g__pdfmeta_standard_pdf/A-2A_-
 prop](#) [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](#)
[__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](#), [623](#), [624](#), [630](#), [631](#), [1074](#), [1075](#),
[1078](#), [1079](#), [1082](#), [1083](#), [1163](#), [1165](#)
[\g__pdfmeta_tmpa_str](#)
 [13](#), [606](#), [607](#), [608](#), [609](#), [610](#), [612](#)
[\l__pdfmeta_tmpa_str](#)
 [10](#), [391](#), [393](#), [675](#),
[676](#), [685](#), [686](#), [695](#), [696](#), [1112](#), [1113](#),
[1117](#), [1120](#), [1121](#), [1122](#), [1126](#), [1129](#)
[\l__pdfmeta_tmpa_tl](#) [10](#), [389](#),
[391](#), [605](#), [606](#), [705](#), [708](#), [710](#), [739](#),
[741](#), [1074](#), [1076](#), [1078](#), [1080](#), [1082](#),
[1095](#), [1098](#), [1100](#), [1147](#), [1149](#), [1163](#),
[1234](#), [1237](#), [1241](#), [1244](#), [1273](#), [1276](#)
[\l__pdfmeta_tmpb_seq](#) [10](#)
[\l__pdfmeta_tmpb_tl](#) . [10](#), [430](#), [431](#),
[436](#), [739](#), [741](#), [1234](#), [1238](#), [1241](#), [1245](#)
[__pdfmeta_verify_pdfa_annot_-
 flags:](#) 90, 105

__pdfmeta_write_outputintent:nn	__pdfmeta_xmp_build_packet: . . .
..... 362, 377, 409, 435 748, 748, 1297
__pdfmeta_xmp_add_packet_-	__pdfmeta_xmp_build_pdf:
chunk:n 636, 636, 643, 654, 775, 1056, 1056
661, 668, 676, 696, 759, 790, 792, 1313	__pdfmeta_xmp_build_photshop: .
__pdfmeta_xmp_add_packet_- 779, 1103, 1103
chunk:nN 644, 644, 651, 686	__pdfmeta_xmp_build_prism:
__pdfmeta_xmp_add_packet_- 782, 1227, 1227
close:nn 665,	__pdfmeta_xmp_build_standards: .
665, 725, 726, 743, 744, 772, 773, 777, 1085, 1085
786, 787, 788, 843, 844, 846, 859,	__pdfmeta_xmp_build_user:
869, 1050, 1051, 1052, 1053, 1054, 1223 784, 1279, 1279
__pdfmeta_xmp_add_packet_-	__pdfmeta_xmp_build_xmp:
field:nnn 862, 862, 1034, 1036, 780, 1064, 1064
1038, 1040, 1042, 1044, 1046, 1048	__pdfmeta_xmp_build_xmpMM:
__pdfmeta_xmp_add_packet_- 781, 1110, 1110
line:nnn 670,	__pdfmeta_xmp_build_xmpRights: .
670, 679, 710, 722, 837, 838, 839, 776, 1174, 1174
855, 856, 857, 858, 866, 867, 868,	__pdfmeta_xmp_create_uuid:nN . . .
1026, 1027, 1029, 1030, 1062, 1075, 587, 587, 1115, 1124
1079, 1083, 1087, 1088, 1091, 1092,	\l__pdfmeta_xmp_currentdate_seq .
1093, 1097, 1099, 1105, 1107, 1119, 572, 580, 1296
1128, 1130, 1132, 1161, 1166, 1176,	\l__pdfmeta_xmp_currentdate_tl . .
1229, 1246, 1249, 1252, 1255, 572, 581, 1125, 1291, 1294, 1296
1258, 1261, 1264, 1267, 1270, 1274	__pdfmeta_xmp_date_get:nNN
__pdfmeta_xmp_add_packet_- 574, 574, 1073, 1077, 1081, 1163
line:nnnN . 680, 680, 689, 1186,	\l__pdfmeta_xmp_date_regex . 536, 541
1190, 1194, 1198, 1202, 1206, 1210	__pdfmeta_xmp_date_split:nN . . .
__pdfmeta_xmp_add_packet_line_- 539, 539, 543, 584, 1296
attr:nnnn	__pdfmeta_xmp_decr_indent:
690, 690, 699, 740, 1235, 1242 515, 532, 667, 1214
__pdfmeta_xmp_add_packet_line_-	\l__pdfmeta_xmp_doclang_tl
default:nnnn 700, 616, 750, 753, 1160
700, 712, 1058, 1066, 1070, 1167	\g__pdfmeta_xmp_export_bool
__pdfmeta_xmp_add_packet_- 476, 484, 489, 493, 1301
list:nnnn 730,	\g__pdfmeta_xmp_export_str
747, 1137, 1141, 1143, 1154, 1156, 1171 477, 485, 494, 1303
__pdfmeta_xmp_add_packet_list_-	__pdfmeta_xmp_generate_bom: . . .
simple:nnnn 499, 503, 507, 760
713, 729, 1149, 1152, 1159, 1164	__pdfmeta_xmp_incr_indent:
__pdfmeta_xmp_add_packet_- 515, 527, 655, 662, 1185
open:nn 652, 652, 657, 718,	__pdfmeta_xmp_indent:
719, 735, 736, 761, 762, 766, 767, 515, 515, 640, 648
840, 841, 854, 1023, 1024, 1032, 1033	__pdfmeta_xmp_indent:n 515, 521, 801
__pdfmeta_xmp_add_packet_open_-	\l__pdfmeta_xmp_indent_int . 514,
attr:nnn 658,	518, 529, 534, 789, 791, 1281, 1283
658, 664, 764, 836, 865, 1025, 1220	\l__pdfmeta_xmp_iptc_data_tl . . .
\g__pdfmeta_xmp_bool . 443, 474, 1287 754, 755, 1181, 1218, 1222
__pdfmeta_xmp_build_dc:	__pdfmeta_xmp_lang_get:nNN
..... 778, 1135, 1135 620, 634, 739, 1232, 1239
__pdfmeta_xmp_build_iptc:	\l__pdfmeta_xmp_lang_regex . 618, 623
..... 783, 1216, 1216	\l__pdfmeta_xmp_metalang_tl
__pdfmeta_xmp_build_iptc_data:N 616, 626, 751, 752, 753
..... 754, 1182, 1182	

<code>\sys_if_engine_xetex_p:</code>	501	564, 578, 673, 683, 693, 703, 752, 1276	
<code>\c_sys_jobname_str</code>	485, 1169	<code>\tl_if_empty:N</code> TF	755, 1218
<code>\c_sys_timestamp_str ..</code>	5, 1289, 1291	<code>\tl_if_eq:nn</code> TF	86
T			
tex commands:			
<code>\tex_mdffivesum:D</code>	589	<code>\tl_if_in:nn</code> TF	74, 80
text commands:			
<code>\text_declare_purify_equivalent:Nn</code>		<code>\tl_new:N</code>	10, 11,
.....	603, 604		572, 616, 617, 635, 832, 833, 1181, 1278
<code>\text_purify:n</code>	605	<code>\tl_put_right:Nn</code>	646
<code>\texttilde</code>	604	<code>\tl_set:Nn</code>	577, 605, 626,
tl commands:			
<code>\c_space_tl</code>	369, 518, 524		627, 630, 631, 705, 708, 750, 751, 1273
<code>\tl_clear:N</code>	1184	<code>\tl_set_eq:NN</code>	581, 1291
<code>\tl_gput_right:Nn</code>		<code>\tl_to_str:N</code>	603, 606
.....	638, 799, 834, 852, 1222, 1311	<code>\tl_use:N</code>	770, 842
<code>\tl_if_blank:n</code> TF	303, 311,	U	
319, 327, 335, 546, 553, 556, 559,		use commands:	
		<code>\use:N</code>	42
		<code>\use_i:nn</code>	1098
		<code>\use_ii:nn</code>	1100