The l3pdfmeta module
PDF standards
LaTeX PDF management testphase bundle

The LaTeX Project*

Version 0.95r, released 2022-08-24

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.

In future is will probably also contain code to setup XMP-metadata. Until then XMP-metadata can be added by one of two mutual incompatible packages: hyperxmp and pdfx. Both packages aren't yet compatible with the new PDF management, but for hyperxmp some patches are provided, so the basic functions works.

1.1 Verifying requirements of PDF standards

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

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

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

Verifying against a PDF-standard involves two different task:

- 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]

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

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

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 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:nTF` ⋆

`\pdfmeta_standard_verify:n{⟨requirement⟩}`

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 restrictment 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.

`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. The engines do not offer currently an option to suppress the dictionary completly, one can only give the entries the value null (it only works for all entries with lualatex and pdflatex). The next pdflatex will offer `\pdfomitinfodict`. Until then l3pdfmeta does nothing with this requirement.

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 then one entry in the `/OutputIntent` array (which is allowed), their `/DestOutputProfile` should all reference the same color profile[2].

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

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

The interface looks like this

```
\DocumentMetadata
  {
    %other options for example pdfstandard
     colorprofiles=
      {
        A = sRGB.icc, %or a or longer GTS_PDFA1 = sRGB.icc
        X = FOGRA39L_coated.icc, % or x or longer GTS_PDFX
        ISO_PDFE1 = whatever.icc
      }

  }
```

`sRGB.icc` and `FOGRA39L_coated.icc` (from the colorprofiles package are predefined and will work directly[3]. `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.

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

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

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

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

  }
```

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

1.3 Regression tests

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

`\pdfmeta_set_regression_data:` `\pdfmeta_set_regression_data:`

This sets various metadata to values needed by the LaTeX regression tests. It also sets the seed for random functions.

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 code[5], 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.

[5]with 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 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 "grÃ¼ÃŸe". 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\000l\000l\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/.

2.2 User interfaces and differences to **hyperxmp**

2.2.1 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. LaTeX 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.

2.2.2 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.

2.3 Language

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

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

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

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

2.4 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.

2.5 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.

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

2.6 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.

2.7 User commands

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

`\pdfmeta_xmp_add:n`

`\pdfmeta_xmp_add:n{⟨XML⟩}`

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

`\pdfmeta_xmp_xmlns_new:nn`

`\pdfmeta_xmp_xmlns_new:nn{⟨prefix⟩}{⟨uri⟩}`

With this command a xmlns name space can be added.

3 l3pdfmeta implementation

```
⟨@@=pdfmeta⟩
⟨*header⟩
\ProvidesExplPackage{l3pdfmeta}{2022-08-24}{0.95r}
  {PDF-Standards---LaTeX PDF management testphase bundle}
⟨/header⟩
```

Message for unknown standards

```
⟨*package⟩
\msg_new:nnn  {pdf }{unknown-standard}{The~standard~'#1'~is~unknown~and~has~been~ignored}
```

Message for not fitting pdf version

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

`\l__pdfmeta_tmpa_tl`
`\l__pdfmeta_tmpb_tl`
`\l__pdfmeta_tmpa_str`
`\g__pdfmetatmpa_str`
`\l__pdfmeta_tmpa_seq`
`\l__pdfmeta_tmpb_seq`

```
\tl_new:N  \l__pdfmeta_tmpa_tl
\tl_new:N  \l__pdfmeta_tmpb_tl
\str_new:N \l__pdfmeta_tmpa_str
\str_new:N \g__pdfmeta_tmpa_str
\seq_new:N \l__pdfmeta_tmpa_seq
\seq_new:N \l__pdfmeta_tmpb_seq
```

(*End 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`

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

(*End 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

```
\cs_new:Npn \pdfmeta_standard_item:n #1
 {
   \prop_item:Nn \g__pdfmeta_standard_prop {#1}
 }
```

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

\pdfmeta_standard_get:nN

```
\cs_new_protected:Npn \pdfmeta_standard_get:nN #1 #2
 {
   \prop_get:NnN \g__pdfmeta_standard_prop {#1} #2
 }
```

(*End definition for* \pdfmeta_standard_get:nN. *This function is documented on page 2.*)

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

\pdfmeta_standard_verify_p:n
\pdfmeta_standard_verify:n*TF*

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

```
\prg_new_conditional:Npnn \pdfmeta_standard_verify:n #1 {T,F,TF}
  {
     \prop_if_in:NnTF \g__pdfmeta_standard_prop {#1}
       {
         \prg_return_false:
       }
       {
         \prg_return_true:
       }
  }
```

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

\pdfmeta_standard_verify:nn*TF*

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.

```
\prg_new_protected_conditional:Npnn \pdfmeta_standard_verify:nn #1 #2  {T,F,TF}
  {
    \prop_if_in:NnTF \g__pdfmeta_standard_prop {#1}
      {
        \cs_if_exist:cTF {__pdfmeta_standard_verify_handler_#1:nn}
          {
            \exp_args:Nnnx
            \use:c
              {__pdfmeta_standard_verify_handler_#1:nn}
              { #2 }
              { \prop_item:Nn \g__pdfmeta_standard_prop {#1} }
          }
          {
            \prg_return_false:
          }
      }
      {
        \prg_return_true:
      }
   }
```

(*End 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

```
%
\cs_new_protected:Npn \__pdfmeta_standard_verify_handler_min_pdf_version:nn #1 #2
 {
   \pdf_version_compare:NnTF <
     { #2 }
     {\prg_return_false:}
     {\prg_return_true:}
 }
```

(*End definition for* `\__pdfmeta_standard_verify_handler_min_pdf_version:nn`.)

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

_standard_verify_handler_max_pdf_version:nn

```
%
\cs_new_protected:Npn \__pdfmeta_standard_verify_handler_max_pdf_version:nn #1 #2
 {
   \pdf_version_compare:NnTF >
     { #2 }
     {\prg_return_false:}
     {\prg_return_true:}
 }
```

(*End 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

```

\cs_new_protected:Npn \__pdfmeta_standard_verify_handler_named_actions:nn #1 #2
 {
   \tl_if_in:nnTF { #2 }{ #1 }
     {\prg_return_true:}
     {\prg_return_false:}
 }
```

(*End 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

```
\cs_new_protected:Npn \__pdfmeta_standard_verify_handler_annot_action_A:nn #1 #2
 {
   \tl_if_in:nnTF { #2 }{ #1 }
     {\prg_return_true:}
     {\prg_return_false:}
 }
```

(*End definition for* `\__pdfmeta_standard_verify_handler_annot_action_A:nn`.)

This check is probably not needed, but for completeness dard_verify_handler_outputintent_subtype:nn

```
\cs_new_protected:Npn \__pdfmeta_standard_verify_handler_outputintent_subtype:nn #1 #2
 {
   \tl_if_eq:nnTF { #2 }{ #1 }
     {\prg_return_true:}
     {\prg_return_false:}
 }
```

(*End 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:

```
\cs_new_protected:Npn \__pdfmeta_verify_pdfa_annot_flags:
  {
    \bitset_set_true:Nn  \l_pdfannot_F_bitset {Print}
    \bitset_set_false:Nn \l_pdfannot_F_bitset {Hidden}
    \bitset_set_false:Nn \l_pdfannot_F_bitset {Invisible}
    \bitset_set_false:Nn \l_pdfannot_F_bitset {NoView}
    \pdfannot_dict_put:nnn {link/URI}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
    \pdfannot_dict_put:nnn {link/GoTo}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
    \pdfannot_dict_put:nnn {link/GoToR}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
    \pdfannot_dict_put:nnn {link/Launch}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
    \pdfannot_dict_put:nnn {link/Named}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
  }
```

At begin document this should be checked:

```
\hook_gput_code:nnn {begindocument} {pdf}
  {
    \pdfmeta_standard_verify:nF { annot_flags }
     { \__pdfmeta_verify_pdfa_annot_flags: }
    \pdfmeta_standard_verify:nnF { min_pdf_version }
     { \pdf_version: }
     { \msg_warning:nnxxx {pdf}{wrong-pdfversion}
       {\pdf_version:}{low}
       {
        \pdfmeta_standard_item:n{type}
        -
        \pdfmeta_standard_item:n{level}
       }
     }
    \pdfmeta_standard_verify:nnF { max_pdf_version }
     { \pdf_version: }
     { \msg_warning:nnxxx {pdf}{wrong-pdfversion}
       {\pdf_version:}{high}
       {
        \pdfmeta_standard_item:n{type}
        -
        \pdfmeta_standard_item:n{level}
       }
     }
  }
```

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

```
\prop_new:c { g__pdfmeta_standard_pdf/A-1B_prop }
\prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/A-1B_prop }
  {
    ,name             = pdf/A-1B
    ,type             = A
    ,level            = 1
    ,conformance      = B
    ,year             = 2005
    ,min_pdf_version  = 1.4        %minimum
    ,max_pdf_version  = 1.4        %minimum
    ,no_encryption    =
    ,no_external_content =  % no F, FFilter, or FDecodeParms in stream dicts
    ,no_embed_content = % no EF key in filespec, no /Type/EmbeddedFiles
    ,max_string_size  = 65535
    ,max_array_size   = 8191
    ,max_dict_size    = 4095
    ,max_obj_num      = 8388607
    ,max_nest_qQ      = 28
    ,named_actions    = {NextPage, PrevPage, FirstPage, LastPage}
    ,annot_flags      =
    %booleans. Only the existence of the key matter.
    %If the entry is added it means a requirements is there
    %(in most cases "don't use ...")
    %
    %===============
    % Rule 6.1.13-1 CosDocument, isOptionalContentPresent == false
      ,Catalog_no_OCProperties =
    %===============
    % Rule 6.6.1-1: PDAction, S == "GoTo" || S == "GoToR" || S == "Thread"
    %               || S == "URI" || S == "Named" || S == "SubmitForm"
    % means: no /S/Launch, /S/Sound, /S/Movie, /S/ResetForm, /S/ImportData,
    %        /S/JavaScript, /S/Hide
      ,annot_action_A        = {GoTo,GoToR,Thread,URI,Named,SubmitForm}
    %===============
    % Rule 6.6.2-1: PDAnnot, Subtype != "Widget" || AA_size == 0
    % means: no AA dictionary
      ,annot_widget_no_AA      =
    %===============
    % Rule 6.9-2: PDAnnot, Subtype != "Widget" || (A_size == 0 && AA_size == 0)
    % (looks like a tightening of the previous rule)
      ,annot_widget_no_A_AA    =
    %===============
    % Rule 6.9-1 PDAcroForm, NeedAppearances == null || NeedAppearances == false
    ,form_no_NeedAppearances =
    %===============
    %Rule 6.9-3 PDFormField, AA_size == 0
    ,form_no_AA              =
    %===============
    % to be continued https://docs.verapdf.org/validation/pdfa-part1/
    % - Outputintent/colorprofiles requirements
    % an outputintent should be loaded and is unique.
    ,outputintent_A         = {GTS_PDFA1}
    % - no Alternates key in image dictionaries
    % - no OPI, Ref, Subtype2 with PS key in xobjects
    % - Interpolate  = false in images
    % - no TR, TR2 in ExtGstate
  }

%A-2b ==============
\prop_new:c { g__pdfmeta_standard_pdf/A-2B_prop }
\prop_gset_eq:cc
  { g__pdfmeta_standard_pdf/A-2B_prop }
  { g__pdfmeta_standard_pdf/A-1B_prop }
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-2B_prop }{name}{pdf/A-2B}
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-2B_prop }{year}{2011}
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-2B_prop }{level}{2}
% embedding files is allowed (with restrictions)
\prop_gremove:cn
  { g__pdfmeta_standard_pdf/A-2B_prop }
  { embed_content}
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-2B_prop }{max_pdf_version}{1.7}
%A-2u ==============
\prop_new:c { g__pdfmeta_standard_pdf/A-2U_prop }
\prop_gset_eq:cc
  { g__pdfmeta_standard_pdf/A-2U_prop }
  { g__pdfmeta_standard_pdf/A-2B_prop }
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-2U_prop }{name}{pdf/A-2U}
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-2U_prop }{conformance}{U}
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-2U_prop }{unicode}{}

%A-2a ==============
\prop_new:c { g__pdfmeta_standard_pdf/A-2A_prop }
\prop_gset_eq:cc
  { g__pdfmeta_standard_pdf/A-2A_prop }
  { g__pdfmeta_standard_pdf/A-2B_prop }
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-2A_prop }{name}{pdf/A-2A}
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-2A_prop }{conformance}{A}
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-2A_prop }{tagged}{}
```

```
%A-3b ==============
\prop_new:c { g__pdfmeta_standard_pdf/A-3B_prop }
\prop_gset_eq:cc
  { g__pdfmeta_standard_pdf/A-3B_prop }
  { g__pdfmeta_standard_pdf/A-2B_prop }
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-3B_prop }{name}{pdf/A-3B}
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-3B_prop }{year}{2012}
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-3B_prop }{level}{3}
% embedding files is allowed (with restrictions)
\prop_gremove:cn
  { g__pdfmeta_standard_pdf/A-3B_prop }
  { embed_content}
%A-3u ==============
\prop_new:c { g__pdfmeta_standard_pdf/A-3U_prop }
\prop_gset_eq:cc
  { g__pdfmeta_standard_pdf/A-3U_prop }
  { g__pdfmeta_standard_pdf/A-3B_prop }
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-3U_prop }{name}{pdf/A-3U}
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-3U_prop }{conformance}{U}
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-3U_prop }{unicode}{}

%A-3a ==============
\prop_new:c { g__pdfmeta_standard_pdf/A-3A_prop }
\prop_gset_eq:cc
  { g__pdfmeta_standard_pdf/A-3A_prop }
  { g__pdfmeta_standard_pdf/A-3B_prop }
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-3A_prop }{name}{pdf/A-3A}
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-3A_prop }{conformance}{A}
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-3A_prop }{tagged}{}

%A-4 ==============
\prop_new:c { g__pdfmeta_standard_pdf/A-4_prop }
\prop_gset_eq:cc
  { g__pdfmeta_standard_pdf/A-4_prop }
  { g__pdfmeta_standard_pdf/A-3U_prop }
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-4_prop }{name}{pdf/A-4}
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-4_prop }{level}{4}
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-4_prop }{min_pdf_version}{2.0}
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-4_prop }{year}{2020}
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-4_prop }{Trailer_no_Info}{}
```

```
\prop_gremove:cn
  { g__pdfmeta_standard_pdf/A-4_prop }{conformance}
\prop_gremove:cn
  { g__pdfmeta_standard_pdf/A-4_prop }{max_pdf_version}
```

(*End 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:nx {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:nnx {Catalog}{OutputIntents}{\pdf_object_ref_last:}
   ```

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

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

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

(*End definition for* `\g__pdfmeta_outputintents_prop`.)

Some keys to fill the property.

```
\keys_define:nn { document / metadata }
  {
    colorprofiles .code:n =
     {
       \keys_set:nn { document / metadata / colorprofiles }{#1}
     }
  }
\keys_define:nn { document / metadata / colorprofiles }
 {
   ,A .code:n =
     {
       \tl_if_blank:nF {#1}
```

```
          {
            \prop_gput:Nnn \g__pdfmeta_outputintents_prop
             { GTS_PDFA1  } {#1}
          }
      }
   ,a .code:n =
      {
        \tl_if_blank:nF {#1}
          {
            \prop_gput:Nnn \g__pdfmeta_outputintents_prop
              { GTS_PDFA1  } {#1}
          }
      }
   ,X .code:n =
      {
        \tl_if_blank:nF {#1}
          {
             \prop_gput:Nnn \g__pdfmeta_outputintents_prop
              { GTS_PDFX  } {#1}
          }
      }
   ,x .code:n =
      {
        \tl_if_blank:nF {#1}
          {
            \prop_gput:Nnn \g__pdfmeta_outputintents_prop
              { GTS_PDFX  } {#1}
          }
      }
   ,unknown .code:n =
     {
       \tl_if_blank:nF {#1}
          {
           \exp_args:NNo
            \prop_gput:Nnn \g__pdfmeta_outputintents_prop
              { \l_keys_key_str  } {#1}
          }
     }
 }
```

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

```
\pdfdict_new:n   {l_pdfmeta/outputintent}
\pdfdict_put:nnn {l_pdfmeta/outputintent}
  {Type}{/OutputIntent}
\prop_const_from_keyval:cn { c__pdfmeta_colorprofile_sRGB.icc}
  {
    ,OutputConditionIdentifier=IEC~sRGB
    ,Info=IEC~61966-2.1~Default~RGB~colour~space~-~sRGB
    ,RegistryName=http://www.iec.ch
    ,N = 3
  }
\prop_const_from_keyval:cn { c__pdfmeta_colorprofile_FOGRA39L_coated.icc}
  {
    ,OutputConditionIdentifier=FOGRA39L~Coated
    ,Info={Offset~printing,~according~to~ISO~12647-2:2004/Amd~1,~OFCOM,~ %
           paper~type~1~or~2~=~coated~art,~115~g/m2,~tone~value~increase~
           curves~A~(CMY)~and~B~(K)}
    ,RegistryName=http://www.fogra.org
    ,N = 4
  }
```

__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.

```
\cs_new_protected:Npn \__pdfmeta_embed_colorprofile:n #1%#1 file name
  {
    \pdf_object_if_exist:nF { __color_icc_ #1 }
      {
        \pdf_object_new:n  { __color_icc_ #1 }
        \pdf_object_write:nnx { __color_icc_ #1 } { fstream }
         {
           {/N\c_space_tl
             \prop_item:cn{c__pdfmeta_colorprofile_#1}{N}
           }
           {#1}
         }
      }
  }

\cs_new_protected:Npn \__pdfmeta_write_outputintent:nn #1 #2 %#1 file name, #2 subtype
  {
    \group_begin:
     \pdfdict_put:nnx {l_pdfmeta/outputintent}{S}{/\str_convert_pdfname:n{#2}}
     \pdfdict_put:nnx {l_pdfmeta/outputintent}
       {DestOutputProfile}
       {\pdf_object_ref:n{ __color_icc_ #1 }}
     \clist_map_inline:nn { OutputConditionIdentifier, Info, RegistryName }
       {
         \prop_get:cnNT
          { c__pdfmeta_colorprofile_#1}
          { ##1 }
          \l__pdfmeta_tmpa_tl
          {
            \pdf_string_from_unicode:nVN {utf8/string}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_st
            \pdfdict_put:nnx
              {l_pdfmeta/outputintent}{##1}{\l__pdfmeta_tmpa_str}
          }
       }
     \pdf_object_unnamed_write:nx {dict}{\pdfdict_use:n {l_pdfmeta/outputintent} }
     \pdfmanagement_add:nnx {Catalog}{OutputIntents}{\pdf_object_ref_last:}
    \group_end:
  }
```

(*End 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

```
\AddToHook{begindocument/end}
  {
    \pdfmeta_standard_verify:nTF {outputintent_A}
      {
         \prop_map_inline:Nn \g__pdfmeta_outputintents_prop
           {
             \__pdfmeta_embed_colorprofile:n
               {#2}
             \__pdfmeta_write_outputintent:nn
               {#2}
               {#1}
           }
      }
```

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.

```
      {
         \exp_args:NNx
         \prop_if_in:NnF
           \g__pdfmeta_outputintents_prop
           { \pdfmeta_standard_item:n { outputintent_A } }
           {
             \exp_args:NNx
             \prop_gput:Nnn
               \g__pdfmeta_outputintents_prop
               { \pdfmeta_standard_item:n { outputintent_A } }
               { sRGB.icc }
           }
         \exp_args:NNx
         \prop_get:NnN
           \g__pdfmeta_outputintents_prop
           { \pdfmeta_standard_item:n { outputintent_A } }
           \l__pdfmeta_tmpb_tl
         \exp_args:NV \__pdfmeta_embed_colorprofile:n \l__pdfmeta_tmpb_tl
         \prop_map_inline:Nn \g__pdfmeta_outputintents_prop
           {
             \exp_args:NV
             \__pdfmeta_write_outputintent:nn
               \l__pdfmeta_tmpb_tl
               { #1 }
           }
       }
   }
```

3.2 Regression test

This is simply a copy of the backend function.

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

4 XMP-Metadata implementation

\g__pdfmeta_xmp_bool

This boolean decides if the metadata are included

```
\bool_new:N\g__pdfmeta_xmp_bool
\bool_gset_true:N \g__pdfmeta_xmp_bool
```

(*End definition for \g__pdfmeta_xmp_bool.*)

Preset the two fields to avoid problems with standards.

```
\hook_gput_code:nnn{pdfmanagement/add}{pdfmanagement}
  {
   \pdfmanagement_add:nnx {Info}{Producer}{(\c_sys_engine_exec_str-\c_sys_engine_version_str
   \pdfmanagement_add:nnx {Info}{Creator}{(LaTeX)}
  }
```

4.1 New document keys

```
\keys_define:nn { document / metadata }
 {
   _pdfstandard / X-4 .code:n =
    {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-4}},
   _pdfstandard / X-4p .code:n =
    {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-4p}},
   _pdfstandard / X-5g .code:n =
    {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5g}},
   _pdfstandard / X-5n .code:n =
    {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5n}},
   _pdfstandard / X-5pg .code:n =
    {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5pg}},
   _pdfstandard / X-6 .code:n =
    {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6p}},
   _pdfstandard / X-6n .code:n =
    {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6n}},
   _pdfstandard / X-6p .code:n =
    {\AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6p}},
   _pdfstandard / UA-1 .code:n =
    {\AddToDocumentProperties [document]{pdfstandard-UA}{1}},
   xmp  .bool_gset:N = \g__pdfmeta_xmp_bool
 }
```

4.2 Messages

```
\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:

```
\bool_lazy_or:nnTF
  { \sys_if_engine_luatex_p: }
  { \sys_if_engine_xetex_p: }
  {
    \cs_new:Npn \__pdfmeta_xmp_generate_bom:
      { \char_generate:nn {"FEFF}{12} }
  }
  {
    \cs_new:Npn \__pdfmeta_xmp_generate_bom:
      {
        \char_generate:nn {"EF}{12}
        \char_generate:nn {"BB}{12}
        \char_generate:nn {"BF}{12}
      }
  }
```

(*End 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`

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

(*End definition for* `\l__pdfmeta_xmp_indent_int`*.*)

`\__pdfmeta_xmp_indent:`
`\__pdfmeta_xmp_indent:n`
`\__pdfmeta_xmp_incr_indent:`
`\__pdfmeta_xmp_decr_indent:`

```
\cs_new:Npn \__pdfmeta_xmp_indent:
  {
    \iow_newline:
    \prg_replicate:nn {\l__pdfmeta_xmp_indent_int}{\c_space_tl}
  }

\cs_new:Npn \__pdfmeta_xmp_indent:n #1
  {
    \iow_newline:
    \prg_replicate:nn {#1}{\c_space_tl}
  }

\cs_new_protected:Npn \__pdfmeta_xmp_incr_indent:
  {
    \int_incr:N \l__pdfmeta_xmp_indent_int
  }

\cs_new_protected:Npn \__pdfmeta_xmp_decr_indent:
  {
    \int_decr:N \l__pdfmeta_xmp_indent_int
  }
```

(*End 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

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

(*End 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.

```
\cs_new_protected:Npn \__pdfmeta_xmp_date_split:nN #1 #2 %#1 date, #2 seq
  {
    \regex_split:NnN \l__pdfmeta_xmp_date_regex {#1} #2
  }
\cs_generate_variant:Nn \__pdfmeta_xmp_date_split:nN {VN,eN}
```

(*End 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.

```
\cs_new:Npn\__pdfmeta_xmp_print_date:N #1 % seq
  {
    \tl_if_blank:eTF { \seq_item:Nn #1 {1} }
     {
       \seq_item:Nn #1 {2} %year
        -
       \seq_item:Nn #1 {3} %month
        -
       \seq_item:Nn #1 {4} % day
       \tl_if_blank:eF
         { \seq_item:Nn #1 {5} }
         { T \seq_item:Nn #1 {5} } %hour
       \tl_if_blank:eF
         { \seq_item:Nn #1 {6} }
         { : \seq_item:Nn #1 {6} } %minutes
       \tl_if_blank:eF
         { \seq_item:Nn #1 {7} }
         { : \seq_item:Nn #1 {7} } %seconds
       \seq_item:Nn #1 {8}  %Z,+,-
       \seq_item:Nn #1 {9}
       \tl_if_blank:eF
         { \seq_item:Nn #1 {10} }
         { : \seq_item:Nn #1 {10} }
      }
      {
        \seq_item:Nn #1 {1}
      }
  }
```

(*End 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.

```
\tl_new:N  \l__pdfmeta_xmp_currentdate_tl
\seq_new:N \l__pdfmeta_xmp_currentdate_seq
```

(*End 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.

```
\cs_new_protected:Npn \__pdfmeta_xmp_date_get:nNN #1 #2 #3
  %#1 property, #2 tl var with PDF date, #3 seq for splitted date
  {
    \tl_set:Nx #2 { \GetDocumentProperties{#1} }
    \tl_if_blank:VTF #2
      {
        \seq_set_eq:NN #3 \l__pdfmeta_xmp_currentdate_seq
        \tl_set_eq:NN  #2 \l__pdfmeta_xmp_currentdate_tl
      }
      {
        \__pdfmeta_xmp_date_split:VN #2 #3
      }
  }
```

(*End definition for* `\__pdfmeta_xmp_date_get:nNN`.)

4.3.4 UUID

We need a command to generate an uuid

__pdfmeta_xmp_create_uuid:nN

```
\cs_new_protected:Npn \__pdfmeta_xmp_create_uuid:nN #1 #2
  {
    \str_set:Nx#2 {\str_lowercase:f{\tex_mdfivesum:D{#1}}}
    \str_set:Nx#2
      { uuid:
        \str_range:Nnn #2{1}{8}
        -\str_range:Nnn#2{9}{12}
        -4\str_range:Nnn#2{13}{15}
        -8\str_range:Nnn#2{16}{18}
        -\str_range:Nnn#2{19}{30}
      }
  }
```

(*End 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.

```
\cs_new_protected:Npn \__pdfmeta_xmp_sanitize:nN #1 #2
%#1 input string, #2 str with the output
  {
    \group_begin:
     \text_declare_purify_equivalent:Nn \& {\tl_to_str:N & }
     \text_declare_purify_equivalent:Nn \texttilde {\c_tilde_str}
     \tl_set:Nx \l__pdfmeta_tmpa_tl { \text_purify:n {#1} }
     \str_gset:Nx \g__pdfmeta_tmpa_str { \tl_to_str:N \l__pdfmeta_tmpa_tl }
     \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {&}{&amp;}
     \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {<}{&lt;}
     \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {>}{&gt;}
     \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {"}{&quot;}
    \group_end:
```

```
    \str_set_eq:NN #2 \g__pdfmeta_tmpa_str
  }

\cs_generate_variant:Nn\__pdfmeta_xmp_sanitize:nN {VN}
```

(*End definition for* `\__pdfmeta_xmp_sanitize:nN`.)

4.4 Language handling

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

`\l__pdfmeta_xmp_doclang_tl`
`\l__pdfmeta_xmp_metalang_tl`

```
\tl_new:N \l__pdfmeta_xmp_doclang_tl
\tl_new:N \l__pdfmeta_xmp_metalang_tl
```

(*End 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`

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

(*End definition for* `\l__pdfmeta_xmp_lang_regex`.)

```
\cs_new_protected:Npn \__pdfmeta_xmp_lang_get:nNN #1 #2 #3
% #1 text, #2 tl var for lang match (or default), #3 tl var for text
  {
    \regex_extract_once:NnN \l__pdfmeta_xmp_lang_regex {#1}\l__pdfmeta_tmpa_seq
    \seq_if_empty:NTF \l__pdfmeta_tmpa_seq
      {
        \tl_set:Nn #2 \l__pdfmeta_xmp_metalang_tl
        \tl_set:Nn #3 {#1}
      }
      {
        \tl_set:Nx #2 {\seq_item:Nn\l__pdfmeta_tmpa_seq{2}}
        \tl_set:Nx #3 {\seq_item:Nn\l__pdfmeta_tmpa_seq{3}}
      }
  }
\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`

```
\tl_new:N \g__pdfmeta_xmp_packet_tl
```

(*End 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.

```
\cs_new_protected:Npn \__pdfmeta_xmp_add_packet_chunk:n #1
  {
    \tl_gput_right:Nx\g__pdfmeta_xmp_packet_tl
      {
        \__pdfmeta_xmp_indent:  \exp_not:n{#1}
      }
  }
\cs_generate_variant:Nn \__pdfmeta_xmp_add_packet_chunk:n {e}
```

(*End 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.

```
\cs_new_protected:Npn \__pdfmeta_xmp_add_packet_chunk:nN #1 #2
  {
    \tl_put_right:Nx#2
      {
        \__pdfmeta_xmp_indent:  \exp_not:n{#1}
      }
  }
\cs_generate_variant:Nn \__pdfmeta_xmp_add_packet_chunk:nN {eN}
```

(*End definition for __pdfmeta_xmp_add_packet_chunk:nN.*)

__pdfmeta_xmp_add_packet_open:nn

This commands opens a xml structure and increases the indent.

```
\cs_new_protected:Npn \__pdfmeta_xmp_add_packet_open:nn #1 #2 %#1 prefix #2 name
  {
    \__pdfmeta_xmp_add_packet_chunk:n {<#1:#2>}
    \__pdfmeta_xmp_incr_indent:
  }
\cs_generate_variant:Nn \__pdfmeta_xmp_add_packet_open:nn  {ne}
```

(*End 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.

```
\cs_new_protected:Npn \__pdfmeta_xmp_add_packet_open_attr:nnn #1 #2 #3
  %#1 prefix #2 name #3 attr
  {
    \__pdfmeta_xmp_add_packet_chunk:n {<#1:#2~#3>}
    \__pdfmeta_xmp_incr_indent:
  }
\cs_generate_variant:Nn \__pdfmeta_xmp_add_packet_open_attr:nnn  {nne}
```

(*End definition for __pdfmeta_xmp_add_packet_open_attr:nnn.*)

__pdfmeta_xmp_add_packet_close:nn

This closes a structure and decreases the indent.

```
\cs_new_protected:Npn \__pdfmeta_xmp_add_packet_close:nn #1 #2 %#1 prefix #2:name
  {
    \__pdfmeta_xmp_decr_indent:
    \__pdfmeta_xmp_add_packet_chunk:n {</#1:#2>}
  }
```

(*End 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.

```
\cs_new_protected:Npn \__pdfmeta_xmp_add_packet_line:nnn #1 #2 #3
 %#1 prefix #2 name #3 content
  {
    \tl_if_blank:nF {#3}
     {
      \__pdfmeta_xmp_sanitize:nN {#3}\l__pdfmeta_tmpa_str
      \__pdfmeta_xmp_add_packet_chunk:e {<#1:#2>\l__pdfmeta_tmpa_str</#1:#2>}
     }
  }
\cs_generate_variant:Nn \__pdfmeta_xmp_add_packet_line:nnn {nne,nnV,nee}
```

(*End 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.

```
\cs_new_protected:Npn \__pdfmeta_xmp_add_packet_line:nnnN #1 #2 #3 #4
 %#1 prefix #2 name #3 content #4 tl_var to prebuilt.
  {
    \tl_if_blank:nF {#3}
     {
      \__pdfmeta_xmp_sanitize:nN {#3}\l__pdfmeta_tmpa_str
      \__pdfmeta_xmp_add_packet_chunk:eN {<#1:#2>\l__pdfmeta_tmpa_str</#1:#2>} #4
     }
  }
\cs_generate_variant:Nn \__pdfmeta_xmp_add_packet_line:nnnN {nneN}
```

(*End definition for* `\__pdfmeta_xmp_add_packet_line:nnnN`.)

__pdfmeta_xmp_add_packet_line_attr:nnnn A similar command with attribute

```
\cs_new_protected:Npn \__pdfmeta_xmp_add_packet_line_attr:nnnn #1 #2 #3 #4
 %#1 prefix #2 name #3 attribute #4 content
  {
    \tl_if_blank:nF {#4}
     {
      \__pdfmeta_xmp_sanitize:nN {#4}\l__pdfmeta_tmpa_str
      \__pdfmeta_xmp_add_packet_chunk:e {<#1:#2~#3>\l__pdfmeta_tmpa_str</#1:#2>}
     }
  }
\cs_generate_variant:Nn \__pdfmeta_xmp_add_packet_line_attr:nnnn {nnee,nneV}
```

(*End definition for* `\__pdfmeta_xmp_add_packet_line_attr:nnnn`.)

__pdfmeta_xmp_add_packet_line_default:nnnn

```
 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_line_default:nnnn #1 #2 #3 #4
   % #1 prefix #2 name #3 default #4 content
   {
     \tl_if_blank:nTF { #4 }
      {
       \tl_set:Nn  \l__pdfmeta_tmpa_tl  {#3}
      }
      {
        \tl_set:Nn \l__pdfmeta_tmpa_tl  {#4}
      }
     \__pdfmeta_xmp_add_packet_line:nnV {#1}{#2}\l__pdfmeta_tmpa_tl
   }
\cs_generate_variant:Nn \__pdfmeta_xmp_add_packet_line_default:nnnn {nnee}
```

(*End definition for* `\__pdfmeta_xmp_add_packet_line_default:nnnn`.)

Some data are stored as unordered (Bag) or ordered lists (Seq) or (Alt). Here we check also for the language.

```
\cs_new_protected:Npn \__pdfmeta_xmp_add_packet_list:nnnn #1 #2 #3 #4
  %#1 prefix, #2 name,  #3 type (Seq/Bag/Alt) #4 a clist
  {
    \clist_if_empty:nF { #4 }
      {
        \__pdfmeta_xmp_add_packet_open:nn {#1}{#2}
         \__pdfmeta_xmp_add_packet_open:nn {rdf}{#3}
          \clist_map_inline:nn {#4}
            {
              \__pdfmeta_xmp_lang_get:nNN {##1}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl
              \__pdfmeta_xmp_add_packet_line_attr:nneV
               {rdf}{li}{xml:lang="\l__pdfmeta_tmpa_tl" }\l__pdfmeta_tmpb_tl
            }
         \__pdfmeta_xmp_add_packet_close:nn{rdf}{#3}
        \__pdfmeta_xmp_add_packet_close:nn {#1}{#2}
      }
   }
\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.

```
\cs_new_protected:Npn \__pdfmeta_xmp_build_packet:
  {
```

Get the main languages

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

```
   \__pdfmeta_xmp_build_iptc_data:N \l__pdfmeta_xmp_iptc_data_tl
   \tl_if_empty:NT \l__pdfmeta_xmp_iptc_data_tl
     {
       \seq_remove_all:Nn \l__pdfmeta_xmp_schema_seq { Iptc4xmpCore }
     }
```

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

```
     \__pdfmeta_xmp_add_packet_chunk:e
      {<?xpacket~begin="\__pdfmeta_xmp_generate_bom:"~id="W5M0MpCehiHzreSzNTczkc9d"?>}
```

```
    \__pdfmeta_xmp_add_packet_open:nn{x}{xmpmeta~xmlns:x="adobe:ns:meta/"}
     \__pdfmeta_xmp_add_packet_open:ne{rdf}
       {RDF~xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\c_hash_str"}
```

The rdf namespaces

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

The extensions

```
        \__pdfmeta_xmp_add_packet_open:nn{pdfaExtension}{schemas}
         \__pdfmeta_xmp_add_packet_open:nn {rdf}{Bag}
           \seq_map_inline:Nn \l__pdfmeta_xmp_schema_seq
             {
               \tl_use:c { g__pdfmeta_xmp_schema_##1_tl }
             }
         \__pdfmeta_xmp_add_packet_close:nn {rdf}{Bag}
        \__pdfmeta_xmp_add_packet_close:nn {pdfaExtension}{schemas}
```

Now starts the part with the data.

```
    % data
        \__pdfmeta_xmp_build_pdf:
        \__pdfmeta_xmp_build_xmpRights:
        \__pdfmeta_xmp_build_standards: %pdfaid,pdfxid,pdfuaid
        \__pdfmeta_xmp_build_dc:
        \__pdfmeta_xmp_build_photoshop:
        \__pdfmeta_xmp_build_xmp:
        \__pdfmeta_xmp_build_xmpMM:
        \__pdfmeta_xmp_build_prism:
        \__pdfmeta_xmp_build_iptc:
        \__pdfmeta_xmp_build_user: %user additions
    % end
      \__pdfmeta_xmp_add_packet_close:nn {rdf}{Description}
     \__pdfmeta_xmp_add_packet_close:nn {rdf}{RDF}
    \__pdfmeta_xmp_add_packet_close:nn {x}{xmpmeta}
    \int_set:Nn  \l__pdfmeta_xmp_indent_int{20}
    \prg_replicate:nn{10}{\__pdfmeta_xmp_add_packet_chunk:n {}}
    \int_zero:N \l__pdfmeta_xmp_indent_int
    \__pdfmeta_xmp_add_packet_chunk:n {<?xpacket~end="w"?>}
}
```

(*End 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.

```
\str_new:N  \g__pdfmeta_xmp_xmlns_tl
\prop_new:N \g__pdfmeta_xmp_xmlns_prop
```

(*End definition for* `\g__pdfmeta_xmp_xmlns_tl` *and* `\g__pdfmeta_xmp_xmlns_prop`.)

__pdfmeta_xmp_xmlns_new:nn
__pdfmeta_xmp_xmlns_new:nx

```
\cs_new_protected:Npn \__pdfmeta_xmp_xmlns_new:nn #1 #2
  {
    \prop_gput:Nnn \g__pdfmeta_xmp_xmlns_prop {#1}{#2}
    \tl_gput_right:Nx \g__pdfmeta_xmp_xmlns_tl
      {
        \__pdfmeta_xmp_indent:n{4} xmlns:\exp_not:n{#1="#2"}
      }
  }
\cs_generate_variant:Nn \__pdfmeta_xmp_xmlns_new:nn {nx}
```

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

Now we fill the data. The list is more or less the same as in hyperxmp

```
\__pdfmeta_xmp_xmlns_new:nn {pdf}      {http://ns.adobe.com/pdf/1.3/}
\__pdfmeta_xmp_xmlns_new:nn {xmpRights}{http://ns.adobe.com/xap/1.0/rights/}
\__pdfmeta_xmp_xmlns_new:nn {dc}       {http://purl.org/dc/elements/1.1/}
\__pdfmeta_xmp_xmlns_new:nn {photoshop}{http://ns.adobe.com/photoshop/1.0/}
\__pdfmeta_xmp_xmlns_new:nn {xmp}      {http://ns.adobe.com/xap/1.0/}
\__pdfmeta_xmp_xmlns_new:nn {xmpMM}    {http://ns.adobe.com/xap/1.0/mm/}
\__pdfmeta_xmp_xmlns_new:nx {stEvt}
  {http://ns.adobe.com/xap/1.0/sType/ResourceEvent\c_hash_str}
\__pdfmeta_xmp_xmlns_new:nn {pdfaid}   {http://www.aiim.org/pdfa/ns/id/}
\__pdfmeta_xmp_xmlns_new:nn {pdfuaid}  {http://www.aiim.org/pdfua/ns/id/}
\__pdfmeta_xmp_xmlns_new:nn {pdfx}     {http://ns.adobe.com/pdfx/1.3/}
\__pdfmeta_xmp_xmlns_new:nn {pdfxid}   {http://www.npes.org/pdfx/ns/id/}
\__pdfmeta_xmp_xmlns_new:nn {prism}    {http://prismstandard.org/namespaces/basic/3.0/}
%\__pdfmeta_xmp_xmlns_new:nn {jav}      {http://www.niso.org/schemas/jav/1.0/}
%\__pdfmeta_xmp_xmlns_new:nn {xmpTPg}   {http://ns.adobe.com/xap/1.0/t/pg/}
\__pdfmeta_xmp_xmlns_new:nx {stFnt}    {http://ns.adobe.com/xap/1.0/sType/Font\c_hash_str}
\__pdfmeta_xmp_xmlns_new:nn {Iptc4xmpCore}{http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
\__pdfmeta_xmp_xmlns_new:nn {pdfaExtension}{http://www.aiim.org/pdfa/ns/extension/}
\__pdfmeta_xmp_xmlns_new:nx {pdfaSchema}{http://www.aiim.org/pdfa/ns/schema\c_hash_str}
\__pdfmeta_xmp_xmlns_new:nx {pdfaProperty}{http://www.aiim.org/pdfa/ns/property\c_hash_str}
\__pdfmeta_xmp_xmlns_new:nx {pdfaType} {http://www.aiim.org/pdfa/ns/type\c_hash_str}
\__pdfmeta_xmp_xmlns_new:nx {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

```
\seq_new:N \l__pdfmeta_xmp_schema_seq
```

(*End 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.

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

```
%#1 name #2 prefix, #3 text
{
  \seq_put_right:Nn \l__pdfmeta_xmp_schema_seq { #2 }
  \tl_new:c { g__pdfmeta_xmp_schema_#2_tl }
  \tl_new:c { g__pdfmeta_xmp_schema_#2_properties_tl }
  \tl_gput_right:cn { g__pdfmeta_xmp_schema_#2_tl }
    {
      \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
       \__pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{schema}{#1}
       \__pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{prefix}{#2}
       \__pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{namespaceURI}{#3}
       \__pdfmeta_xmp_add_packet_open:nn {pdfaSchema}{property}
        \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
            \tl_use:c { g__pdfmeta_xmp_schema_#2_properties_tl }
        \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
       \__pdfmeta_xmp_add_packet_close:nn {pdfaSchema}{property}
      \cs_if_exist_use:c {__pdfmeta_xmp_schema_#2_additions:}
      \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
    }
}
```

(*End definition for* `\__pdfmeta_xmp_schema_new:nnn`.)

`\__pdfmeta_xmp_property_new:nnn`

This adds a property to a schema.

```
\cs_new_protected:Npn \__pdfmeta_xmp_property_new:nnnnn #1 #2 #3 #4 #5 %
    %#1 schema #2 name, #3 type, #4 category #5 description
  {
    \tl_gput_right:cn { g__pdfmeta_xmp_schema_#1_properties_tl }
      {
        \__pdfmeta_xmp_add_packet_open:nn {rdf}{li~rdf:parseType="Resource"}
          \__pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{name}{#2}
          \__pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{valueType}{#3}
          \__pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{category}{#4}
          \__pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{description}{#5}
        \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
     }
  }
```

(*End definition for* `\__pdfmeta_xmp_property_new:nnn`.)

`\__pdfmeta_xmp_add_packet_field:nnn`

This adds a field to a schema.

```
\cs_new_protected:Npn \__pdfmeta_xmp_add_packet_field:nnn #1 #2 #3 %
  %#1 name #2 valuetype #3 description
  {
    \__pdfmeta_xmp_add_packet_open_attr:nnn {rdf}{li}{rdf:parseType="Resource"}
          \__pdfmeta_xmp_add_packet_line:nnn {pdfaField}{name}{#1}
          \__pdfmeta_xmp_add_packet_line:nnn {pdfaField}{valueType}{#2}
          \__pdfmeta_xmp_add_packet_line:nnn {pdfaField}{description}{#3}
    \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
  }
```

(*End definition for* `\__pdfmeta_xmp_add_packet_field:nnn`.)

4.7.1 The extension data

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

[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. We ignore OriginalDocumentID until requested.

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)`

```
    \__pdfmeta_xmp_schema_new:nnn
      {PDF/A~Identification~Schema}
      {pdfaid}
      {http://www.aiim.org/pdfa/ns/id/}
    \__pdfmeta_xmp_property_new:nnnnn
      {pdfaid}
      {year}
      {Integer}
      {internal}
      {Year~of~standard}
```

(*End definition for* `pdfaid~(schema)`. *This function is documented on page* **??**.)

pdfuaid here we need to declare the property "part".

`pdfuaid~(schema)`

```
    \__pdfmeta_xmp_schema_new:nnn
      {PDF/UA~Universal~Accessibility~Schema}
      {pdfuaid}
      {http://www.aiim.org/pdfua/ns/id/}
    \__pdfmeta_xmp_property_new:nnnnn
      {pdfuaid}
      {part}
      {Integer}
      {internal}
      {Part~of~ISO~14289~standard}
```

(*End definition for* `pdfuaid~(schema)`. *This function is documented on page* **??**.)

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

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)`

```
    \__pdfmeta_xmp_schema_new:nnn
        {PDF/X~ID~Schema}
        {pdfxid}
        {http://www.npes.org/pdfx/ns/id/}
    \__pdfmeta_xmp_property_new:nnnnn
        {pdfxid}
        {GTS_PDFXVersion}
        {Text}
        {internal}
        {ID~of~PDF/X~standard}
```

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

`prism~(schema)` **Prism**

```
    \__pdfmeta_xmp_schema_new:nnn
      {PRISM~Basic~Metadata}
      {prism}
      {http://prismstandard.org/namespaces/basic/3.0/}
    \__pdfmeta_xmp_property_new:nnnnn
      {prism}
      {complianceProfile}
      {Text}
      {internal}
      {PRISM~specification~compliance~profile~to~which~this~document~adheres}
    \__pdfmeta_xmp_property_new:nnnnn
      {prism}
      {publicationName}
      {Text}
      {external}
      {Publication name}
    \__pdfmeta_xmp_property_new:nnnnn
      {prism}
      {aggregationType}
      {Text}
      {external}
      {Publication type}
    \__pdfmeta_xmp_property_new:nnnnn
      {prism}
      {bookEdition}
      {Text}
      {external}
      {Edition~of~the~book~in~which~the~document~was~published}
    \__pdfmeta_xmp_property_new:nnnnn
      {prism}
    {volume}
    {Text}
    {external}
    {Publication~volume~number}
  \__pdfmeta_xmp_property_new:nnnnn
    {prism}
    {number}
    {Text}
    {external}
    {Publication~issue~number~within~a~volume}
  \__pdfmeta_xmp_property_new:nnnnn
    {prism}
    {pageRange}
    {Text}
    {external}
    {Page~range~for~the~document~within~the~print~version~of~its~publication}
  \__pdfmeta_xmp_property_new:nnnnn
    {prism}
    {issn}
    {Text}
    {external}
    {ISSN~for~the~printed~publication~in~which~the~document~was~published}
  \__pdfmeta_xmp_property_new:nnnnn
    {prism}
    {eIssn}
    {Text}
    {external}
    {ISSN~for~the~electronic~publication~in~which~the~document~was~published}
  \__pdfmeta_xmp_property_new:nnnnn
    {prism}
    {isbn}
    {Text}
    {external}
    {ISBN for the publication in which the document was published}
  \__pdfmeta_xmp_property_new:nnnnn
    {prism}
    {doi}
    {Text}
    {external}
    {Digital~Object~Identifier~for~the~document}
  \__pdfmeta_xmp_property_new:nnnnn
    {prism}
    {url}
    {URL}
    {external}
    {URL~at~which~the~document~can~be~found}
  \__pdfmeta_xmp_property_new:nnnnn
    {prism}
    {byteCount}
    {Integer}
    {internal}
    {Approximate~file~size~in~octets}
  \__pdfmeta_xmp_property_new:nnnnn
    {prism}
```

```
      {pageCount}
      {Integer}
      {internal}
      {Number~of~pages~in~the~print~version~of~the~document}
    \__pdfmeta_xmp_property_new:nnnnn
      {prism}
      {subtitle}
      {Text}
      {external}
      {Document’s subtitle}
```

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

iptc

```
    \__pdfmeta_xmp_schema_new:nnn
      {IPTC~Core~Schema}
      {Iptc4xmpCore}
      {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
    \__pdfmeta_xmp_property_new:nnnnn
      {Iptc4xmpCore}
      {CreatorContactInfo}
      {ContactInfo}
      {external}
      {Document~creator’s~contact~information}
    \cs_new_protected:cpn { __pdfmeta_xmp_schema_Iptc4xmpCore_additions: }
      {
        \__pdfmeta_xmp_add_packet_open:nn{pdfaSchema}{valueType}
          \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
            \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
              \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{ContactInfo}
              \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
                 {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
              \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{Iptc4xmpCore}
              \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
                {Basic~set~of~information~to~get~in~contact~with~a~person}
              \__pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
               \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
                \__pdfmeta_xmp_add_packet_field:nnn{CiAdrCity}{Text}
                  {Contact~information~city}
                \__pdfmeta_xmp_add_packet_field:nnn{CiAdrCtry}{Text}
                  {Contact~information~country}
                \__pdfmeta_xmp_add_packet_field:nnn{CiAdrExtadr}{Text}
                  {Contact~information~address}
                \__pdfmeta_xmp_add_packet_field:nnn{CiAdrPcode}{Text}
                  {Contact~information~local~postal~code}
                \__pdfmeta_xmp_add_packet_field:nnn{CiAdrRegion}{Text}
                  {Contact~information~regional~information~such~as~state~or~province}
                \__pdfmeta_xmp_add_packet_field:nnn{CiEmailWork}{Text}
                  {Contact~information~email~address(es)}
                \__pdfmeta_xmp_add_packet_field:nnn{CiTelWork}{Text}
                  {Contact~information~telephone~number(s)}
                \__pdfmeta_xmp_add_packet_field:nnn{CiUrlWork}{Text}
                  {Contact~information~Web~URL(s)}
               \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
              \__pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
             \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
           \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
         \__pdfmeta_xmp_add_packet_close:nn{pdfaSchema}{valueType}
       }
```

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
PDFversion

```
\cs_new_protected:Npn \__pdfmeta_xmp_build_pdf:
  {
```

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

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

Now the PDF version

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

(*End 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

```
\cs_new_protected:Npn \__pdfmeta_xmp_build_xmp:
  {
```

The creator

```
  \__pdfmeta_xmp_add_packet_line_default:nnee
    {xmp}{CreatorTool}
    {LaTeX}
    { \GetDocumentProperties{hyperref/pdfcreator} }
```

The baseurl

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

CreationDate

```
    \__pdfmeta_xmp_date_get:nNN
      {document/pdfcreationdate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
    \__pdfmeta_xmp_add_packet_line:nne{xmp}{CreateDate}{\__pdfmeta_xmp_print_date:N\l__pdfme
    \pdfmanagement_add:nnx{Info}{CreationDate}{(\l__pdfmeta_tmpa_tl)}
```

ModifyDate

```
    \__pdfmeta_xmp_date_get:nNN
      {document/pdfmoddate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
    \__pdfmeta_xmp_add_packet_line:nne{xmp}{ModifyDate}{\__pdfmeta_xmp_print_date:N\l__pdfme
    \pdfmanagement_add:nnx{Info}{ModDate}{(\l__pdfmeta_tmpa_tl)}
```

MetadataDate

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

  }
```

(*End 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:

```
\cs_new_protected:Npn \__pdfmeta_xmp_build_standards:
  {
    \__pdfmeta_xmp_add_packet_line:nne {pdfaid}{part}{\pdfmeta_standard_item:n{level}}
    \__pdfmeta_xmp_add_packet_line:nne
      {pdfaid}{conformance}{\pdfmeta_standard_item:n{conformance}}
    \__pdfmeta_xmp_add_packet_line:nne {pdfaid}{year} {\pdfmeta_standard_item:n{year}}
    \__pdfmeta_xmp_add_packet_line:nne
      {pdfxid}{GTS_PDFXVersion}{\GetDocumentProperties{document/pdfstandard-X}}
    \__pdfmeta_xmp_add_packet_line:nne
      {pdfuaid}{part}{\GetDocumentProperties{document/pdfstandard-UA}}
  }
```

(*End definition for* `\__pdfmeta_xmp_build_standards:`.)

4.8.4 Photoshop

__pdfmeta_xmp_build_photoshop:

```
\cs_new_protected:Npn \__pdfmeta_xmp_build_photoshop:
  {
```

pdfauthortitle/photoshop:AuthorsPosition

```
   \__pdfmeta_xmp_add_packet_line:nne{photoshop}{AuthorsPosition}
     { \GetDocumentProperties{hyperref/pdfauthortitle} }
```

pdfcaptionwriter/photoshop:CaptionWriter

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

  }
```

(*End definition for* `\__pdfmeta_xmp_build_photoshop:`.)

4.9 XMP Media Management

__pdfmeta_xmp_build_xmpMM:

```
\cs_new_protected:Npn \__pdfmeta_xmp_build_xmpMM:
  {
```

pdfdocumentid / xmpMM:DocumentID

```
    \str_set:Nx\l__pdfmeta_tmpa_str {\GetDocumentProperties{hyperref/pdfdocumentid}}
    \str_if_empty:NT \l__pdfmeta_tmpa_str
      {
        \__pdfmeta_xmp_create_uuid:nN
          {\jobname\GetDocumentProperties{hyperref/pdftitle}}
          \l__pdfmeta_tmpa_str
      }
    \__pdfmeta_xmp_add_packet_line:nnV{xmpMM}{DocumentID}
      \l__pdfmeta_tmpa_str
```

pdfinstanceid / xmpMM:InstanceID

```
    \str_set:Nx\l__pdfmeta_tmpa_str {\GetDocumentProperties{hyperref/pdfinstanceid}}
    \str_if_empty:NT \l__pdfmeta_tmpa_str
      {
        \__pdfmeta_xmp_create_uuid:nN
          {\jobname\l__pdfmeta_xmp_currentdate_tl}
          \l__pdfmeta_tmpa_str
      }
    \__pdfmeta_xmp_add_packet_line:nnV{xmpMM}{InstanceID}
      \l__pdfmeta_tmpa_str
```

pdfversionid/xmpMM:VersionID

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

pdfrendition/xmpMM:RenditionClass

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

(*End definition for* `\__pdfmeta_xmp_build_xmpMM:`.)

4.10 Rest of dublin Core data

__pdfmeta_xmp_build_dc:
dc:creator/pdfauthor
dc:subject/pdfkeywords
dc:type/pdftype
dc:publisher/pdfpublisher
dc:description/pdfsubject
dc:language/lang/pdflang
dc:identifier/pdfidentifier
photoshop:AuthorsPosition/pdfauthortitle
photoshop:CaptionWriter/pdfcaptionwriter

```
\cs_new_protected:Npn \__pdfmeta_xmp_build_dc:
  {
```

pdfauthor/dc:creator

```
    \__pdfmeta_xmp_add_packet_list:nnne {dc}{creator}{Seq}
       { \GetDocumentProperties{hyperref/pdfauthor} }
    \int_compare:nNnT {0\pdfmeta_standard_item:n{level}}={1}
       { \pdfmanagement_remove:nn{Info}{Author} }
```

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

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

pdfkeywords/dc:subject

```
    \__pdfmeta_xmp_add_packet_list:nnne {dc}{subject}{Bag}
      { \GetDocumentProperties{hyperref/pdfkeywords} }
    \int_compare:nNnT {0\pdfmeta_standard_item:n{level}}={1}
      { \pdfmanagement_remove:nn{Info}{Keywords} }
```

pdftype/dc:type

```
    \tl_if_blank:eTF { \GetDocumentProperties{hyperref/pdftype} }
      {
        \__pdfmeta_xmp_add_packet_list:nnne {dc}{type}{Bag}{Text}
      }
      {
       \__pdfmeta_xmp_add_packet_list:nnne {dc}{type}{Bag}
         { { \GetDocumentProperties{hyperref/pdftype} }}
      }
```

pdfpublisher/dc:publisher

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

pdfsubject/dc:description

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

lang/pdflang/dc:language

```
   \__pdfmeta_xmp_add_packet_list:nnne {dc}{language}{Bag}
     { \l__pdfmeta_xmp_doclang_tl }
```

pdfidentifier/dc:identifier

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

pdfdate/dc:date

```
   \__pdfmeta_xmp_date_get:nNN {hyperref/pdfdate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
    \__pdfmeta_xmp_add_packet_list:nnne {dc}{date}{Seq}
          {\__pdfmeta_xmp_print_date:N\l__pdfmeta_tmpa_seq}
```

The file format

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

The source

```
    \__pdfmeta_xmp_add_packet_line_default:nnee
     {dc}{source}
     { \c_sys_jobname_str.tex }
     { \GetDocumentProperties{hyperref/pdfsource} }

    \__pdfmeta_xmp_add_packet_list:nnne{dc}{rights}{Alt}
     {\GetDocumentProperties{hyperref/pdfcopyright}}

  }
```

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

4.11 xmpRights

__pdfmeta_xmp_build_xmpRights:

```
\cs_new_protected:Npn \__pdfmeta_xmp_build_xmpRights:
  {
    \__pdfmeta_xmp_add_packet_line:nne
      {xmpRights}
      {WebStatement}
      {\GetDocumentProperties{hyperref/pdflicenseurl}}
  }
```

(*End 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

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

(*End definition for* `\l__pdfmeta_xmp_iptc_data_tl`.)

__pdfmeta_xmp_build_iptc_data:N

```
\cs_new_protected:Npn \__pdfmeta_xmp_build_iptc_data:N #1
  {
     \tl_clear:N #1
     \__pdfmeta_xmp_incr_indent:\__pdfmeta_xmp_incr_indent:\__pdfmeta_xmp_incr_indent:\__pdf
     \__pdfmeta_xmp_add_packet_line:nneN
       {Iptc4xmpCore}{CiAdrExtadr}
       {\GetDocumentProperties{hyperref/pdfcontactaddress}}
       #1
     \__pdfmeta_xmp_add_packet_line:nneN
       {Iptc4xmpCore}{CiAdrCity}
       {\GetDocumentProperties{hyperref/pdfcontactcity}}
       #1
     \__pdfmeta_xmp_add_packet_line:nneN
       {Iptc4xmpCore}{CiAdrPcode}
       {\GetDocumentProperties{hyperref/pdfcontactpostcode}}
       #1
     \__pdfmeta_xmp_add_packet_line:nneN
       {Iptc4xmpCore}{CiAdrCtry}
       {\GetDocumentProperties{hyperref/pdfcontactcountry}}
       #1
     \__pdfmeta_xmp_add_packet_line:nneN
       {Iptc4xmpCore}{CiTelWork}
       {\GetDocumentProperties{hyperref/pdfcontactphone}}
       #1
     \__pdfmeta_xmp_add_packet_line:nneN
       {Iptc4xmpCore}{CiEmailWork}
       {\GetDocumentProperties{hyperref/pdfcontactemail}}
       #1
     \__pdfmeta_xmp_add_packet_line:nneN
       {Iptc4xmpCore}{CiUrlWork}
       {\GetDocumentProperties{hyperref/pdfcontacturl}}
```

```
        #1
      \__pdfmeta_xmp_decr_indent:\__pdfmeta_xmp_decr_indent:\__pdfmeta_xmp_decr_indent:\__pdf
  }
```

(*End definition for* `\__pdfmeta_xmp_build_iptc_data:N`*.*)

`\__pdfmeta_xmp_build_iptc:`

```
\cs_new_protected:Npn \__pdfmeta_xmp_build_iptc:
  {
    \tl_if_empty:NF\l__pdfmeta_xmp_iptc_data_tl
     {
       \__pdfmeta_xmp_add_packet_open_attr:nnn
        {Iptc4xmpCore}{CreatorContactInfo}{rdf:parseType="Resource"}
       \tl_gput_right:Nx\g__pdfmeta_xmp_packet_tl { \l__pdfmeta_xmp_iptc_data_tl }
       \__pdfmeta_xmp_add_packet_close:nn
        {Iptc4xmpCore}{CreatorContactInfo}
    }
  }
```

(*End definition for* `\__pdfmeta_xmp_build_iptc:`*.*)

4.13 Prism

`\__pdfmeta_xmp_build_prism:`
`complianceProfile`
`prism:subtitle/pdfsubtitle`

```
\cs_new_protected:Npn \__pdfmeta_xmp_build_prism:
  {
```

The compliance profile is a fix value taken from hyperxmp

```
    \__pdfmeta_xmp_add_packet_line:nnn
      {prism}{complianceProfile}
      {three}
```

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

```
    \__pdfmeta_xmp_lang_get:eNN
     {\GetDocumentProperties{hyperref/pdfsubtitle}}
     \l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl
    \__pdfmeta_xmp_add_packet_line_attr:nneV
      {prism}{subtitle}
      {xml:lang="\l__pdfmeta_tmpa_tl"}
      \l__pdfmeta_tmpb_tl
```

Then publicationName

```
    \__pdfmeta_xmp_lang_get:eNN
     {\GetDocumentProperties{hyperref/pdfpublication}}
     \l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl
    \__pdfmeta_xmp_add_packet_line_attr:nneV
      {prism}{publicationName}
      {xml:lang="\l__pdfmeta_tmpa_tl"}
      \l__pdfmeta_tmpb_tl
```

Now the rest

```
    \__pdfmeta_xmp_add_packet_line:nne
      {prism}{bookEdition}
      {\GetDocumentProperties{hyperref/pdfbookedition}}
    \__pdfmeta_xmp_add_packet_line:nne
      {prism}{aggregationType}
      {\GetDocumentProperties{hyperref/pdfpubtype}}
    \__pdfmeta_xmp_add_packet_line:nne
      {prism}{volume}
      {\GetDocumentProperties{hyperref/pdfvolumenum}}
    \__pdfmeta_xmp_add_packet_line:nne
      {prism}{number}
      {\GetDocumentProperties{hyperref/pdfissuenum}}
    \__pdfmeta_xmp_add_packet_line:nne
      {prism}{pageRange}
      {\GetDocumentProperties{hyperref/pdfpagerange}}
    \__pdfmeta_xmp_add_packet_line:nne
      {prism}{issn}
      {\GetDocumentProperties{hyperref/pdfissn}}
    \__pdfmeta_xmp_add_packet_line:nne
      {prism}{eIssn}
      {\GetDocumentProperties{hyperref/pdfeissn}}
    \__pdfmeta_xmp_add_packet_line:nne
      {prism}{doi}
      {\GetDocumentProperties{hyperref/pdfdoi}}
    \__pdfmeta_xmp_add_packet_line:nne
      {prism}{url}
      {\GetDocumentProperties{hyperref/pdfurl}}
```

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

```
     \tl_set:Nx \l__pdfmeta_tmpa_tl { \GetDocumentProperties{hyperref/pdfnumpages} }
     \__pdfmeta_xmp_add_packet_line:nne
      {prism}{pageCount}
      {\tl_if_blank:VTF \l__pdfmeta_tmpa_tl {\PreviousTotalPages}{\l__pdfmeta_tmpa_tl}}
  }
```

(*End 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

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

(*End definition for* `\g__pdfmeta_xmp_user_packet_str`.)

__pdfmeta_xmp_build_user:

```
\cs_new_protected:Npn \__pdfmeta_xmp_build_user:
 {
   \int_zero:N \l__pdfmeta_xmp_indent_int
   \g__pdfmeta_xmp_user_packet_tl
   \int_set:Nn \l__pdfmeta_xmp_indent_int {3}
 }
```

(*End 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

```
\AddToHook{shipout/lastpage}
  {
    \bool_if:NT\g__pdfmeta_xmp_bool
     {
       \file_get_timestamp:nN{\jobname.log}\l__pdfmeta_xmp_currentdate_tl
       \__pdfmeta_xmp_date_split:VN\l__pdfmeta_xmp_currentdate_tl\l__pdfmeta_xmp_currentdate
       \__pdfmeta_xmp_build_packet:
       \exp_args:No
       \__pdf_backend_metadata_stream:n {\g__pdfmeta_xmp_packet_tl}
        \pdfmanagement_add:nnx {Catalog} {Metadata}{\pdf_object_ref_last:}
     }
  }
```

4.15 User commands

`\pdfmeta_xmp_add:n`

```
\cs_new_protected:Npn \pdfmeta_xmp_add:n #1
  {
    \tl_gput_right:Nn \g__pdfmeta_xmp_user_packet_tl
      {
        \__pdfmeta_xmp_add_packet_chunk:n { #1 }
      }
  }
```

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

`\pdfmeta_xmp_xmlns_new:nn`

```
\cs_new_protected:Npn \pdfmeta_xmp_xmlns_new:nn #1 #2
  {
    \prop_if_in:NnTF \g__pdfmeta_xmp_xmlns_prop {#1}
      {\msg_warning:nnn{pdfmeta}{namespace-defined}{#1}}
      {\__pdfmeta_xmp_xmlns_new:nn {#1}{#2}}
  }
```

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

```
⟨/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.