

The `l3pdfmeta` module

PDF standards

LaTeX PDF management bundle

The LaTeX Project*

Version 0.96z, released 2026-04-15

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 Standard families

A PDF can claim that it complies to more than one standard but always only to one version of a specific family. For LaTeX relevant are the A-, UA- and X-standards.

The `l3pdfmeta` module started with support for the A-standards, some basic support for UA- and X-requirements were added in a rather ad-hoc way because there isn't a easy way to merge requirements and because a merge loses the option to report for which standard a requirement failed: if e.g. A-4 requires PDF 2.0 and UA-1 PDF 1.7 it is not clear what a merge should do.

So instead of merging the requirements the code now keeps track of the requirements of standard *families* (currently A, UA and X) and offers a command to switch the family to enable validation of their requirements. The default family is always the A-family – it has the largest numbers of requirements and also the largest numbers of requirements that the code can actually check.

1.2 Verifying requirements of PDF standards

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

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

This is work in progress and more tests will be added. But it should be noted that it will probably never be possible to prevent all forbidden actions or enforce all required

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

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

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_family:nn` `\pdfmeta_standard_family:nn` $\langle family \rangle$ $\langle code \rangle$

This command allows to change the family for which requirements should be checked. $\langle family \rangle$ should be one of A, X or UA. The function switches to the requirements of this family, then executes $\langle code \rangle$ and then switches back to the A-standard family (this means that the command is not needed for a test of an A-standard requirement as that is the default anyway). $\langle code \rangle$ can do testing or retrieve values or other things but it shouldn't contain another `\pdfmeta_standard_family:nn`.

`\pdfmeta_standard_verify:nTF` \star `\pdfmeta_standard_verify:nTF` $\langle requirement \rangle$ $\langle true code \rangle$ $\langle false code \rangle$

This checks if $\langle requirement \rangle$ 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:nnTF` $\langle requirement \rangle$ $\langle value \rangle$ $\langle true code \rangle$ $\langle false code \rangle$

This checks if $\langle requirement \rangle$ is listed in the standard, if yes it tries to find a pre-defined test handler for the requirement and passes $\langle value \rangle$ and the value recorded in the standard to it. The handler returns `FALSE` if some special action is needed (e.g. if $\langle value \rangle$ 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` \star `\pdfmeta_standard_item:n` $\langle requirement \rangle$

This retrieves the value of $\langle requirement \rangle$ 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.

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

`\pdfmeta_standard_get:nN` `\pdfmeta_standard_get:nN` $\langle requirement \rangle$ $\langle t1 var \rangle$

This retrieves the value of $\langle requirement \rangle$ and stores it in the $\langle t1 var \rangle$. If the $\langle requirement \rangle$ is not found the special value `\q_no_value` is used. The $\langle t1 var \rangle$ 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/>

1.2.1 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` or `\SetKeys[document/metadata]` is used, see below.*

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

`no_encryption` don't encrypt

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

`no_embed_content` no `/EF` key in filespec, no `/Type/EmbeddedFiles`. *This will be checked in future by l3pdfmeta for the files it embeds.* The restriction is set only for PDF/A-1 versions. PDF/A-2 and PDF/A-3 lifted this restriction: PDF/A-2 allows to embed other PDF documents conforming to either PDF/A-1 or PDF/A-2, and PDF/A-3 and PDF/A-4F allows any embedded files.

`only_pdfa_embed_content` This is set for PDF/A-2a, PDF/A-2b, PDF/A-2u and PDF/A-4. I don't see a way to test the PDF/A-2 requirement so currently it will simply allow everything. Perhaps a test for at least the PDF-format will be added in future.

`Catalog_no_OCProperties` don't add `/OCProperties` to the catalog *l3pdfmeta removes this entry at the end of the document*

`Catalog_OCProperties_no_AS` do not use `/AS` optional content configuration dictionary.

`Catalog_EmbeddedFiles` ensure that an `EmbeddedFiles` name tree is in the catalog. This is required for PDF/A-4f.

`annot_widget_no_AA` (rule 6.6.2-1) no `AA` dictionary in widget annotation, this will e.g. be checked by the new hyperref driver.

`annot_widget_no_A_AA` (rule 6.9-2) no `A` and `AA` dictionary in widget.

`form_no_AA` (6.9-3) no `/AA` dictionary in form field

`unicode` that is set in the U-standards, A-2u and A-3u and means that every text should be in unicode. This is not something that can be enforced or tested from TeX, but in a current LaTeX normally `ToUnicode` are set for all fonts.

`tagged` that is set in A-2a and A-3a and means that the pdf must be tagged. This is currently neither tested not enforced somewhere.

`no_CharSet` `CharSet` is deprecated is pdf 2.0 and should not be used in A-4. `l3pdfmeta` will therefore suppress it for the engines `pdftex` and `luatex` (the other engines have no suitable option)

`omit_CID` This avoids with PDF/A-2 and newer a failure because of with missing CID identifications (e.g. from rule ISO 19005-2:2011, Clause: 6.2.11.4.2) It has only with `luatex` an effect.

`Trailer_no_Info` The `Info` dictionary has been deprecated since quite some time. Metadata should be set with XMP-data instead. In PDF A-4 now the `Info` dictionary shall not be present in the trailer dictionary at all (unless there exists a `PieceInfo` entry in the Catalog). And if it is present it should only contain the `/ModDate` entry. In texlive 2023 the engines `pdftex` and `luatex` have primitives to suppress the dictionary and `l3pdfmeta` will make use of it.

`tagged` (A- and UA-standards) The document must be a tagged PDF.

`link_Contents` (UA-standards) A link annotation must have a `Contents` entry (true for UA-1)

1.2.2 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. UA-2 requires PDF 2.0. These requirements are checked by `l3pdfmeta` when the standard is set with `\DocumentMetadata` or `\SetKeys[document/metadata]` and the version is if needed changed. At the begin of the document another check is done and error is issued if the version doesn't fit as this means that the document has used conflicting standard and version setting.

`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 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, for UA-1 which requires PDF 1.7 or lower, and for UA-2 which requires PDF 2.0. These requirements are checked by `l3pdfmeta` when the standard is set with `\DocumentMetadata` or `\SetKeys[document/metadata]` and the version is if needed changed. At the begin of the document another check is done and error is issued if the version doesn't fit as this means that the document has used conflicting standard and version setting.

`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).

1.3 Colorprofiles and OutputIntent

The pdf/A standards require that a color profile is embedded and referenced in the catalog in the `/OutputIntent` array.

The problem is that the pdf/A standards also require, that if the PDF has more than one entry in the `/OutputIntent` array (which is allowed), their `/DestOutputProfile` should all reference the same color profile².

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

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

The interface looks like this

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

or

```
\RequirePackage{pdfmanagement}
\Setkeys[document/metadata]
{
  %other options for example pdfstandard
  colorprofiles=
  {
    A = sRGB.icc, %or a or longer GTS_PDFA1 = sRGB.icc
    X = FOGRA39L_coated.icc, % or x or longer GTS_PDFX
    ISO_PDFE1 = whatever.icc
  }
}
```

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

sRGB.icc and FOGRA39L_coated.icc (from the colorprofiles package are predefined and will work directly³. whatever.icc will need special setup in the document preamble to declare the values for the OutputIntent dictionary, but the interface hasn't be added yet. This will be decided later.

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

```
\SetKeys[document/metadata]
{
  %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.

1.4 Regression tests

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

`\pdfmeta_set_regression_data:` `\pdfmeta_set_regression_data:`

This sets various metadata to values needed by the L^AT_EX regression tests. It also sets the seed for random functions. If a current l3backend is used and `\c_sys_timestamp_str` is available, the command does not set dates, but assumes that the environment variable `SOURCE_DATE_EPOCH` is used.

2 XMP-metadata

XMP-metadata are data in XML format embedded in a stream inside the PDF and referenced from the /Catalog. Such a XMP-metadata stream contains various document related data, is required by various PDF standards and can replace or extend the data in the /Info dictionary. In PDF 2.0 the /Info dictionary is actually deprecated and only XMP-metadata should be used for the metadata of the PDF.

The content of a XMP-metadata stream is not a fix set of data. Typically fields like the title, the author, the language and keywords will be there. But standards like e.g. ZUGferd (a standard for electronic bills) can require to add more fields, and it is also possible to define and add purely local data.

In some workflows (e.g. if dvips + ghostscript is used) a XMP-metadata stream with some standard content is added automatically by the backend, but normally it must be created with code.

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

For this task the packages `hyperxmp`, `xmpincl` or `pdfx` (which uses `xmpincl`) can be used, but all these packages are not compatible with the `pdfmanagement`⁴. The following code is meant as replacement for these packages.

`hyperxmp` uses `\hypersetup` as user interface to enter the XMP-metadata. This syntax is also supported by the new code⁵, so if `hyperref` has been loaded, e.g. `pdftitle=xxx` can be used to set the title. But XMP-metadata shouldn't require to use `hyperref` and in a future version an interface without `hyperref` will be added.

There is currently no full user interface command to extend the XMP-metadata with for example the code needed for ZUGferd, they will be added in a second step.

2.1 Debug option

The resulting XMP-packet can be written to an external file by activating a debug option

```
\DocumentMetadata{debug={xmp-export}}
%or
\DocumentMetadata{debug={xmp-export=true}}
%or
\DocumentMetadata{debug={xmp-export=filename}}
%or
\SetKeys[document/metadata]{debug={xmp-export}}
```

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\000c` and then back to UTF-8 by `hyperxmp` and in the course of this action the XML-escapings are applied. `pdfx` uses `\pdfstringdef` together with a special fontencoding (similar to the PU-encoding of `hyperref`) for a similar aim. The code here is based on `\text_purify:n` followed by a few replacements for the escaping.

User data should normally be declared in the preamble (or even in the `\DocumentMetadata` command), and consist of rather simple text; `&` can be entered as `\&` (but directly `&` will normally work too), babel shorthands should not be used. Some data are interpreted as comma lists, in this cases commas which are part of the text should be protected by braces. In some cases a text in brackets like `[en]` is interpreted as language tag, if they are part of a text they should be protected by braces too. XMP-metadata are stored uncompressed in the PDF so if you are unsure if a value has been passed correctly, open the PDF in an editor, copy the whole block and pass it to a validator, e.g. <https://www.w3.org/RDF/Validator/>.

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

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

2.3 User interfaces and differences to hyperxmp

2.3.1 PDF standards

The hyperxmp/hyperref keys pdfapart, pdfaconformance, pdfuapart, pdfxstandard and pdfa are ignored by this code. Standards must be set with the pdfstandard key of \DocumentMetadata or \SetKeys[document/metadata]. This key can be used more than once, e.g.

```
pdfstandard=A-2b,pdfstandard=X-4,pdfstandard=UA-1.
```

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

2.3.2 Declarations

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

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

2.3.3 Dates

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

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

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

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

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

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

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

2.4 Language

The code assumes that a default language is always declared (as the `pdfmanagement` gives the `/Lang` entry in the catalog a default value) This language can be changed with the key `lang` of `\DocumentMetadata` and `\SetKeys[document/metadata]`. This is the preferred method. `babel` will look for this value and adjust its language settings. The `hyperref` key `pdflang` is also honored. Its value should be a simple language tag like `de` or `de-DE`.

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

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

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

2.5 Rights

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

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

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

2.6 PDF related data

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

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

2.7 Document data

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

2.8 User commands

The XMP-meta data are added automatically. This can be suppressed with the `document/metadata` key `xmp`.

`\pdfmeta_xmp_add:n` `\pdfmeta_xmp_add:n` $\langle XML \rangle$

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` $\langle prefix \rangle$ $\langle uri \rangle$

With this command a xmlns name space can be added. The $\langle uri \rangle$ argument is expanded, a hash can be input with `\c_hash_str`.

With the two following commands PDF declarations can be added to the XMP metadata (see <https://pdfa.org/wp-content/uploads/2019/09/PDF-Declarations.pdf>).

`\pdfmeta_xmp_add_declaration:n` `\pdfmeta_xmp_add_declaration:n` $\langle uri \rangle$
`\pdfmeta_xmp_add_declaration:e`

This add a PDF declaration with the required `conformsTo` property to the XMP metadata. $\langle uri \rangle$ should not be empty and is a URI specifying the standard or profile referred to by the PDF Declaration. If the uri contains a hash, use `\c_hash_str` to escape it and use the `e` variant to expand it.

`\pdfmeta_xmp_add_declaration:nnnnn` `\pdfmeta_xmp_add_declaration:nnnnn`
`\pdfmeta_xmp_add_declaration:(ennnn|eeenn)` $\langle uri \rangle$ $\langle By \rangle$ $\langle Date \rangle$ $\langle Credentials \rangle$ $\langle Report \rangle$

This add a PDF declaration to the XMP metadata similar to `\pdfmeta_xmp_add_declaration:n`. With $\langle By \rangle$, $\langle Date \rangle$, $\langle Credentials \rangle$, $\langle Report \rangle$ the optional fields `claimBy` (text), `claimDate` (iso date), `claimCredentials` (text) and `claimReport` (uri) of the `claimData` property can be given. If `\pdfmeta_xmp_add_declaration:nnnnn` is used twice with the same $\langle uri \rangle$ argument the `claimData` are concatenated. There is no check if the `claimData` are identical.

The following two commands can be used to extend the schema declarations in the XMP metadata. This is for example needed to implement a standard like ZUGferd/Factor X for invoices. A schema declaration should be added only once but as this task is probably not needed frequently only light guards are there to avoid duplicated entries.

```
\pdfmeta_xmp_schema_new:nnn \pdfmeta_xmp_schema_new:nnn {<text>} {<prefix>} {<uri>}
```

`<text>` is some string describing the schema, e.g. `PDF/A~Identification~Schema`, `<prefix>` is the unique prefix used by the schema. This prefix must be declared first with `\pdfmeta_xmp_xmlns_new:nn`. If a schema with this prefix has already been declared, it will currently be ignored with a warning. The `<uri>` is expanded, so a hash can for example be given as `\c_hash_str`.

```
\pdfmeta_xmp_property_new:nnnnn \pdfmeta_xmp_property_new:nnnnn {<schema prefix>}
                               {<name>} {<type>} {<category>} {<description>}
```

If the new property already exists in the schema (as identified by the combination of `<schema prefix>` and `<name>`) the property is silently ignore. `<schema prefix>` is the prefix declared with the previous command. schema, e.g. `PDF/A~Identification~Schema`, `<name>` is a short string that identifies the property, e.g. `xmpMM` or `year`. It must be unique in the properties of a schema. `<type>` is e.g. `URI` or `Integer` or `Text`, `<category>` is e.g. `internal` or `external`, `<description>` is a free description string.

3 l3pdfmeta implementation

```
1 <@@=pdfmeta>
2 <*header>
3 \ProvidesExplPackage{l3pdfmeta}{2026-04-15}{0.96z}
4   {PDF-Standards---LaTeX PDF management bundle}
5 </header>
```

Message for unknown standards

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

Message for unknown standard family

```
8 <*package>
9 \msg_new:nnn {pdf }{unknown-standard-family}{The-standard-family-'#1'-is-unknown}
```

Message for not fitting pdf version

```
10 \msg_new:nnn {pdf }{wrong-pdfversion}
11   {PDF-version-#1-is-too-#2-for-standard-'#3'.\}
12   {Check-if-there-are-conflicting-PDF-standard\}
13   {and-PDF-version-settings!}
```

Messages for embedded files

```
14 \msg_new:nnn {pdf }{validation-failure}
15   {
16     PDF-standard-validation-failure.\}
17     #1
18   }
```

```
\l__pdfmeta_tmpa_tl
\l__pdfmeta_tmpb_tl
19 \tl_new:N \l__pdfmeta_tmpa_tl
\l__pdfmeta_tmpa_str
20 \tl_new:N \l__pdfmeta_tmpb_tl
\g__pdfmetatmpa_str
21 \str_new:N \l__pdfmeta_tmpa_str
\l__pdfmeta_tmpa_seq
22 \str_new:N \g__pdfmeta_tmpa_str
\l__pdfmeta_tmpb_seq
23 \seq_new:N \l__pdfmeta_tmpa_seq
```

```
24 \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

These internal properties will contain the active settings for a standard family. `\g__pdfmeta_standard_prop` will be used to contain the requirements of the current family for a validation.

```
\g__pdfmeta_standard_prop
\g__pdfmeta_standard_A_prop
\g__pdfmeta_standard_UA_prop25 \prop_new:N \g__pdfmeta_standard_prop
\g__pdfmeta_standard_X_prop26 \prop_new:N \g__pdfmeta_standard_A_prop
27 \prop_new:N \g__pdfmeta_standard_UA_prop
28 \prop_new:N \g__pdfmeta_standard_X_prop
```

(End of definition for \g__pdfmeta_standard_prop and others.)

3.1.2 Functions to check a requirement

`\pdfmeta_standard_family:nn` This allows to run tests for another standard family: it switches the requirements, executes the code stored in the second argument and then switches back.

```
29 \cs_new_protected:Npn \pdfmeta_standard_family:nn #1 #2
30 {
31   \prop_if_exist:cTF { g__pdfmeta_standard_#1_prop }
32   {
```

until documentmetadata-support is update we can not be sure that the A prop is properly filled.

```
33   \prop_gset_eq:NN \g__pdfmeta_standard_A_prop \g__pdfmeta_standard_prop
34   \prop_gset_eq:Nc \g__pdfmeta_standard_prop { g__pdfmeta_standard_#1_prop }
35   #2
36   \prop_gset_eq:NN \g__pdfmeta_standard_prop \g__pdfmeta_standard_A_prop
37   }
38   {
39     \msg_warning:nnn {pdf} {unknown-standard-family}{#1}
40   }
41 }
```

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

At first two commands to get the standard value if needed:

`\pdfmeta_standard_item:n`

```
42 \cs_new:Npn \pdfmeta_standard_item:n #1
43 {
44   \prop_item:Nn \g__pdfmeta_standard_prop {#1}
45 }
```

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

`\pdfmeta_standard_get:nN`

```
46 \cs_new_protected:Npn \pdfmeta_standard_get:nN #1 #2
47 {
48   \prop_get:NnN \g__pdfmeta_standard_prop {#1} #2
49 }
```

(End of definition for `\pdfmeta_standard_get:nN`. This function is documented on page 3.)

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`

```
50 \prg_new_conditional:Npnn \pdfmeta_standard_verify:n #1 {T,F,TF}
51 {
52   \prop_if_in:NnTF \g__pdfmeta_standard_prop {#1}
53   {
54     \prg_return_false:
55   }
56   {
57     \prg_return_true:
58   }
59 }
```

(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.

```
60 \prg_new_protected_conditional:Npnn \pdfmeta_standard_verify:nn #1 #2 {T,F,TF}
61 {
62   \prop_if_in:NnTF \g__pdfmeta_standard_prop {#1}
63   {
64     \cs_if_exist:cTF {__pdfmeta_standard_verify_handler_#1:nn}
65     {
66       \exp_args:Nnne
67       \use:c
68       {__pdfmeta_standard_verify_handler_#1:nn}
69       { #2 }
70       { \prop_item:Nn \g__pdfmeta_standard_prop {#1} }
71     }
72     {
73       \prg_return_false:
74     }
75   }
76   {
77     \prg_return_true:
78   }
79 }
```

(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

```
80 %
81 \cs_new_protected:Npn \__pdfmeta_standard_verify_handler_min_pdf_version:nn #1 #2
82 {
83   \pdf_version_compare:NnTF <
84     { #2 }
85     {\prg_return_false:}
86     {\prg_return_true:}
87 }
```

(End of definition for __pdfmeta_standard_verify_handler_min_pdf_version:nn.)

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

_standard_verify_handler_max_pdf_version:nn

```
88 %
89 \cs_new_protected:Npn \__pdfmeta_standard_verify_handler_max_pdf_version:nn #1 #2
90 {
91   \pdf_version_compare:NnTF >
92     { #2 }
93     {\prg_return_false:}
94     {\prg_return_true:}
95 }
```

(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

```
96 \cs_new_protected:Npn \__pdfmeta_standard_verify_handler_named_actions:nn #1 #2
97 {
98   \tl_if_in:nnTF { #2 }{ #1 }
99     {\prg_return_true:}
100     {\prg_return_false:}
101 }
```

(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

```
102 \cs_new_protected:Npn \__pdfmeta_standard_verify_handler_annot_action_A:nn #1 #2
103 {
104   \tl_if_in:nnTF { #2 }{ #1 }
105     {\prg_return_true:}
106     {\prg_return_false:}
107 }
```

(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`

```
108 \cs_new_protected:Npn \_pdfmeta_standard_verify_handler_outputintent_subtype:nn #1 #2
109 {
110   \tl_if_eq:nnTF { #2 }{ #1 }
111     {\prg_return_true:}
112     {\prg_return_false:}
113 }
```

(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:

```
114 \cs_new_protected:Npn \_pdfmeta_verify_pdfa_annot_flags:
115 {
116   \bitset_set_true:Nn \l_pdfannot_F_bitset {Print}
117   \bitset_set_false:Nn \l_pdfannot_F_bitset {Hidden}
118   \bitset_set_false:Nn \l_pdfannot_F_bitset {Invisible}
119   \bitset_set_false:Nn \l_pdfannot_F_bitset {NoView}
120   \pdfannot_dict_put:nnn {link/URI}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
121   \pdfannot_dict_put:nnn {link/GoTo}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
122   \pdfannot_dict_put:nnn {link/GoToR}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
123   \pdfannot_dict_put:nnn {link/Launch}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
124   \pdfannot_dict_put:nnn {link/Named}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
125 }
```

At begin document this should be checked:

```
126 \hook_gput_code:nnn {begindocument} {pdf}
127 {
128   \pdfmeta_standard_verify:nF { annot_flags }
129   { \_pdfmeta_verify_pdfa_annot_flags: }
130   \pdfmeta_standard_verify:nF { Trailer_no_Info }
131   { \_pdf_backend_omit_info:n {1} }
132   \pdfmeta_standard_verify:nF { no_CharSet }
133   { \_pdf_backend_omit_charset:n {1} }
134   \pdfmeta_standard_verify:nF { omit_CID }
135   { \_pdf_backend_omit_cidset:n {1} }
136   \_pdfmeta_check_standard_pdfversion:
137 }
```

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_prop138 \prop_new:c { g_pdfmeta_standard_pdf/A-1B_prop }
\g_pdfmeta_standard_pdf/A-2U_prop139 \prop_gset_from_keyval:cn { g_pdfmeta_standard_pdf/A-1B_prop }
\g_pdfmeta_standard_pdf/A-3A_prop140 {
\g_pdfmeta_standard_pdf/A-3B_prop141 ,name = pdf/A-1B
\g_pdfmeta_standard_pdf/A-3U_prop142 ,type = A
\g_pdfmeta_standard_pdf/A-4_prop143 ,level = 1
\g_pdfmeta_standard_pdf/A-4F_prop144 ,conformance = B
145 ,year = 2005
146 ,min_pdf_version = 1.4 %minimum
147 ,max_pdf_version = 1.4 %maximum
148 ,no_encryption =
149 ,no_external_content = % no F, FFilter, or FDecodeParms in stream dicts
150 ,no_embed_content = % no EF key in filespec, no /Type/EmbeddedFiles
151 ,max_string_size = 65535
152 ,max_array_size = 8191
153 ,max_dict_size = 4095
154 ,max_obj_num = 8388607
155 ,max_nest_qQ = 28
156 ,named_actions = {NextPage, PrevPage, FirstPage, LastPage}
157 ,annot_flags =
158 %booleans. Only the existence of the key matter.
159 %If the entry is added it means a requirements is there
160 %(in most cases "don't use ...")
161 %
162 %=====
163 % Rule 6.1.13-1 CosDocument, isOptionalContentPresent == false
164 ,Catalog_no_OCProperties =
165 % Rule 6.9-4 The AS key shall not appear in any optional content configuration dictionary.
166 % actually only starting with A-2 but doesn't harm here either
167 ,Catalog_OCProperties_no_AS=
168 %=====
169 % Rule 6.6.1-1: PDAction, S == "GoTo" || S == "GoToR" || S == "Thread"
170 % || S == "URI" || S == "Named" || S == "SubmitForm"
171 % means: no /S/Launch, /S/Sound, /S/Movie, /S/ResetForm, /S/ImportData,
172 % /S/JavaScript, /S/Hide
173 ,annot_action_A = {GoTo,GoToR,Thread,URI,Named,SubmitForm}
174 %=====
175 % Rule 6.6.2-1: PDAnnot, Subtype != "Widget" || AA_size == 0
176 % means: no AA dictionary
177 ,annot_widget_no_AA =
178 %=====
179 % Rule 6.9-2: PDAnnot, Subtype != "Widget" || (A_size == 0 && AA_size == 0)
180 % (looks like a tightening of the previous rule)
181 ,annot_widget_no_A_AA =
182 %=====
183 % Rule 6.9-1 PDAcroForm, NeedAppearances == null || NeedAppearances == false
184 ,form_no_NeedAppearances =
185 %=====
```

```

186 %Rule 6.9-3 PDFormField, AA_size == 0
187 ,form_no_AA =
188 %=====
189 % to be continued https://docs.verapdf.org/validation/pdfa-part1/
190 % - Outputintent/colorprofiles requirements
191 % an outputintent should be loaded and is unique.
192 ,outputintent_A = {GTS_PDFA1}
193 % - no Alternates key in image dictionaries
194 % - no OPI, Ref, Subtype2 with PS key in xobjects
195 % - Interpolate = false in images
196 % - no TR, TR2 in ExtGstate
197 }
198
199 %A-2b =====
200 \prop_new:c { g__pdfmeta_standard_pdf/A-2B_prop }
201 \prop_gset_eq:cc
202 { g__pdfmeta_standard_pdf/A-2B_prop }
203 { g__pdfmeta_standard_pdf/A-1B_prop }
204 \prop_gput:cnn
205 { g__pdfmeta_standard_pdf/A-2B_prop }{name}{pdf/A-2B}
206 \prop_gput:cnn
207 { g__pdfmeta_standard_pdf/A-2B_prop }{year}{2011}
208 \prop_gput:cnn
209 { g__pdfmeta_standard_pdf/A-2B_prop }{level}{2}
210 % embedding files is allowed (with restrictions)
211 \prop_gremove:cn
212 { g__pdfmeta_standard_pdf/A-2B_prop }
213 { no_embed_content }
214 \prop_gput:cnn
215 { g__pdfmeta_standard_pdf/A-2B_prop }
216 { only_pdfa_embed_content }
217 {}
218 \prop_gput:cnn
219 { g__pdfmeta_standard_pdf/A-2B_prop }{max_pdf_version}{1.7}
220 \prop_gput:cnn
221 { g__pdfmeta_standard_pdf/A-2B_prop }{omit_CID}{}
222 % OCG layers are allowed (with restrictions)
223 \prop_gremove:cn
224 { g__pdfmeta_standard_pdf/A-2B_prop }
225 { Catalog_no_OCProperties }
226
227 %A-2u =====
228 \prop_new:c { g__pdfmeta_standard_pdf/A-2U_prop }
229 \prop_gset_eq:cc
230 { g__pdfmeta_standard_pdf/A-2U_prop }
231 { g__pdfmeta_standard_pdf/A-2B_prop }
232 \prop_gput:cnn
233 { g__pdfmeta_standard_pdf/A-2U_prop }{name}{pdf/A-2U}
234 \prop_gput:cnn
235 { g__pdfmeta_standard_pdf/A-2U_prop }{conformance}{U}
236 \prop_gput:cnn
237 { g__pdfmeta_standard_pdf/A-2U_prop }{unicode}{}
238
239 %A-2a =====

```

```

240 \prop_new:c { g__pdfmeta_standard_pdf/A-2A_prop }
241 \prop_gset_eq:cc
242   { g__pdfmeta_standard_pdf/A-2A_prop }
243   { g__pdfmeta_standard_pdf/A-2B_prop }
244 \prop_gput:cnn
245   { g__pdfmeta_standard_pdf/A-2A_prop }{name}{pdf/A-2A}
246 \prop_gput:cnn
247   { g__pdfmeta_standard_pdf/A-2A_prop }{conformance}{A}
248 \prop_gput:cnn
249   { g__pdfmeta_standard_pdf/A-2A_prop }{tagged}{ }
250
251
252 %A-3b =====
253 \prop_new:c { g__pdfmeta_standard_pdf/A-3B_prop }
254 \prop_gset_eq:cc
255   { g__pdfmeta_standard_pdf/A-3B_prop }
256   { g__pdfmeta_standard_pdf/A-2B_prop }
257 \prop_gput:cnn
258   { g__pdfmeta_standard_pdf/A-3B_prop }{name}{pdf/A-3B}
259 \prop_gput:cnn
260   { g__pdfmeta_standard_pdf/A-3B_prop }{year}{2012}
261 \prop_gput:cnn
262   { g__pdfmeta_standard_pdf/A-3B_prop }{level}{3}
263 % embedding files is allowed
264 \prop_gremove:cn
265   { g__pdfmeta_standard_pdf/A-3B_prop }
266   { only_pdfa_embed_content }
267 %A-3u =====
268 \prop_new:c { g__pdfmeta_standard_pdf/A-3U_prop }
269 \prop_gset_eq:cc
270   { g__pdfmeta_standard_pdf/A-3U_prop }
271   { g__pdfmeta_standard_pdf/A-3B_prop }
272 \prop_gput:cnn
273   { g__pdfmeta_standard_pdf/A-3U_prop }{name}{pdf/A-3U}
274 \prop_gput:cnn
275   { g__pdfmeta_standard_pdf/A-3U_prop }{conformance}{U}
276 \prop_gput:cnn
277   { g__pdfmeta_standard_pdf/A-3U_prop }{unicode}{ }
278
279 %A-3a =====
280 \prop_new:c { g__pdfmeta_standard_pdf/A-3A_prop }
281 \prop_gset_eq:cc
282   { g__pdfmeta_standard_pdf/A-3A_prop }
283   { g__pdfmeta_standard_pdf/A-3B_prop }
284 \prop_gput:cnn
285   { g__pdfmeta_standard_pdf/A-3A_prop }{name}{pdf/A-3A}
286 \prop_gput:cnn
287   { g__pdfmeta_standard_pdf/A-3A_prop }{conformance}{A}
288 \prop_gput:cnn
289   { g__pdfmeta_standard_pdf/A-3A_prop }{tagged}{ }
290
291 %A-4 =====
292 \prop_new:c { g__pdfmeta_standard_pdf/A-4_prop }
293 \prop_gset_eq:cc

```

```

294 { g__pdfmeta_standard_pdf/A-4_prop }
295 { g__pdfmeta_standard_pdf/A-3U_prop }
296 \prop_gput:cnn
297 { g__pdfmeta_standard_pdf/A-4_prop }{name}{pdf/A-4}
298 \prop_gput:cnn
299 { g__pdfmeta_standard_pdf/A-4_prop }{level}{4}
300 \prop_gput:cnn
301 { g__pdfmeta_standard_pdf/A-4_prop }{min_pdf_version}{2.0}
302 \prop_gput:cnn
303 { g__pdfmeta_standard_pdf/A-4_prop }{year}{2020}
304 \prop_gput:cnn
305 { g__pdfmeta_standard_pdf/A-4_prop }{no_CharSet}{}
306 \prop_gput:cnn
307 { g__pdfmeta_standard_pdf/A-4_prop }{Trailer_no_Info}{}
308 \prop_gput:cnn
309 { g__pdfmeta_standard_pdf/A-4_prop }{only_pdfa_embed_content}{}
310 \prop_gremove:cn
311 { g__pdfmeta_standard_pdf/A-4_prop }{conformance}
312 \prop_gremove:cn
313 { g__pdfmeta_standard_pdf/A-4_prop }{max_pdf_version}
314 \prop_gremove:cn
315 { g__pdfmeta_standard_pdf/A-4_prop }{Catalog_OCProperties_no_AS}
316 %A-4f =====
317 \prop_new:c { g__pdfmeta_standard_pdf/A-4F_prop }
318 \prop_gset_eq:cc
319 { g__pdfmeta_standard_pdf/A-4F_prop }
320 { g__pdfmeta_standard_pdf/A-4_prop }
321 \prop_gput:cnn
322 { g__pdfmeta_standard_pdf/A-4F_prop }{conformance}{F}
323 % containsEmbeddedFiles == true ISO 19005-4:2020, Clause: 6.9, Test number: 5
324 \prop_gput:cnn
325 { g__pdfmeta_standard_pdf/A-4F_prop }{Catalog_EmbeddedFiles}{}
326 % can contain any file
327 \prop_gremove:cn
328 { g__pdfmeta_standard_pdf/A-4F_prop }{only_pdfa_embed_content}

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

```

3.1.5 pdf/UA

\g__pdfmeta_standard_pdf/UA-1_prop

\g__pdfmeta_standard_pdf/UA-2_prop

```

329 \prop_new:c { g__pdfmeta_standard_pdf/UA-1_prop}
330 \prop_new:c { g__pdfmeta_standard_pdf/UA-2_prop}
331 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/UA-1_prop }
332 {
333     ,name                = pdf/UA-1
334     ,level                = 1
335     ,year                 = 2014
336     ,min_pdf_version     = 1.4           %minimum
337     ,max_pdf_version     = 1.7           %maximum
338     ,tagged               =
339     ,link_Contents       =
340 }

```

```

341 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/UA-2_prop }
342 {
343   ,name           = pdf/UA-2
344   ,level          = 2
345   ,year           = 2024
346   ,min_pdf_version = 2.0           %minimum
347   ,tagged         =
348 }

```

(End of definition for `\g__pdfmeta_standard_pdf/UA-1_prop` and `\g__pdfmeta_standard_pdf/UA-2_prop`.)

3.1.6 pdf/X

We know only the names ...

```

\g__pdfmeta_standard_pdf/X-4_prop
\g__pdfmeta_standard_pdf/X-4P_prop
\g__pdfmeta_standard_pdf/X-5G_prop349 \prop_new:c { g__pdfmeta_standard_pdf/X-4_prop}
\g__pdfmeta_standard_pdf/X-5N_prop350 \prop_new:c { g__pdfmeta_standard_pdf/X-4P_prop}
\g__pdfmeta_standard_pdf/X-5PG_prop351 \prop_new:c { g__pdfmeta_standard_pdf/X-5G_prop}
\g__pdfmeta_standard_pdf/X-6_prop352 \prop_new:c { g__pdfmeta_standard_pdf/X-5N_prop}
\g__pdfmeta_standard_pdf/X-6N_prop353 \prop_new:c { g__pdfmeta_standard_pdf/X-5PG_prop}
\g__pdfmeta_standard_pdf/X-6P_prop354 \prop_new:c { g__pdfmeta_standard_pdf/X-6_prop}
355 \prop_new:c { g__pdfmeta_standard_pdf/X-6N_prop}
356 \prop_new:c { g__pdfmeta_standard_pdf/X-6P_prop}
357
358 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/X-4_prop }
359 {
360   ,name           = PDF/X-4
361 }
362 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/X-4P_prop }
363 {
364   ,name           = PDF/X-4p
365 }
366 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/X-5G_prop }
367 {
368   ,name           = PDF/X-5g
369 }
370 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/X-5N_prop }
371 {
372   ,name           = PDF/X-5n
373 }
374 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/X-5PG_prop }
375 {
376   ,name           = PDF/X-5pg
377 }
378 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/X-6_prop }
379 {
380   ,name           = PDF/X-6
381 }
382 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/X-6N_prop }
383 {
384   ,name           = PDF/X-6n
385 }

```

```

386 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/X-6P_prop }
387 {
388   ,name          = PDF/X-6p
389 }

```

(End of definition for `\g__pdfmeta_standard_pdf/X-4_prop` and others.)

3.1.7 Embedded Files

Standard 4-AF is needed if we add AF files for tagging but it also requires an Embedded-Files name tree, so we test at the end if the name tree is empty and add a small readme if yes

```

390 \AddToHook{begindocument/end}
391 {
392   \pdfmeta_standard_verify:nF{Catalog_EmbeddedFiles}
393   {
394     \tl_gput_right:Nn\g__kernel_pdfmanagement_end_run_code_tl
395     {
396       \pdfdict_if_empty:nT { g__pdf_Core/Catalog/Names/EmbeddedFiles }
397       {
398         \group_begin:
399         \pdfdict_put:nne {l_pdffile/Filespec} {Desc}{(note~about~PDF/A-4F)}
400         \pdfdict_put:nnn { l_pdffile/Filespec }{AFRelationship} { /Unspecified }
401         \pdffile_embed_stream:nnN
402         {The~document~was~declared~to~be~of~type~PDF/A-4f~but~hasn't~any~attachments.~
403           LaTeX~therefore~added~this~dummy~file.}
404         {pdf-A4f.txt}
405         \l__pdfmeta_tmpa_tl
406         \exp_args:Nne \__pdf_backend_Names_gpush:nn{EmbeddedFiles}{(pdf-A4f)~\l__pdfmeta_tmpa_
407         \group_end:
408       }
409     }
410   }
411 }

```

Before writing the xml we check if there are embedded files we know of. For A-4 we adjust the standard to A-4F is needed.

```

412 \AddToHook{enddocument/end}
413 {
414   \pdfmeta_standard_verify:nF{no_embed_content}
415   {
416     \bool_lazy_or:nnT
417     { ! \int_if_zero_p:n { \g_pdffile_embed_pdfa_int } }
418     { ! \int_if_zero_p:n { \g_pdffile_embed_nonpdfa_int } }
419     {
420       \prop_get:NnNT\g__pdfmeta_standard_prop { name } \l__pdfmeta_tmpa_tl
421       {
422         \msg_warning:nne { pdf } { validation-failure }
423         {
424           Embedded~files~detected.\iow_newline:
425           This~is~not~allowed~in~standard~\l__pdfmeta_tmpa_tl
426         }
427       }
428     }
429   }
430 }

```

```

428     }
429   }
430   \pdfmeta_standard_verify:nF {only_pdfa_embed_content}
431   {
432     \int_if_zero:nF { \g_pdffile_embed_nonpdfa_int }
433     {
434       \prop_get:NnNT\g__pdfmeta_standard_prop { name } \l__pdfmeta_tmpa_tl
435       {
436         \str_if_eq:VnTF {\l__pdfmeta_tmpa_tl} { pdf/A-4 }
437         {
438           \msg_note:nne { pdf } { validation-failure }
439           {
440             Embedded-non-PDF-files-detected.\iow_newline:
441             Check~if~the~standard~must~be~changed~to~PDF/A-4F
442           }
443         }
444         {
445           \msg_warning:nne { pdf } { validation-failure }
446           {
447             Embedded~non-PDF-files-detected.\iow_newline:
448             This~is~not~allowed~in~standard~\l__pdfmeta_tmpa_tl
449           }
450         }
451       }
452     }
453   }
454 }

```

3.1.8 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.

```
455 \prop_new:N \g_pdfmeta_outputintents_prop
    (End of definition for \g_pdfmeta_outputintents_prop.)
```

Some keys to fill the property.

```
456 \keys_define:nn { document / metadata }
457   {
458     colorprofiles .code:n =
459     {
460       \keys_set:nn { document / metadata / colorprofiles }{#1}
461     }
462   }
463 \keys_define:nn { document / metadata / colorprofiles }
464   {
465     ,A .code:n =
466     {
467       \tl_if_blank:nF {#1}
468       {
469         \prop_gput:Nnn \g_pdfmeta_outputintents_prop
470         { GTS_PDFA1 } {#1}
471       }
472     }
473     ,a .code:n =
474     {
475       \tl_if_blank:nF {#1}
476       {
477         \prop_gput:Nnn \g_pdfmeta_outputintents_prop
478         { GTS_PDFA1 } {#1}
479       }
480     }
481     ,X .code:n =
482     {
483       \tl_if_blank:nF {#1}
484       {
485         \prop_gput:Nnn \g_pdfmeta_outputintents_prop
486         { GTS_PDFX } {#1}
487       }
488     }
489     ,x .code:n =
490     {
491       \tl_if_blank:nF {#1}
492       {
493         \prop_gput:Nnn \g_pdfmeta_outputintents_prop
494         { GTS_PDFX } {#1}
495       }
496     }
497   }
```

```

497     ,unknown .code:n =
498     {
499         \tl_if_blank:nF {#1}
500         {
501             \exp_args:NNo
502             \prop_gput:Nnn \g__pdfmeta_outputintents_prop
503             { \l_keys_key_str } {#1}
504         }
505     }
506 }

```

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

```

507 \pdfdict_new:n {l_pdfmeta/outputintent}
508 \pdfdict_put:nnn {l_pdfmeta/outputintent}
509 {Type}{/OutputIntent}
510 \prop_const_from_keyval:cn { c__pdfmeta_colorprofile_sRGB.icc}
511 {
512     ,OutputConditionIdentifier=IEC~sRGB
513     ,Info=IEC-61966-2.1-Default~RGB~colour~space~~~sRGB
514     ,RegistryName=http://www.iec.ch
515     ,N = 3
516 }
517 \prop_const_from_keyval:cn { c__pdfmeta_colorprofile_FOGRA39L_coated.icc}
518 {
519     ,OutputConditionIdentifier=FOGRA39L~Coated
520     ,Info={Offset~printing,~according~to~ISO~12647-2:2004/Amd-1,~OFCOM,~%
521         paper~type~1~or~2~==~coated~art,~115~g/m2,~tone~value~increase~
522         curves~A~(CMY)~and~B~(K)}
523     ,RegistryName=http://www.fogra.org
524     ,N = 4
525 }

```

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

```

526 \cs_new_protected:Npn \_pdfmeta_embed_colorprofile:n #1%#1 file name
527 {
528     \pdf_object_if_exist:nF { __color_icc_ #1 }
529     {
530         \pdf_object_new:n { __color_icc_ #1 }
531         \pdf_object_write:nne { __color_icc_ #1 } { fstream }
532         {
533             {/N\c_space_tl
534                 \prop_item:cn{c__pdfmeta_colorprofile_#1}{N}
535             }
536             {#1}
537         }
538     }
539 }
540

```

```

541 \cs_new_protected:Npn \__pdfmeta_write_outputintent:nn #1 #2 %#1 file name, #2 subtype
542 {
543   \group_begin:
544   \pdfdict_put:nne {l_pdfmeta/outputintent}{S}{/\str_convert_pdfname:n{#2}}
545   \pdfdict_put:nne {l_pdfmeta/outputintent}
546     {DestOutputProfile}
547     {\pdf_object_ref:n{ __color_icc_ #1 }}
548   \clist_map_inline:nn { OutputConditionIdentifier, Info, RegistryName }
549     {
550       \prop_get:cnNT
551       { c__pdfmeta_colorprofile_#1}
552       { ##1 }
553       \l__pdfmeta_tmpa_tl
554       {
555         \pdf_string_from_unicode:nVN {utf8/string}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_str
556         \pdfdict_put:nne
557           {l_pdfmeta/outputintent}{##1}{\l__pdfmeta_tmpa_str}
558       }
559     }
560   \pdf_object_unnamed_write:ne {dict}{\pdfdict_use:n {l_pdfmeta/outputintent} }
561   \pdfmanagement_add:nne {Catalog}{OutputIntents}{\pdf_object_ref_last:}
562   \group_end:
563 }

```

(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

```

564 \AddToHook{begindocument/end}
565 {
566   \pdfmeta_standard_verify:nTF {outputintent_A}
567   {
568     \prop_map_inline:Nn \g__pdfmeta_outputintents_prop
569     {
570       \prop_if_exist:cTF {c__pdfmeta_colorprofile_#2}
571       {
572         \__pdfmeta_embed_colorprofile:n
573         {#2}
574         \__pdfmeta_write_outputintent:nn
575         {#2}
576         {#1}
577       }
578       {
579         \msg_warning:nnn{pdfmeta}{colorprofile-undefined}{#2}
580       }
581     }
582   }

```

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.

```

583   {
584     \exp_args:NNe
585     \prop_if_in:NnF

```

```

586     \g__pdfmeta_outputintents_prop
587     { \pdfmeta_standard_item:n { outputintent_A } }
588     {
589         \exp_args:NNe
590         \prop_gput:Nnn
591         \g__pdfmeta_outputintents_prop
592         { \pdfmeta_standard_item:n { outputintent_A } }
593         { sRGB.icc }
594     }
595     \exp_args:NNe
596     \prop_get:NnN
597     \g__pdfmeta_outputintents_prop
598     { \pdfmeta_standard_item:n { outputintent_A } }
599     \l__pdfmeta_tmpb_tl
600     \prop_if_exist:cTF {c__pdfmeta_colorprofile_\l__pdfmeta_tmpb_tl}
601     {
602         \exp_args:NV \__pdfmeta_embed_colorprofile:n \l__pdfmeta_tmpb_tl
603         \prop_map_inline:Nn \g__pdfmeta_outputintents_prop
604         {
605             \exp_args:NV
606             \__pdfmeta_write_outputintent:nn
607             \l__pdfmeta_tmpb_tl
608             { #1 }
609         }
610     }
611     {
612         \msg_warning:nne{pdfmeta}{colorprofile-undefined}{\l__pdfmeta_tmpb_tl}
613     }
614 }
615 }

```

3.2 Regression test

This is simply a copy of the backend function.

```

616 \cs_new_protected:Npn \pdfmeta_set_regression_data:
617   { \__pdf_backend_set_regression_data: }

```

4 XMP-Metadata implementation

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

```

618 \bool_new:N\g__pdfmeta_xmp_bool
619 \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.

```

620 \hook_gput_code:nnn{pdfmanagement/add}{pdfmanagement}
621   {
622     \pdfmanagement_add:nne {Info}{Producer}{(\c_sys_engine_exec_str-\c_sys_engine_version_str)}
623     \pdfmanagement_add:nne {Info}{Creator}{(LaTeX)}
624   }

```

4.1 New document keys

```
625 \cs_generate_variant:Nn\pdf_version_gset:n{e}

if the pdf version is wrong for the standard we force the highest possible.
626 \cs_new_protected:Nn\__pdfmeta_force_standard_pdfversion:
627 {
628   \pdfmeta_standard_verify:nnF { min_pdf_version }
629   { \pdf_version: }
630   {
631     \pdfmeta_standard_verify:nTF { max_pdf_version }
632     {
633       \pdf_version_gset:n { 2.0 }
634     }
635     {
636       \pdf_version_gset:e{ \pdfmeta_standard_item:n{ max_pdf_version } }
637     }
638   }
639   \pdfmeta_standard_verify:nnF { max_pdf_version }
640   { \pdf_version: }
641   {
642     \pdf_version_gset:e{ \pdfmeta_standard_item:n{ max_pdf_version } }
643   }
644 }

At begin document we then want to test if there is a version problem.
645 \cs_new_protected:Npn \__pdfmeta_check_standard_pdfversion:
646 {
647   \clist_map_inline:nn{A,UA,X}
648   {
649     \pdfmeta_standard_family:nn { ##1 }
650     {
651       \pdfmeta_standard_verify:nnF { min_pdf_version }
652       { \pdf_version: }
653       { \msg_warning:nneee {pdf}{wrong-pdfversion} %TODO make error
654         {\pdf_version:}{low}
655         {
656           \pdfmeta_standard_item:n{name}
657         }
658       }
659       \pdfmeta_standard_verify:nnF { max_pdf_version }
660       { \pdf_version: }
661       { \msg_warning:nneee {pdf}{wrong-pdfversion}
662         {\pdf_version:}{high}
663         {
664           \pdfmeta_standard_item:n{name}
665         }
666       }
667     }
668   }
669 }

670 \keys_define:nn { document / metadata }
671 {
672   _pdfstandard .choices:nn =
```

```

673 {A-1B,A-2A,A-2B,A-2U,A-3A,A-3B,A-3U,A-4}
674 {
675   \prop_gset_eq:Nc \g__pdfmeta_standard_A_prop { g__pdfmeta_standard_pdf/#1 _prop }
676   \prop_gset_eq:NN \g__pdfmeta_standard_prop \g__pdfmeta_standard_A_prop
677   \__pdfmeta_force_standard_pdfversion:
678   \AddToDocumentProperties [document]{pdfstandard}{#1}
679 },
680 _pdfstandard / A-4F .code:n =
681 {
682   \prop_gset_eq:Nc \g__pdfmeta_standard_A_prop { g__pdfmeta_standard_pdf/#1 _prop }
683   \prop_gset_eq:NN \g__pdfmeta_standard_prop \g__pdfmeta_standard_A_prop
684   \__pdfmeta_force_standard_pdfversion:
685   \AddToDocumentProperties [document]{pdfstandard}{A-4F}
686 },
687 _pdfstandard / A-4E .code:n =
688 {
689   \prop_gset_eq:Nc \g__pdfmeta_standard_A_prop { g__pdfmeta_standard_pdf/A-4E_prop }
690   \prop_gset_eq:NN \g__pdfmeta_standard_prop \g__pdfmeta_standard_A_prop
691   \__pdfmeta_force_standard_pdfversion:
692   \AddToDocumentProperties [document]{pdfstandard}{A-4E}
693 },
694 _pdfstandard / unknown .code:n =
695 {
696   \msg_error:nnn{pdf}{unknown-standard}{#1}
697 },
698 _pdfstandard / X-4 .code:n =
699 {
700   \prop_gset_eq:Nc \g__pdfmeta_standard_X_prop { g__pdfmeta_standard_pdf/X-4_prop }
701   \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-4}
702   \__pdfmeta_xmp_add_pdfxid:
703 },
704 _pdfstandard / X-4P .code:n =
705 {
706   \prop_gset_eq:Nc \g__pdfmeta_standard_X_prop { g__pdfmeta_standard_pdf/X-4P_prop }
707   \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-4p}
708   \__pdfmeta_xmp_add_pdfxid:
709 },
710 _pdfstandard / X-5G .code:n =
711 {
712   \prop_gset_eq:Nc \g__pdfmeta_standard_X_prop { g__pdfmeta_standard_pdf/X-5G_prop }
713   \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5g}
714   \__pdfmeta_xmp_add_pdfxid:
715 },
716 _pdfstandard / X-5N .code:n =
717 {
718   \prop_gset_eq:Nc \g__pdfmeta_standard_X_prop { g__pdfmeta_standard_pdf/X-5N_prop }
719   \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5n}
720   \__pdfmeta_xmp_add_pdfxid:
721 },
722 _pdfstandard / X-5PG .code:n =
723 {
724   \prop_gset_eq:Nc \g__pdfmeta_standard_X_prop { g__pdfmeta_standard_pdf/X-5PG_prop }
725   \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5pg}
726   \__pdfmeta_xmp_add_pdfxid:

```

```

727     },
728     _pdfstandard / X-6 .code:n =
729     {
730         \prop_gset_eq:Nc \g__pdfmeta_standard_X_prop { g__pdfmeta_standard_pdf/X-6_prop }
731         \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6}
732         \__pdfmeta_xmp_add_pdfxid:
733     },
734     _pdfstandard / X-6N .code:n =
735     {
736         \prop_gset_eq:Nc \g__pdfmeta_standard_X_prop { g__pdfmeta_standard_pdf/X-6N_prop }
737         \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6n}
738         \__pdfmeta_xmp_add_pdfxid:
739     },
740     _pdfstandard / X-6P .code:n =
741     {
742         \prop_gset_eq:Nc \g__pdfmeta_standard_X_prop { g__pdfmeta_standard_pdf/X-6P_prop }
743         \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6p}
744         \__pdfmeta_xmp_add_pdfxid:
745     },
746     _pdfstandard / UA-1 .code:n =
747     {
748         \prop_gset_eq:Nc \g__pdfmeta_standard_UA_prop { g__pdfmeta_standard_pdf/UA-1_prop }
749         \pdfmeta_standard_family:nn{UA}{ \__pdfmeta_force_standard_pdfversion: }
750         \AddToDocumentProperties [document]{pdfstandard-UA}{{1}}{}}
751         \AddToHook{begindocument/before}
752         {
753             \prop_gput:Nnn \g__pdfmeta_standard_prop {omit_CID}}{}}
754     }
755 },

```

currently it is not possible to merge requirements - these need some thoughts as every standard has some common keys like the name or the yes. We therefore add some requirements manually.

```

756     _pdfstandard / UA-2 .code:n =
757     {
758         \prop_gset_eq:Nc \g__pdfmeta_standard_UA_prop { g__pdfmeta_standard_pdf/UA-2_prop }
759         \pdfmeta_standard_family:nn{UA}{ \__pdfmeta_force_standard_pdfversion: }
760         \AddToDocumentProperties [document]{pdfstandard-UA}{{2}}{2024}}

```

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

```

761     %\AddToHook{begindocument/before}
762     %{\prop_gput:Nnn \g__pdfmeta_standard_prop {Trailer_no_Info}}{}}
763     \AddToHook{begindocument/before}
764     {
765         \__pdfmeta_xmp_wtpdf_accessibility_declaration:
766         \__pdfmeta_xmp_wtpdf_reuse_declaration:
767     }
768 },
769 xmp .choice:,
770 xmp / true .code:n = { \bool_gset_true:N \g__pdfmeta_xmp_bool },
771 xmp / false .code:n = { \bool_gset_false:N \g__pdfmeta_xmp_bool},
772 xmp .default:n = true,

```

These keys allow to disable or force the wtpdf declarations. Currently the content can not be changed and once they have been disabled there are gone. This will perhaps change.

```

773 xmp / wtpdf .code:n =
774 {
775     \keys_set:nn {__pdfmeta/xmp}{#1}
776 },
777 }
778 \keys_define:nn {__pdfmeta/xmp}
779 {
780     reuse .choice:,
781     reuse / true .code:n = \__pdfmeta_xmp_wtpdf_reuse_declaration:,
782     reuse / false .code:n =
783     {
784         \cs_set_eq:NN \__pdfmeta_xmp_wtpdf_reuse_declaration: \prg_do_nothing:
785     },
786     accessibility .choice:,
787     accessibility / true .code:n = \__pdfmeta_xmp_wtpdf_accessibility_declaration:,
788     accessibility /false .code:n =
789     {
790         \cs_set_eq:NN \__pdfmeta_xmp_wtpdf_accessibility_declaration: \prg_do_nothing:
791     },
792 }

```

XMP debugging option

```

793 \bool_new:N \g__pdfmeta_xmp_export_bool
794 \str_new:N \g__pdfmeta_xmp_export_str
795
796 \keys_define:nn { document / metadata }
797 {
798     ,debug .code:n =
799     {
800         \keys_set:nn { document / metadata / debug } {#1}
801     }
802     ,debug / xmp-export .choice:
803     ,debug / xmp-export / true .code:n=
804     {
805         \bool_gset_true:N \g__pdfmeta_xmp_export_bool
806         \str_gset_eq:NN \g__pdfmeta_xmp_export_str \c_sys_jobname_str
807     }
808     ,debug / xmp-export / false .code:n =
809     {
810         \bool_gset_false:N \g__pdfmeta_xmp_export_bool
811     }
812     ,debug / xmp-export /unknown .code:n =
813     {
814         \bool_gset_true:N \g__pdfmeta_xmp_export_bool
815         \str_gset:Nn \g__pdfmeta_xmp_export_str { #1 }
816     }
817     ,debug / xmp-export .default:n = true
818 }

```

4.2 Messages

```

819 \msg_new:nnn{pdfmeta}{xmp-defined}{The~XMP~#1~`#2`~is~already~declared}
820 \msg_new:nnn{pdfmeta}{xmp-undefined}{The~XMP~#1~`#2`~is~undefined}
821 \msg_new:nnn{pdfmeta}{colorprofile-undefined}{The~colorprofile~`#1`~is~unknown}

```

4.3 Some helper commands

4.3.1 Generate a BOM

`_pdfmeta_xmp_generate_bom:`

```
822 \bool_lazy_or:nnTF
823   { \sys_if_engine luatex_p: }
824   { \sys_if_engine xetex_p: }
825   {
826     \cs_new:Npn \_pdfmeta_xmp_generate_bom:
827       { \char_generate:nn {"FEFF"}{12} }
828   }
829   {
830     \cs_new:Npn \_pdfmeta_xmp_generate_bom:
831       {
832         \char_generate:nn {"EF"}{12}
833         \char_generate:nn {"BB"}{12}
834         \char_generate:nn {"BF"}{12}
835       }
836   }
```

(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`

```
837 \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 #38 \cs_new:Npn \_pdfmeta_xmp_indent:
```

```
\_pdfmeta_xmp_decr_indent: 839 {
```

```
840   \iow_newline:
841   \prg_replicate:nn {\l_pdfmeta_xmp_indent_int}{\c_space_tl}
842 }
```

```
844 \cs_new:Npn \_pdfmeta_xmp_indent:n #1
```

```
845   {
846     \iow_newline:
847     \prg_replicate:nn {#1}{\c_space_tl}
848   }
```

```
850 \cs_new_protected:Npn \_pdfmeta_xmp_incr_indent:
```

```
851   {
852     \int_incr:N \l_pdfmeta_xmp_indent_int
853   }
```

```
854 \cs_new_protected:Npn \_pdfmeta_xmp_decr_indent:
```

```

856 {
857   \int_decr:N \l__pdfmeta_xmp_indent_int
858 }

```

(End of definition for __pdfmeta_xmp_indent: and others.)

4.3.3 Date and time handling

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

\l__pdfmeta_xmp_date_regex

```

859 \regex_new:N \l__pdfmeta_xmp_date_regex
860 \regex_set:Nn \l__pdfmeta_xmp_date_regex
861 {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.

```

862 \cs_new_protected:Npn \__pdfmeta_xmp_date_split:nN #1 #2 %#1 date, #2 seq
863 {
864   \regex_split:NnN \l__pdfmeta_xmp_date_regex {#1} #2
865 }
866 \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.

```

867 \cs_new:Npn \__pdfmeta_xmp_print_date:N #1 % seq
868 {
869   \tl_if_blank:eTF { \seq_item:Nn #1 {1} }
870   {
871     \seq_item:Nn #1 {2} %year
872     -
873     \seq_item:Nn #1 {3} %month
874     -
875     \seq_item:Nn #1 {4} % day
876     \tl_if_blank:eF
877     { \seq_item:Nn #1 {5} }
878     { T \seq_item:Nn #1 {5} } %hour
879     \tl_if_blank:eF
880     { \seq_item:Nn #1 {6} }
881     { : \seq_item:Nn #1 {6} } %minutes
882     \tl_if_blank:eF
883     { \seq_item:Nn #1 {7} }
884     { : \seq_item:Nn #1 {7} } %seconds
885     \seq_item:Nn #1 {8} %Z,+, -
886     \seq_item:Nn #1 {9}
887     \tl_if_blank:eF

```

```

888     { \seq_item:Nn #1 {10} }
889     { : \seq_item:Nn #1 {10} }
890   }
891   {
892     \seq_item:Nn #1 {1}
893   }
894 }

```

(End of definition for `__pdfmeta_xmp_print_date:N`.)

`\l__pdfmeta_xmp_currentdate_tl` The tl var contains the date of the log-file in PDF format, the seq the result split with
`\l__pdfmeta_xmp_currentdate_seq` the regex.

```

895 \tl_new:N \l__pdfmeta_xmp_currentdate_tl
896 \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.

```

897 \cs_new_protected:Npn \__pdfmeta_xmp_date_get:nNN #1 #2 #3
898   {%#1 property, #2 tl var with PDF date, #3 seq for split date
899   {
900     \tl_set:Nc #2 { \GetDocumentProperties{#1} }
901     \tl_if_blank:VTF #2
902     {
903       \seq_set_eq:NN #3 \l__pdfmeta_xmp_currentdate_seq
904       \tl_set_eq:NN #2 \l__pdfmeta_xmp_currentdate_tl
905     }
906     {
907       \__pdfmeta_xmp_date_split:VN #2 #3
908     }
909   }

```

(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`

```

910 \cs_new_protected:Npn \__pdfmeta_xmp_create_uuid:nN #1 #2
911   {
912     \str_set:Nc #2 { \str_lowercase:f{ \tex_mdffivesum:D{#1} } }
913     \str_set:Nc #2
914     { uuid:
915       \str_range:Nnn #2 {1} {8}
916       - \str_range:Nnn #2 {9} {12}
917       -4 \str_range:Nnn #2 {13} {15}
918       -8 \str_range:Nnn #2 {16} {18}
919       - \str_range:Nnn #2 {19} {30}
920     }
921   }

```

(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.

```
922 \cs_new_protected:Npn \__pdfmeta_xmp_sanitize:nN #1 #2
923 % #1 input string, #2 str with the output
924 {
925   \group_begin:
926   \text_declare_purify_equivalent:Nn \& {\tl_to_str:N & }
927   \text_declare_purify_equivalent:Nn \texttilde {\c_tilde_str}
928   \tl_set:Ne \l__pdfmeta_tmpa_tl { \text_purify:n {#1} }
929   \str_gset:Ne \g__pdfmeta_tmpa_str { \tl_to_str:N \l__pdfmeta_tmpa_tl }
930   \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {\&}{&}
931   \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {<}{&lt;}
932   \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {>}{&gt;}
933   \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {"}{&quot;}
934   \group_end:
935   \str_set_eq:NN #2 \g__pdfmeta_tmpa_str
936 }
937
938 \cs_generate_variant:Nn\__pdfmeta_xmp_sanitize:nN {VN}
```

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

4.4 Language handling

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

```
\l__pdfmeta_xmp_doclang_tl
\l__pdfmeta_xmp_metalang_tl
939 \tl_new:N \l__pdfmeta_xmp_doclang_tl
940 \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`

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

(End of definition for `\l__pdfmeta_xmp_lang_regex`.)

```
943 \cs_new_protected:Npn \__pdfmeta_xmp_lang_get:nNN #1 #2 #3
944 % #1 text, #2 tl var for lang match (or default), #3 tl var for text
945 {
946   \regex_extract_once:Nnn \l__pdfmeta_xmp_lang_regex {#1}\l__pdfmeta_tmpa_seq
947   \seq_if_empty:NTF \l__pdfmeta_tmpa_seq
948   {
949     \tl_set:Nn #2 \l__pdfmeta_xmp_metalang_tl
950     \tl_set:Nn #3 {#1}
951   }
```

```

952     {
953       \tl_set:Ne #2 {\seq_item:Nn\l__pdfmeta_tmpa_seq{2}}
954       \tl_set:Ne #3 {\seq_item:Nn\l__pdfmeta_tmpa_seq{3}}
955     }
956   }
957 \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`

```

958 \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.

```

959 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_chunk:n #1
960 {
961   \tl_gput_right:Ne\g__pdfmeta_xmp_packet_tl
962   {
963     \__pdfmeta_xmp_indent: \exp_not:n{#1}
964   }
965 }
966 \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.

```

967 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_chunk:nN #1 #2
968 {
969   \tl_put_right:Ne#2
970   {
971     \__pdfmeta_xmp_indent: \exp_not:n{#1}
972   }
973 }
974 \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.

```

975 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_open:nn #1 #2 %#1 prefix #2 name
976 {
977   \__pdfmeta_xmp_add_packet_chunk:n {<#1:#2>}
978   \__pdfmeta_xmp_incr_indent:
979 }
980 \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.

```
981 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_open_attr:nnn #1 #2 #3
982   % #1 prefix #2 name #3 attr
983   {
984     \_pdfmeta_xmp_add_packet_chunk:n {<#1:#2~#3>}
985     \_pdfmeta_xmp_incr_indent:
986   }
987 \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.

```
988 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_close:nn #1 #2 % #1 prefix #2: name
989   {
990     \_pdfmeta_xmp_decr_indent:
991     \_pdfmeta_xmp_add_packet_chunk:n {</#1:#2>}
992   }
```

(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.

```
993 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_line:nnn #1 #2 #3
994   % #1 prefix #2 name #3 content
995   {
996     \tl_if_blank:nF {#3}
997     {
998       \_pdfmeta_xmp_sanitize:nN {#3}\l_pdfmeta_tmpa_str
999       \_pdfmeta_xmp_add_packet_chunk:e {<#1:#2>\l_pdfmeta_tmpa_str</#1:#2>}
1000     }
1001   }
1002 \cs_generate_variant:Nn \_pdfmeta_xmp_add_packet_line:nnn {nne,nnV,nee}
```

(End of definition for _pdfmeta_xmp_add_packet_line:nnn.)

_pdfmeta_xmp_add_packet_line:nnnN This will produce a full line with open and closing xml and store it in the given tl-var. This allows to prebuild blocks and then to test if there are empty. The content is sanitized. We test if there is content to be able to suppress data which has not be set.

```
1003 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_line:nnnN #1 #2 #3 #4
1004   % #1 prefix #2 name #3 content #4 tl_var to prebuilt.
1005   {
1006     \tl_if_blank:nF {#3}
1007     {
1008       \_pdfmeta_xmp_sanitize:nN {#3}\l_pdfmeta_tmpa_str
1009       \_pdfmeta_xmp_add_packet_chunk:eN {<#1:#2>\l_pdfmeta_tmpa_str</#1:#2>} #4
1010     }
1011   }
1012 \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

```
1013 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_line_attr:nnnn #1 #2 #3 #4
1014   % #1 prefix #2 name #3 attribute #4 content
1015   {
1016     \tl_if_blank:nF {#4}
1017     {
1018       \__pdfmeta_xmp_sanitizе:nN {#4}\l__pdfmeta_tmpa_str
1019       \__pdfmeta_xmp_add_packet_chunk:e {<#1:#2-#3>\l__pdfmeta_tmpa_str</#1:#2>}
1020     }
1021   }
1022 \cs_generate_variant:Nn \__pdfmeta_xmp_add_packet_line_attr:nnnn {nnee,nneV}
```

(End of definition for __pdfmeta_xmp_add_packet_line_attr:nnnn.)

__pdfmeta_xmp_add_packet_line_default:nnnn

```
1023 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_line_default:nnnn #1 #2 #3 #4
1024   % #1 prefix #2 name #3 default #4 content
1025   {
1026     \tl_if_blank:nTF { #4 }
1027     {
1028       \tl_set:Nn \l__pdfmeta_tmpa_tl {#3}
1029     }
1030     {
1031       \tl_set:Nn \l__pdfmeta_tmpa_tl {#4}
1032     }
1033     \__pdfmeta_xmp_add_packet_line:nnV {#1}{#2}\l__pdfmeta_tmpa_tl
1034   }
1035 \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:

```
1036 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_list_simple:nnnn #1 #2 #3 #4
1037   % #1 prefix, #2 name, #3 type (Seq/Bag/Alt) #4 a clist
1038   {
1039     \clist_if_empty:nF { #4 }
1040     {
1041       \__pdfmeta_xmp_add_packet_open:nn {#1}{#2}
1042       \__pdfmeta_xmp_add_packet_open:nn {rdf}{#3}
1043       \clist_map_inline:nn {#4}
1044       {
1045         \__pdfmeta_xmp_add_packet_line:nnn
1046         {rdf}{li}{##1}
1047       }
1048       \__pdfmeta_xmp_add_packet_close:nn{rdf}{#3}
1049       \__pdfmeta_xmp_add_packet_close:nn {#1}{#2}
1050     }
1051   }
1052 \cs_generate_variant:Nn \__pdfmeta_xmp_add_packet_list_simple:nnnn {nnnV,nnne}
```

Here we check also for the language.

```

1053 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_list:nnnn #1 #2 #3 #4
1054   %#1 prefix, #2 name, #3 type (Seq/Bag/Alt) #4 a clist
1055   {
1056     \clist_if_empty:nF { #4 }
1057     {
1058       \__pdfmeta_xmp_add_packet_open:nn {#1}{#2}
1059       \__pdfmeta_xmp_add_packet_open:nn {rdf}{#3}
1060       \clist_map_inline:nn {#4}
1061       {
1062         \__pdfmeta_xmp_lang_get:nNN {##1}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl

```

change 2024-02-22. There should be if possible a x-default entry as some viewers need that. So if the language is equal to the main language we use that. This assumes that the user hasn't marked every entry as some other language! x-default has to be the first entry, see issue #92, so we have to go through the list twice.

```

1063         \tl_if_eq:eeT{\l__pdfmeta_tmpa_tl}{\l__pdfmeta_xmp_metalang_tl}
1064         {
1065           \__pdfmeta_xmp_add_packet_line_attr:nneV
1066             {rdf}{li}{xml:lang="x-default" }\l__pdfmeta_tmpb_tl
1067           \__pdfmeta_xmp_add_packet_line_attr:nneV
1068             {rdf}{li}{xml:lang="\l__pdfmeta_tmpa_tl" }\l__pdfmeta_tmpb_tl
1069         }
1070       }
1071     \clist_map_inline:nn {#4}
1072     {
1073       \__pdfmeta_xmp_lang_get:nNN {##1}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl
1074       \tl_if_eq:eeF{\l__pdfmeta_tmpa_tl}{\l__pdfmeta_xmp_metalang_tl}
1075       {
1076         \__pdfmeta_xmp_add_packet_line_attr:nneV
1077           {rdf}{li}{xml:lang="\l__pdfmeta_tmpa_tl" }\l__pdfmeta_tmpb_tl
1078       }
1079     }
1080     \__pdfmeta_xmp_add_packet_close:nn{rdf}{#3}
1081     \__pdfmeta_xmp_add_packet_close:nn {#1}{#2}
1082   }
1083 }
1084 \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.

```

1085 \cs_new_protected:Npn \__pdfmeta_xmp_build_packet:
1086   {

```

Get the main languages

```

1087   \tl_set:Ne \l__pdfmeta_xmp_doclang_tl {\GetDocumentProperties{document/lang}}
1088   \tl_set:Ne \l__pdfmeta_xmp_metalang_tl {\GetDocumentProperties{hyperref/pdfmetalang}}
1089   \tl_if_blank:VT \l__pdfmeta_xmp_metalang_tl
1090   { \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

```
1091 \__pdfmeta_xmp_build_iptc_data:N \l__pdfmeta_xmp_iptc_data_tl
1092 \tl_if_empty:NT \l__pdfmeta_xmp_iptc_data_tl
1093 {
1094   \seq_remove_all:Nn \l__pdfmeta_xmp_schema_seq { Iptc4xmpCore }
1095 }
```

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

```
1096 \__pdfmeta_xmp_add_packet_chunk:e
1097 {<?xpacket~begin="\__pdfmeta_xmp_generate_bom:"~id="W5MOMpCehiHzreSzNTczkc9d"?>}
1098 \__pdfmeta_xmp_add_packet_open:nn{x}{xmpmeta~xmlns:x="adobe:ns:meta/"}
1099 \__pdfmeta_xmp_add_packet_open:ne{rdf}
1100 {RDF~xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\c_hash_str"}
```

The rdf namespaces

```
1101 \__pdfmeta_xmp_add_packet_open_attr:nne
1102 {rdf}{Description}{rdf:about="" \g__pdfmeta_xmp_xmlns_tl}
```

The extensions

```
1103 \__pdfmeta_xmp_add_packet_open:nn{pdfaExtension}{schemas}
1104 \__pdfmeta_xmp_add_packet_open:nn {rdf}{Bag}
1105 \seq_map_inline:Nn \l__pdfmeta_xmp_schema_seq
1106 {
1107   \tl_use:c { g__pdfmeta_xmp_schema_##1_tl }
1108 }
1109 \__pdfmeta_xmp_add_packet_close:nn {rdf}{Bag}
1110 \__pdfmeta_xmp_add_packet_close:nn {pdfaExtension}{schemas}
```

Now starts the part with the data.

```
1111 % data
1112 \__pdfmeta_xmp_build_pdf:
1113 \__pdfmeta_xmp_build_xmpRights:
1114 \__pdfmeta_xmp_build_standards: %pdfaid, pdfxid, pdfuaid
1115 \__pdfmeta_xmp_build_pdfd:
1116 \__pdfmeta_xmp_build_dc:
1117 \__pdfmeta_xmp_build_photoshop:
1118 \__pdfmeta_xmp_build_xmp:
1119 \__pdfmeta_xmp_build_xmpMM:
1120 \__pdfmeta_xmp_build_prism:
1121 \__pdfmeta_xmp_build_iptc:
1122 \__pdfmeta_xmp_build_tdm:
1123 \__pdfmeta_xmp_build_user: %user additions
1124 % end
1125 \__pdfmeta_xmp_add_packet_close:nn {rdf}{Description}
1126 \__pdfmeta_xmp_add_packet_close:nn {rdf}{RDF}
1127 \__pdfmeta_xmp_add_packet_close:nn {x}{xmpmeta}
1128 \int_set:Nn \l__pdfmeta_xmp_indent_int{20}
```

```

1129 \prg_replicate:nn{10}{\__pdfmeta_xmp_add_packet_chunk:n {}}
1130 \int_zero:N \l__pdfmeta_xmp_indent_int
1131 \__pdfmeta_xmp_add_packet_chunk:n {<?xpacket~end="w"?>}
1132 }

```

(End of definition for __pdfmeta_xmp_build_packet:.)

4.6 Building the chunks: rdf namespaces

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

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

```

1133 \str_new:N \g__pdfmeta_xmp_xmlns_tl
1134 \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

```

1135 \cs_new_protected:Npn \__pdfmeta_xmp_xmlns_new:nn #1 #2
1136 {
1137   \prop_gput:Nnn \g__pdfmeta_xmp_xmlns_prop {#1}{#2}
1138   \tl_gput_right:Ne \g__pdfmeta_xmp_xmlns_tl
1139     {
1140       \__pdfmeta_xmp_indent:n{4} xmlns:\exp_not:n{#1="#2"}
1141     }
1142 }

```

(End of definition for __pdfmeta_xmp_xmlns_new:nn.)

Now we fill the data. The list is more or less the same as in hyperxmp The pdfxid entry is only added if an X standard is used, see issue #50 and the schema below.

```

1143 \__pdfmeta_xmp_xmlns_new:nn {pdf} {http://ns.adobe.com/pdf/1.3/}
1144 \__pdfmeta_xmp_xmlns_new:nn {xmpRights}{http://ns.adobe.com/xap/1.0/rights/}
1145 \__pdfmeta_xmp_xmlns_new:nn {dc} {http://purl.org/dc/elements/1.1/}
1146 \__pdfmeta_xmp_xmlns_new:nn {photoshop}{http://ns.adobe.com/photoshop/1.0/}
1147 \__pdfmeta_xmp_xmlns_new:nn {xmp} {http://ns.adobe.com/xap/1.0/}
1148 \__pdfmeta_xmp_xmlns_new:nn {xmpMM} {http://ns.adobe.com/xap/1.0/mm/}
1149 \__pdfmeta_xmp_xmlns_new:nn {stEvt}
1150 {http://ns.adobe.com/xap/1.0/sType/ResourceEvent\c_hash_str}
1151 \__pdfmeta_xmp_xmlns_new:nn {pdfaid} {http://www.aiim.org/pdfa/ns/id/}
1152 \__pdfmeta_xmp_xmlns_new:nn {pdfuaid} {http://www.aiim.org/pdfua/ns/id/}
1153 \__pdfmeta_xmp_xmlns_new:nn {pdfx} {http://ns.adobe.com/pdfx/1.3/}
1154 %\__pdfmeta_xmp_xmlns_new:nn {pdfxid} {http://www.npes.org/pdfx/ns/id/}
1155 \__pdfmeta_xmp_xmlns_new:nn {prism} {http://prismstandard.org/namespaces/basic/3.0/}
1156 %\__pdfmeta_xmp_xmlns_new:nn {jav} {http://www.niso.org/schemas/jav/1.0/}
1157 %\__pdfmeta_xmp_xmlns_new:nn {xmpTPg} {http://ns.adobe.com/xap/1.0/t/pg/}
1158 \__pdfmeta_xmp_xmlns_new:nn {stFnt} {http://ns.adobe.com/xap/1.0/sType/Font\c_hash_str}
1159 \__pdfmeta_xmp_xmlns_new:nn {Iptc4xmpCore}{http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}

```

```

1160 \__pdfmeta_xmp_xmlns_new:nn {pdfaExtension}{http://www.aiim.org/pdfa/ns/extension/}
1161 \__pdfmeta_xmp_xmlns_new:nn {pdfaSchema}{http://www.aiim.org/pdfa/ns/schema\c_hash_str}
1162 \__pdfmeta_xmp_xmlns_new:nn {pdfaProperty}{http://www.aiim.org/pdfa/ns/property\c_hash_str}
1163 \__pdfmeta_xmp_xmlns_new:nn {pdfaType} {http://www.aiim.org/pdfa/ns/type\c_hash_str}
1164 \__pdfmeta_xmp_xmlns_new:nn {pdfaField}{http://www.aiim.org/pdfa/ns/field\c_hash_str}
1165 \__pdfmeta_xmp_xmlns_new:nn {tdm}{http://www.w3.org/ns/tdmrep/}

```

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

```

1166 \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.

```

1167 \cs_new_protected:Npn \__pdfmeta_xmp_schema_new:nnn #1 #2 #3
1168   {%#1 name #2 prefix, #3 text
1169   {
1170     \tl_if_exist:cTF { g__pdfmeta_xmp_schema_#2_tl }
1171     {
1172       \msg_warning:nnnn{pdfmeta}{xmp-defined}{schema}{#2}
1173     }
1174     {
1175       \seq_put_right:Nn \l__pdfmeta_xmp_schema_seq { #2 }
1176       \tl_new:c { g__pdfmeta_xmp_schema_#2_tl }
1177       \tl_new:c { g__pdfmeta_xmp_schema_#2_properties_tl }
1178       \tl_gput_right:cn { g__pdfmeta_xmp_schema_#2_tl }
1179       {
1180         \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1181         \__pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{schema}{#1}
1182         \__pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{prefix}{#2}
1183         \__pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{namespaceURI}{#3}
1184         \__pdfmeta_xmp_add_packet_open:nn {pdfaSchema}{property}
1185         \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1186         \tl_use:c { g__pdfmeta_xmp_schema_#2_properties_tl }
1187         \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1188         \__pdfmeta_xmp_add_packet_close:nn {pdfaSchema}{property}
1189         \cs_if_exist_use:c {__pdfmeta_xmp_schema_#2_additions:}
1190         \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1191       }
1192     }
1193   }

```

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

`_pdfmeta_xmp_property_new:nnnn` This adds a property to a schema.

```
1194 \prop_new:N\g__pdfmeta_xmp_schema_property_prop
1195 \cs_new_protected:Npn \_pdfmeta_xmp_property_new:nnnn #1 #2 #3 #4 #5 %
1196   % #1 schema #2 name, #3 type, #4 category #5 description
1197   {
1198     \tl_if_exist:cTF { g__pdfmeta_xmp_schema_#1_properties_tl }
1199     {
1200       \prop_get:NeNF \g__pdfmeta_xmp_schema_property_prop {#1:#2}\l__pdfmeta_tmpa_tl
1201       {
1202         \prop_gput:Nee \g__pdfmeta_xmp_schema_property_prop {#1:#2}{#3}
1203         \tl_gput_right:cn { g__pdfmeta_xmp_schema_#1_properties_tl }
1204         {
1205           \_pdfmeta_xmp_add_packet_open:nn {rdf}{li}{rdf:parseType="Resource"}
1206           \_pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{name}{#2}
1207           \_pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{valueType}{#3}
1208           \_pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{category}{#4}
1209           \_pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{description}{#5}
1210           \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1211         }
1212       }
1213     }
1214     {
1215       \msg_warning:nnnn{pdfmeta}{xmp-undefined}{schema}{#1}
1216     }
1217   }
```

(End of definition for `_pdfmeta_xmp_property_new:nnnn`.)

`_pdfmeta_xmp_add_packet_field:nnn` This adds a field to a schema.

```
1218 \cs_new_protected:Npn \_pdfmeta_xmp_add_packet_field:nnn #1 #2 #3 %
1219   % #1 name #2 valuetype #3 description
1220   {
1221     \_pdfmeta_xmp_add_packet_open_attr:nnn {rdf}{li}{rdf:parseType="Resource"}
1222     \_pdfmeta_xmp_add_packet_line:nnn {pdfaField}{name}{#1}
1223     \_pdfmeta_xmp_add_packet_line:nnn {pdfaField}{valueType}{#2}
1224     \_pdfmeta_xmp_add_packet_line:nnn {pdfaField}{description}{#3}
1225     \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1226   }
```

(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.

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.

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.

```

1227 \_pdfmeta_xmp_schema_new:nnn
1228   {XMP~Media~Management~Schema}
1229   {xmpMM}
1230   {http://ns.adobe.com/xap/1.0/mm/}
1231 \_pdfmeta_xmp_property_new:nnnnn
1232   {xmpMM}
1233   {OriginalDocumentID}
1234   {URI}
1235   {internal}
1236   {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)

```

1237 \_pdfmeta_xmp_schema_new:nnn
1238   {PDF/A~Identification~Schema}
1239   {pdfaid}
1240   {http://www.aiim.org/pdfa/ns/id/}
1241 \_pdfmeta_xmp_property_new:nnnnn
1242   {pdfaid}
1243   {year}
1244   {Integer}
1245   {internal}
1246   {Year~of~standard}
1247 \_pdfmeta_xmp_property_new:nnnnn
1248   {pdfaid}
1249   {rev}
1250   {Integer}
1251   {internal}
1252   {Revision~year~of~standard}

```

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

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

pdfuaid-(schema)

```

1253 \_pdfmeta_xmp_schema_new:nnn
1254   {PDF/UA~Universal~Accessibility~Schema}
1255   {pdfuaid}
1256   {http://www.aiim.org/pdfua/ns/id/}
1257 \_pdfmeta_xmp_property_new:nnnnn
1258   {pdfuaid}
1259   {part}
1260   {Integer}

```

```

1261     {internal}
1262     {Part-of-ISO-14289-standard}
1263 \_pdfmeta_xmp_property_new:nnnnn
1264     {pdfuaid}
1265     {rev}
1266     {Integer}
1267     {internal}
1268     {Revision-of-ISO-14289-standard}

```

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

pdfx According to [1] not an allowed schema, but it seems to validate and allow to set the pdf/X version, hyperxmp declares here the properties GTS_PDFXVersion and GTS_PDFXConformance. Ignored as only relevant for older pdf/X version not supported by the pdfmanagement.

pdfxid we set this so that we can add the pdf/X version for pdf/X-4 and higher. This is only set if a pdf/X standard is used, see issue #50

pdfxid-(schema)

```

1269 \cs_new_protected:Npn \_pdfmeta_xmp_add_pdfxid:
1270 {
1271   \_pdfmeta_xmp_xmlns_new:nn {pdfxid} {http://www.npes.org/pdfx/ns/id/}
1272   \_pdfmeta_xmp_schema_new:nnn
1273     {PDF/X-ID-Schema}
1274     {pdfxid}
1275     {http://www.npes.org/pdfx/ns/id/}
1276   \_pdfmeta_xmp_property_new:nnnnn
1277     {pdfxid}
1278     {GTS_PDFXVersion}
1279     {Text}
1280     {internal}
1281     {ID-of-PDF/X-standard}
1282 }

```

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

prism-(schema)

```

1283 \_pdfmeta_xmp_schema_new:nnn
1284   {PRISM-Basic-Metadata}
1285   {prism}
1286   {http://prismstandard.org/namespaces/basic/3.0/}
1287 \_pdfmeta_xmp_property_new:nnnnn
1288   {prism}
1289   {complianceProfile}
1290   {Text}
1291   {internal}
1292   {PRISM-specification-compliance-profile-to-which-this-document-adheres}
1293 \_pdfmeta_xmp_property_new:nnnnn
1294   {prism}
1295   {publicationName}
1296   {Text}

```

```

1297 {external}
1298 {Publication-name}
1299 \_pdfmeta_xmp_property_new:nnnnn
1300 {prism}
1301 {aggregationType}
1302 {Text}
1303 {external}
1304 {Publication-type}
1305 \_pdfmeta_xmp_property_new:nnnnn
1306 {prism}
1307 {bookEdition}
1308 {Text}
1309 {external}
1310 {Edition-of-the-book-in-which-the-document-was-published}
1311 \_pdfmeta_xmp_property_new:nnnnn
1312 {prism}
1313 {volume}
1314 {Text}
1315 {external}
1316 {Publication-volume-number}
1317 \_pdfmeta_xmp_property_new:nnnnn
1318 {prism}
1319 {number}
1320 {Text}
1321 {external}
1322 {Publication-issue-number-within-a-volume}
1323 \_pdfmeta_xmp_property_new:nnnnn
1324 {prism}
1325 {pageRange}
1326 {Text}
1327 {external}
1328 {Page-range-for-the-document-within-the-print-version-of-its-publication}
1329 \_pdfmeta_xmp_property_new:nnnnn
1330 {prism}
1331 {issn}
1332 {Text}
1333 {external}
1334 {ISSN-for-the-printed-publication-in-which-the-document-was-published}
1335 \_pdfmeta_xmp_property_new:nnnnn
1336 {prism}
1337 {eIssn}
1338 {Text}
1339 {external}
1340 {ISSN-for-the-electronic-publication-in-which-the-document-was-published}
1341 \_pdfmeta_xmp_property_new:nnnnn
1342 {prism}
1343 {isbn}
1344 {Text}
1345 {external}
1346 {ISBN-for-the-publication-in-which-the-document-was-published}
1347 \_pdfmeta_xmp_property_new:nnnnn
1348 {prism}
1349 {doi}
1350 {Text}

```

```

1351 {external}
1352 {Digital-Object-Identifier-for-the-document}
1353 \_pdfmeta_xmp_property_new:nnnnn
1354 {prism}
1355 {url}
1356 {URL}
1357 {external}
1358 {URL-at-which-the-document-can-be-found}
1359 \_pdfmeta_xmp_property_new:nnnnn
1360 {prism}
1361 {byteCount}
1362 {Integer}
1363 {internal}
1364 {Approximate-file-size-in-octets}
1365 \_pdfmeta_xmp_property_new:nnnnn
1366 {prism}
1367 {pageCount}
1368 {Integer}
1369 {internal}
1370 {Number-of-pages-in-the-print-version-of-the-document}
1371 \_pdfmeta_xmp_property_new:nnnnn
1372 {prism}
1373 {subtitle}
1374 {Text}
1375 {external}
1376 {Document's-subtitle}

```

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

iptc (schema)

```

1377 \_pdfmeta_xmp_schema_new:nnn
1378 {IPTC-Core-Schema}
1379 {Iptc4xmpCore}
1380 {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
1381 \_pdfmeta_xmp_property_new:nnnnn
1382 {Iptc4xmpCore}
1383 {CreatorContactInfo}
1384 {ContactInfo}
1385 {external}
1386 {Document-creator's-contact-information}
1387 \cs_new_protected:cpn { __pdfmeta_xmp_schema_Iptc4xmpCore_additions: }
1388 {
1389   \_pdfmeta_xmp_add_packet_open:nn{pdfaSchema}{valueType}
1390   \_pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1391   \_pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1392   \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{ContactInfo}
1393   \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
1394   {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
1395   \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{Iptc4xmpCore}
1396   \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
1397   {Basic-set-of-information-to-get-in-contact-with-a-person}
1398   \_pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
1399   \_pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1400   \_pdfmeta_xmp_add_packet_field:nnn{CiAdrCity}{Text}

```

```

1401         {Contact~information~city}
1402     \_pdfmeta_xmp_add_packet_field:nnn{CiAdrCtry}{Text}
1403         {Contact~information~country}
1404     \_pdfmeta_xmp_add_packet_field:nnn{CiAdrExtadr}{Text}
1405         {Contact~information~address}
1406     \_pdfmeta_xmp_add_packet_field:nnn{CiAdrPcode}{Text}
1407         {Contact~information~local~postal~code}
1408     \_pdfmeta_xmp_add_packet_field:nnn{CiAdrRegion}{Text}
1409         {Contact~information~regional~information~such~as~state~or~province}
1410     \_pdfmeta_xmp_add_packet_field:nnn{CiEmailWork}{Text}
1411         {Contact~information~email~address(es)}
1412     \_pdfmeta_xmp_add_packet_field:nnn{CiTelWork}{Text}
1413         {Contact~information~telephone~number(s)}
1414     \_pdfmeta_xmp_add_packet_field:nnn{CiUrlWork}{Text}
1415         {Contact~information~Web~URL(s)}
1416     \_pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1417     \_pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
1418     \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1419     \_pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1420     \_pdfmeta_xmp_add_packet_close:nn{pdfaSchema}{valueType}
1421 }

```

(End of definition for iptc (schema).)

jav : currently ignored

tdmrep (schema)

```

1422 \_pdfmeta_xmp_schema_new:nnn
1423     {TDMRep}
1424     {tdm}
1425     {http://www.w3.org/ns/tdmrep/}
1426 \_pdfmeta_xmp_property_new:nnnnn
1427     {tdm}
1428     {reservation}
1429     {Closed-Choice-of-Integer}
1430     {internal}
1431     {TDM-rights-are-reserved-(1)-or-not-reserved-(0)}
1432 \_pdfmeta_xmp_property_new:nnnnn
1433     {tdm}
1434     {policy}
1435     {URI}
1436     {internal}
1437     {URL~pointing~to~a~TDM~Policy~set~by~the~rightsholder}

```

(End of definition for tdmrep (schema).)

declarations The PDF Declarations mechanism allows creation and editing software to declare, via a PDF Declaration, a PDF file to be in conformance with a 3rd party specification or profile that may not be related to PDF technology. Their specification is for example described in <https://pdfa.org/wp-content/uploads/2019/09/PDF-Declarations.pdf>.

If declarations are added to the XMP-metadata they need (for pdf/A compliance) a schema declaration. We do not add it by default but define here a command to

enable it. (This can be done in the document preamble as xmp is built only at the end.)

```

1438 \cs_new_protected:Npn \__pdfmeta_xmp_schema_enable_pdfd:
1439 {
1440   \__pdfmeta_xmp_xmlns_new:nn {pdfd}{http://pdfa.org/declarations/}
1441   \__pdfmeta_xmp_schema_new:nnn
1442     {PDF~Declarations~Schema}
1443     {pdfd}
1444     {http://pdfa.org/declarations/}
1445   \__pdfmeta_xmp_property_new:nnnnn
1446     {pdfd}
1447     {declarations}
1448     {Bag~declaration}
1449     {external}
1450     {An~unordered~array~of~PDF~Declaration~entries,~where~each~PDF~Declaration~representing

```

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

```

1451   \cs_new_protected:cpn { __pdfmeta_xmp_schema_pdfd_additions: }
1452   {
1453     \__pdfmeta_xmp_add_packet_open:nn{pdfaSchema}{valueType}
1454     \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1455     \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1456     \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{claim}
1457     \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
1458       {http://pdfa.org/declarations/}
1459     \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{pdfd}
1460     \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
1461       {A~structure~describing~properties~of~an~individual~claim.}
1462     \__pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
1463     \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1464     \__pdfmeta_xmp_add_packet_field:nnn{claimReport}{Text}
1465       {A~URL~to~a~report~containing~details~of~the~specific~conformance~claim.}
1466     \__pdfmeta_xmp_add_packet_field:nnn{claimCredentials}{Text}
1467       {The~claimant's~credentials.}
1468     \__pdfmeta_xmp_add_packet_field:nnn{claimDate}{Text}
1469       {A~date~identifying~when~the~claim~was~made.}
1470     \__pdfmeta_xmp_add_packet_field:nnn{claimBy}{Text}
1471       {The~name~of~the~organization~and/or~individual~and/or~software~making~the
1472     \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1473     \__pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
1474     \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1475     \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1476     \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{declaration}
1477     \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
1478       {http://pdfa.org/declarations/}
1479     \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{pdfd}
1480     \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
1481       {A~structure~describing~a~single~PDF~ Declaration~asserting~conformance~with
identified~standard~or~ profile.}
1482     \__pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
1483     \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1484     \__pdfmeta_xmp_add_packet_field:nnn{conformsTo}{Text}

```

```

1485         {A~property~containing~a~URI~specifying~the~standard~or~profile~by~the~PDF
1486         \_pdfmeta_xmp_add_packet_field:nnn{claimData}{Bag~claim}
1487         {An~unordered~array~of~claim~data,~where~each~claim~identifies~the~nature~of
1488         \_pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1489         \_pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
1490         \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1491         \_pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1492         \_pdfmeta_xmp_add_packet_close:nn{pdfaSchema}{valueType}
1493     }

```

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

```

1494     \cs_gset_eq:NN \_pdfmeta_xmp_schema_enable_pdfd: \prg_do_nothing:
1495 }

```

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
  PDFversion1496 \cs_new_protected:Npn \_pdfmeta_xmp_build_pdf:
1497   {

```

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

```

1498     \_pdfmeta_xmp_add_packet_line_default:nnee
1499     {pdf}{Producer}
1500     {\c_sys_engine_exec_str-\c_sys_engine_version_str}
1501     {\GetDocumentProperties{hyperref/pdfproducer}}

```

Now the PDF version

```

1502     \_pdfmeta_xmp_add_packet_line:nne{pdf}{PDFVersion}{\pdf_version:}
1503 }

```

(End of definition for _pdfmeta_xmp_build_pdf:, Producer/pdfproducer, and PDFversion.)

4.8.2 xmp

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

```

\_pdfmeta_xmp_build_xmp:
  CreatorTool/pdfcreator
  BaseUrl/baseurl1504 \cs_new_protected:Npn \_pdfmeta_xmp_build_xmp:
1505   {

```

The creator

```

1506 \__pdfmeta_xmp_add_packet_line_default:nnee
1507 {xmp}{CreatorTool}
1508 {LaTeX}
1509 { \GetDocumentProperties{hyperref/pdfcreator} }

```

The baseurl

```

1510 \__pdfmeta_xmp_add_packet_line_default:nnee
1511 {xmp}{BaseURL}{}
1512 { \GetDocumentProperties{hyperref/baseurl} }

```

CreationDate

```

1513 \__pdfmeta_xmp_date_get:nNN
1514 {document/creationdate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1515 \__pdfmeta_xmp_add_packet_line:nne{xmp}{CreateDate}{\__pdfmeta_xmp_print_date:N\l__pdfmeta_t
1516 \pdfmanagement_add:nne{Info}{CreateDate}{(\l__pdfmeta_tmpa_tl)}

```

ModifyDate

```

1517 \__pdfmeta_xmp_date_get:nNN
1518 {document/moddate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1519 \__pdfmeta_xmp_add_packet_line:nne{xmp}{ModifyDate}{\__pdfmeta_xmp_print_date:N\l__pdfmeta_t
1520 \pdfmanagement_add:nne{Info}{ModDate}{(\l__pdfmeta_tmpa_tl)}

```

MetadataDate

```

1521 \__pdfmeta_xmp_date_get:nNN
1522 {hyperref/pdfmetadate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1523 \__pdfmeta_xmp_add_packet_line:nne{xmp}{MetadataDate}{\__pdfmeta_xmp_print_date:N\l__pdfmeta_t
1524 }

```

(End of definition for __pdfmeta_xmp_build_xmp:, CreatorTool/pdfcreator, and BaseUrl/baseurl.)

4.8.3 Standards

The metadata for standards are taken from the pdfstandard key of the document/metadata family. 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:

```

1525 \cs_new_protected:Npn \__pdfmeta_xmp_build_standards:
1526 {
1527 \__pdfmeta_xmp_add_packet_line:nne {pdfaid}{part}{\pdfmeta_standard_item:n{level}}
1528 \__pdfmeta_xmp_add_packet_line:nne
1529 {pdfaid}{conformance}{\pdfmeta_standard_item:n{conformance}}
1530 \int_compare:nNnTF {0\pdfmeta_standard_item:n{level}}<{4}
1531 {\__pdfmeta_xmp_add_packet_line:nne {pdfaid}{year} {\pdfmeta_standard_item:n{year}}}
1532 {\__pdfmeta_xmp_add_packet_line:nne {pdfaid}{rev} {\pdfmeta_standard_item:n{year}}}
1533 \__pdfmeta_xmp_add_packet_line:nne

```

```

1534     {pdfxid}{GTS_PDFXVersion}{\GetDocumentProperties{document/pdfstandard-X}}
1535 \pdfmanagement_get_documentproperties:nNT {document/pdfstandard-UA}\l__pdfmeta_tmpa_tl
1536 {
1537   \__pdfmeta_xmp_add_packet_line:nne
1538   {pdfuaid}{part}{\exp_last_unbraced:No\use_i:nn \l__pdfmeta_tmpa_tl}
1539   \__pdfmeta_xmp_add_packet_line:nne
1540   {pdfuaid}{rev}{\exp_last_unbraced:No\use_ii:nn \l__pdfmeta_tmpa_tl}
1541 }
1542 }

```

(End of definition for __pdfmeta_xmp_build_standards:.)

4.9 Declarations

See <https://pdfa.org/wp-content/uploads/2019/09/PDF-Declarations.pdf>

\g__pdfmeta_xmp_pdfd_data_prop This holds the data for declarations.

```

1543 \prop_new:N \g__pdfmeta_xmp_pdfd_data_prop

```

(End of definition for \g__pdfmeta_xmp_pdfd_data_prop.)

the main building command used in the xmp generation

__pdfmeta_xmp_build_pdfd:

```

1544 \cs_new_protected:Npn \__pdfmeta_xmp_build_pdfd:
1545 {
1546   \prop_if_empty:NF \g__pdfmeta_xmp_pdfd_data_prop
1547   {
1548     \__pdfmeta_xmp_add_packet_open:nn{pdfd}{declarations}
1549     \__pdfmeta_xmp_add_packet_open:nn{rdf}{Bag}
1550     \prop_map_inline:Nn \g__pdfmeta_xmp_pdfd_data_prop
1551     {
1552       \__pdfmeta_xmp_build_pdfd_claim:nn{##1}{##2}
1553     }
1554     \__pdfmeta_xmp_add_packet_close:nn{rdf}{Bag}
1555     \__pdfmeta_xmp_add_packet_close:nn{pdfd}{declarations}
1556   }
1557 }

```

(End of definition for __pdfmeta_xmp_build_pdfd:.)

__pdfmeta_xmp_build_pdfd_claim:nn This build the xml for one claim. If there is no claimData only the conformsTo is output.

```

1558 \cs_new_protected:Npn \__pdfmeta_xmp_build_pdfd_claim:nn #1#2
1559 {
1600   \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1561   \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{conformsTo}{#1}
1562   \tl_if_empty:nF {#2}
1563   {
1564     \__pdfmeta_xmp_add_packet_open:nn{pdfd}{claimData}
1565     \__pdfmeta_xmp_add_packet_open:nn{rdf}{Bag}
1566     #2
1567     \__pdfmeta_xmp_add_packet_close:nn{rdf}{Bag}

```

```

1568         \_pdfmeta_xmp_add_packet_close:nn{pdfd}{claimData}
1569     }
1570     \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1571 }

```

(End of definition for _pdfmeta_xmp_build_pdfd_claim:nn.)

4.10 Photoshop

_pdfmeta_xmp_build_photoshop:

```

1572 \cs_new_protected:Npn \_pdfmeta_xmp_build_photoshop:
1573 {

pdfauthortitle/photoshop:AuthorsPosition

1574     \_pdfmeta_xmp_add_packet_line:nne{photoshop}{AuthorsPosition}
1575     { \GetDocumentProperties{hyperref/pdfauthortitle} }

pdfcaptionwriter/photoshop:CaptionWriter

1576     \_pdfmeta_xmp_add_packet_line:nne{photoshop}{CaptionWriter}
1577     { \GetDocumentProperties{hyperref/pdfcaptionwriter} }

1578 }

```

(End of definition for _pdfmeta_xmp_build_photoshop:.)

4.11 XMP Media Management

_pdfmeta_xmp_build_xmpMM:

```

1579 \cs_new_protected:Npn \_pdfmeta_xmp_build_xmpMM:
1580 {

pdfdocumentid / xmpMM:DocumentID

1581     \str_set:Ne\l__pdfmeta_tmpa_str {\GetDocumentProperties{hyperref/pdfdocumentid}}
1582     \str_if_empty:NT \l__pdfmeta_tmpa_str
1583     {
1584         \_pdfmeta_xmp_create_uuid:nN
1585         {\jobname\GetDocumentProperties{hyperref/pdftitle}}
1586         \l__pdfmeta_tmpa_str
1587     }
1588     \_pdfmeta_xmp_add_packet_line:nnV{xmpMM}{DocumentID}
1589     \l__pdfmeta_tmpa_str

pdfinstanceid / xmpMM:InstanceID

```

```

1590 \str_set:Ne\l__pdfmeta_tmpa_str {\GetDocumentProperties{hyperref/pdfinstanceid}}
1591 \str_if_empty:NT \l__pdfmeta_tmpa_str
1592 {
1593   \__pdfmeta_xmp_create_uuid:nN
1594     {\jobname\l__pdfmeta_xmp_currentdate_tl}
1595     \l__pdfmeta_tmpa_str
1596 }
1597 \__pdfmeta_xmp_add_packet_line:nnV{xmpMM}{InstanceID}
1598 \l__pdfmeta_tmpa_str

```

pdfversionid/xmpMM:VersionID

```

1599 \__pdfmeta_xmp_add_packet_line:nne{xmpMM}{VersionID}
1600 { \GetDocumentProperties{hyperref/pdfversionid} }

```

pdfrendition/xmpMM:RenditionClass

```

1601 \__pdfmeta_xmp_add_packet_line:nne{xmpMM}{RenditionClass}
1602 { \GetDocumentProperties{hyperref/pdfrendition} }

```

```

1603 }

```

(End of definition for __pdfmeta_xmp_build_xmpMM:.)

4.12 Rest of dublin Core data

```

\__pdfmeta_xmp_build_dc:
  dc:creator/pdfauthor
  dc:subject/pdfkeywords1604 \cs_new_protected:Npn \__pdfmeta_xmp_build_dc:
    dc:type/pdftype1605 {
dc:publisher/pdfpublisher
dc:description/pdfsubject pdfauthor/dc:creator
  dc:language/lang/pdflang
dc:identifier/pdfidentifier1606 \__pdfmeta_xmp_add_packet_list_simple:nne {dc}{creator}{Seq}
  { \GetDocumentProperties{hyperref/pdfauthor} }
photoshop:AuthorsPosition/pdfauthortitle1607
  \int_compare:nNnT {0\pdfmeta_standard_item:n{level}}={1}
photoshop:CaptionWriter/pdfcaptionwriter1608
  { \pdfmanagement_remove:nn{Info}{Author} }
1609

```

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

```

1610 \__pdfmeta_xmp_add_packet_list:nne {dc}{title}{Alt}
1611 { \GetDocumentProperties{hyperref/pdftitle} }

```

pdfkeywords/dc:subject

```

1612 \__pdfmeta_xmp_add_packet_list_simple:nne {dc}{subject}{Bag}
1613 { \GetDocumentProperties{hyperref/pdfkeywords} }
1614 \int_compare:nNnT {0\pdfmeta_standard_item:n{level}}={1}
1615 { \pdfmanagement_remove:nn{Info}{Keywords} }

```

pdftype/dc:type

```

1616 \pdfmanagement_get_documentproperties:nNTF { hyperref/pdftype } \l__pdfmeta_tmpa_tl
1617 {
1618   \__pdfmeta_xmp_add_packet_list_simple:nnnV {dc}{type}{Bag}\l__pdfmeta_tmpa_tl
1619 }
1620 {
1621   \__pdfmeta_xmp_add_packet_list_simple:nnnn {dc}{type}{Bag}{Text}
1622 }

```

pdfpublisher/dc:publisher

```

1623 \__pdfmeta_xmp_add_packet_list_simple:nne {dc}{publisher}{Bag}
1624 { \GetDocumentProperties{hyperref/pdfpublisher} }

```

pdfsubject/dc:description

```

1625 \__pdfmeta_xmp_add_packet_list:nne
1626 {dc}{description}{Alt}
1627 { \GetDocumentProperties{hyperref/pdfsubject} }

```

lang/pdflang/dc:language

```

1628 \__pdfmeta_xmp_add_packet_list_simple:nnnV
1629 {dc}{language}{Bag}\l__pdfmeta_xmp_doclang_tl

```

pdfidentifier/dc:identifier

```

1630 \__pdfmeta_xmp_add_packet_line:nne{dc}{identifier}
1631 { \GetDocumentProperties{hyperref/pdfidentifier} }

```

pdfdate/dc:date

```

1632 \__pdfmeta_xmp_date_get:nNN {hyperref/pdfdate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1633 \__pdfmeta_xmp_add_packet_list_simple:nne
1634 {dc}{date}{Seq}{\__pdfmeta_xmp_print_date:N\l__pdfmeta_tmpa_seq}

```

The file format

```

1635 \__pdfmeta_xmp_add_packet_line:nnn{dc}{format}{application/pdf}

```

The source

```

1636 \__pdfmeta_xmp_add_packet_line_default:nnee
1637 {dc}{source}
1638 { \c_sys_jobname_str.tex }
1639 { \GetDocumentProperties{hyperref/pdfsource} }

```

```

1640 \__pdfmeta_xmp_add_packet_list:nne{dc}{rights}{Alt}
1641 { \GetDocumentProperties{hyperref/pdfcopyright} }

```

```

1642 }

```

(End of definition for __pdfmeta_xmp_build_dc: and others.)

4.13 xmpRights

_pdfmeta_xmp_build_xmpRights:

```
1643 \cs_new_protected:Npn \_pdfmeta_xmp_build_xmpRights:
1644 {
1645   \_pdfmeta_xmp_add_packet_line:nne
1646     {xmpRights}
1647     {WebStatement}
1648     {\GetDocumentProperties{hyperref/pdflicenseurl}}
1649   \_pdfmeta_xmp_add_packet_line:nne
1650     {xmpRights}
1651     {Marked}
1652     {
1653       \str_case:en {\GetDocumentProperties{document/copyright}}
1654       {
1655         {true}{True}
1656         {false}{False}
1657       }
1658     }
1659 }
```

(End of definition for _pdfmeta_xmp_build_xmpRights:.)

4.14 IPTC

We want the block and also the resources only if they are actually used. So we pack them first in a local variable

\l__pdfmeta_xmp_iptc_data_tl

```
1660 \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

```
1661 \cs_new_protected:Npn \_pdfmeta_xmp_build_iptc_data:N #1
1662 {
1663   \tl_clear:N #1
1664   \_pdfmeta_xmp_incr_indent:\_pdfmeta_xmp_incr_indent:\_pdfmeta_xmp_incr_indent:\_pdfmeta
1665   \_pdfmeta_xmp_add_packet_line:nneN
1666     {Iptc4xmpCore}{CiAdrExtadr}
1667     {\GetDocumentProperties{hyperref/pdfcontactaddress}}
1668     #1
1669   \_pdfmeta_xmp_add_packet_line:nneN
1670     {Iptc4xmpCore}{CiAdrCity}
1671     {\GetDocumentProperties{hyperref/pdfcontactcity}}
1672     #1
1673   \_pdfmeta_xmp_add_packet_line:nneN
1674     {Iptc4xmpCore}{CiAdrPcode}
1675     {\GetDocumentProperties{hyperref/pdfcontactpostcode}}
1676     #1
1677   \_pdfmeta_xmp_add_packet_line:nneN
1678     {Iptc4xmpCore}{CiAdrCtry}
```

```

1679     {\GetDocumentProperties{hyperref/pdfcontactcountry}}
1680     #1
1681     \__pdfmeta_xmp_add_packet_line:nneN
1682     {Iptc4xmpCore}{CiTelWork}
1683     {\GetDocumentProperties{hyperref/pdfcontactphone}}
1684     #1
1685     \__pdfmeta_xmp_add_packet_line:nneN
1686     {Iptc4xmpCore}{CiEmailWork}
1687     {\GetDocumentProperties{hyperref/pdfcontactemail}}
1688     #1
1689     \__pdfmeta_xmp_add_packet_line:nneN
1690     {Iptc4xmpCore}{CiUrlWork}
1691     {\GetDocumentProperties{hyperref/pdfcontacturl}}
1692     #1
1693     \__pdfmeta_xmp_decr_indent:\__pdfmeta_xmp_decr_indent:\__pdfmeta_xmp_decr_indent:\__pdfmeta
1694 }

```

(End of definition for __pdfmeta_xmp_build_iptc_data:N.)

__pdfmeta_xmp_build_iptc:

```

1695 \cs_new_protected:Npn \__pdfmeta_xmp_build_iptc:
1696 {
1697     \tl_if_empty:NF\l__pdfmeta_xmp_iptc_data_tl
1698     {
1699         \__pdfmeta_xmp_add_packet_open_attr:nnn
1700         {Iptc4xmpCore}{CreatorContactInfo}{rdf:parseType="Resource"}
1701         \tl_gput_right:Ne\g__pdfmeta_xmp_packet_tl { \l__pdfmeta_xmp_iptc_data_tl }
1702         \__pdfmeta_xmp_add_packet_close:nn
1703         {Iptc4xmpCore}{CreatorContactInfo}
1704     }
1705 }

```

(End of definition for __pdfmeta_xmp_build_iptc:.)

4.15 Prism

__pdfmeta_xmp_build_prism:

complianceProfile

```

prism:subtitle/pdfsubtitl06 \cs_new_protected:Npn \__pdfmeta_xmp_build_prism:
1707 {

```

The compliance profile is a fix value taken from hyperxmp

```

1708     \__pdfmeta_xmp_add_packet_line:nnn
1709     {prism}{complianceProfile}
1710     {three}

```

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

```

1711     \__pdfmeta_xmp_lang_get:eNN
1712     {\GetDocumentProperties{hyperref/pdfsubtitle}}
1713     \l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl
1714     \__pdfmeta_xmp_add_packet_line_attr:nneV

```

```

1715     {prism}{subtitle}
1716     {xml:lang="\l__pdfmeta_tmpa_tl"}
1717     \l__pdfmeta_tmpb_tl

```

Then publicationName

```

1718     \__pdfmeta_xmp_lang_get:eNN
1719     {\GetDocumentProperties{hyperref/pdfpublication}}
1720     \l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl
1721     \__pdfmeta_xmp_add_packet_line_attr:nneV
1722     {prism}{publicationName}
1723     {xml:lang="\l__pdfmeta_tmpa_tl"}
1724     \l__pdfmeta_tmpb_tl

```

Now the rest

```

1725     \__pdfmeta_xmp_add_packet_line:nne
1726     {prism}{bookEdition}
1727     {\GetDocumentProperties{hyperref/pdfbookedition}}
1728     \__pdfmeta_xmp_add_packet_line:nne
1729     {prism}{aggregationType}
1730     {\GetDocumentProperties{hyperref/pdfpubtype}}
1731     \__pdfmeta_xmp_add_packet_line:nne
1732     {prism}{volume}
1733     {\GetDocumentProperties{hyperref/pdfvolumenum}}
1734     \__pdfmeta_xmp_add_packet_line:nne
1735     {prism}{number}
1736     {\GetDocumentProperties{hyperref/pdfissuenum}}
1737     \__pdfmeta_xmp_add_packet_line:nne
1738     {prism}{pageRange}
1739     {\GetDocumentProperties{hyperref/pdfpagerange}}
1740     \__pdfmeta_xmp_add_packet_line:nne
1741     {prism}{issn}
1742     {\GetDocumentProperties{hyperref/pdfissn}}
1743     \__pdfmeta_xmp_add_packet_line:nne
1744     {prism}{eIssn}
1745     {\GetDocumentProperties{hyperref/pdfeissn}}
1746     \__pdfmeta_xmp_add_packet_line:nne
1747     {prism}{doi}
1748     {\GetDocumentProperties{hyperref/pdfdoi}}
1749     \__pdfmeta_xmp_add_packet_line:nne
1750     {prism}{url}
1751     {\GetDocumentProperties{hyperref/pdfurl}}

```

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

```

1752     \tl_set:Ne \l__pdfmeta_tmpa_tl { \GetDocumentProperties{hyperref/pdfnumpages} }
1753     \__pdfmeta_xmp_add_packet_line:nne
1754     {prism}{pageCount}
1755     {\tl_if_blank:VTF \l__pdfmeta_tmpa_tl {\PreviousTotalPages}{\l__pdfmeta_tmpa_tl}}
1756 }

```

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

4.16 TDM

_pdfmeta_xmp_build_tdm:

```
1757 \cs_new_protected:Npn \_pdfmeta_xmp_build_tdm:
1758   {
    pdftdmreservation/tdm:reservation
1759   \_pdfmeta_xmp_add_packet_line:nne{tdm}{reservation}
1760     { \GetDocumentProperties{hyperref/pdftdmreservation} }
    pdftdmpolicy/tdm:policy
1761   \_pdfmeta_xmp_add_packet_line:nne{tdm}{policy}
1762     { \GetDocumentProperties{hyperref/pdftdmpolicy} }
1763   }
```

(End of definition for _pdfmeta_xmp_build_tdm:.)

4.17 User additions

\g_pdfmeta_xmp_user_packet_str

```
1764 \tl_new:N \g_pdfmeta_xmp_user_packet_tl
    (End of definition for \g_pdfmeta_xmp_user_packet_str.)
```

_pdfmeta_xmp_build_user:

```
1765 \cs_new_protected:Npn \_pdfmeta_xmp_build_user:
1766   {
1767     \int_zero:N \l_pdfmeta_xmp_indent_int
1768     \g_pdfmeta_xmp_user_packet_tl
1769     \int_set:Nn \l_pdfmeta_xmp_indent_int {3}
1770   }
```

(End of definition for _pdfmeta_xmp_build_user:.)

4.18 Activating the metadata

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

```
1771 \hook_new:n { pdfmeta/xmp }
1772 \AddToHook{shipout/lastpage}[pdfmanagement-testphase]
1773   {
1774     \bool_if:NT\g_pdfmeta_xmp_bool
1775     {
1776       \str_if_exist:NTF\c_sys_timestamp_str
1777         {
1778           \tl_set_eq:NN \l_pdfmeta_xmp_currentdate_tl \c_sys_timestamp_str
1779         }
    }
```

```

1780     {
1781       \file_get_timestamp:nN{\jobname.log}\l__pdfmeta_xmp_currentdate_tl
1782     }
1783     \__pdfmeta_xmp_date_split:VN\l__pdfmeta_xmp_currentdate_tl\l__pdfmeta_xmp_currentdate_seq
1784     \hook_use:n { pdfmeta/xmp }
1785     \__pdfmeta_xmp_build_packet:
1786     \pdf_object_new:n {__pdfmeta/xmp}
1787     \exp_args:No
1788     \__pdf_backend_metadata_stream:n {\g__pdfmeta_xmp_packet_tl}
1789     \pdfmanagement_add:nne {Catalog} {Metadata}{\pdf_object_ref:n{__pdfmeta/xmp}}
1790     \bool_if:NT \g__pdfmeta_xmp_export_bool
1791     {
1792       \iow_open:Nn\g_tmpa_iow{\g__pdfmeta_xmp_export_str.xmpi}
1793       \exp_args:NNo\iow_now:Nn\g_tmpa_iow{\g__pdfmeta_xmp_packet_tl}
1794       \iow_close:N\g_tmpa_iow
1795     }
1796   }
1797 }

```

4.19 User commands

`\pdfmeta_xmp_add:n`

```

1798 \cs_new_protected:Npn \pdfmeta_xmp_add:n #1
1799 {
1800   \tl_gput_right:Nn \g__pdfmeta_xmp_user_packet_tl
1801   {
1802     \__pdfmeta_xmp_add_packet_chunk:n { #1 }
1803   }
1804 }

```

(End of definition for `\pdfmeta_xmp_add:n`. This function is documented on page 10.)

`\pdfmeta_xmp_xmlns_new:nn`

```

1805 \cs_new_protected:Npn \pdfmeta_xmp_xmlns_new:nn #1 #2
1806 {
1807   \prop_if_in:NnTF \g__pdfmeta_xmp_xmlns_prop {#1}
1808   {\msg_warning:nnnn{pdfmeta}{xmp-defined}{xmlns-namespace}{#1}}
1809   {\__pdfmeta_xmp_xmlns_new:nn {#1}{#2}}
1810 }

```

(End of definition for `\pdfmeta_xmp_xmlns_new:nn`. This function is documented on page 10.)

`\pdfmeta_xmp_schema_new:nnn`

```

1811 \cs_set_eq:NN \pdfmeta_xmp_schema_new:nnn \__pdfmeta_xmp_schema_new:nnn

```

(End of definition for `\pdfmeta_xmp_schema_new:nnn`. This function is documented on page 11.)

`\pdfmeta_xmp_property_new:nnnn`

```

1812 \cs_set_eq:NN \pdfmeta_xmp_property_new:nnnn \__pdfmeta_xmp_property_new:nnnn

```

(End of definition for `\pdfmeta_xmp_property_new:nnnn`. This function is documented on page 11.)

```

\pdfmeta_xmp_add_declaration:n
\pdfmeta_xmp_add_declaration:e
1813 \cs_new_protected:Npn \pdfmeta_xmp_add_declaration:n #1 %conformsTo uri
1814 {
1815   \__pdfmeta_xmp_schema_enable_pdfd:
1816   \prop_gput:Nnn\g__pdfmeta_xmp_pdfd_data_prop{#1}{}
1817 }
1818 \cs_generate_variant:Nn \pdfmeta_xmp_add_declaration:n {e}

```

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

```

\pdfmeta_xmp_add_declaration:nnnnn
\pdfmeta_xmp_add_declaration:enenn
1819 \cs_new_protected:Npn \pdfmeta_xmp_add_declaration:nnnnn #1#2#3#4#5
1820 %#1=conformsTo uri, #2 claimBy, #3 claimDate #4 claimCredentials #4 claimReport
1821 {
1822   \__pdfmeta_xmp_schema_enable_pdfd:
1823   \tl_set:Nn \l__pdfmeta_tmpa_tl
1824   {
1825     \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1826     \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimBy}{#2}
1827     \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimDate}{#3}
1828     \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimCredentials}{#4}
1829     \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimReport}{#5}
1830     \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1831   }
1832   \prop_get:NnNT \g__pdfmeta_xmp_pdfd_data_prop {#1}\l__pdfmeta_tmpb_tl
1833   {
1834     \tl_concat:NNN \l__pdfmeta_tmpa_tl \l__pdfmeta_tmpa_tl \l__pdfmeta_tmpb_tl
1835   }
1836   \prop_gput:Nno\g__pdfmeta_xmp_pdfd_data_prop{#1}
1837   {
1838     \l__pdfmeta_tmpa_tl
1839   }
1840 }
1841 \cs_generate_variant:Nn \pdfmeta_xmp_add_declaration:nnnnn {e,eee}

```

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

4.20 Default declarations

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

```

\__pdfmeta_xmp_wtpdf_reuse_declaration:
pdfmeta_xmp_wtpdf_accessibility_declaration:
1842 \cs_new:Npn \__pdfmeta_xmp_iso_today:
1843 {
1844   \int_use:N\c_sys_year_int-
1845   \int_compare:nNnT {\c_sys_month_int} < {10}{0} \int_use:N\c_sys_month_int -
1846   \int_compare:nNnT {\c_sys_day_int} < {10}{0} \int_use:N\c_sys_day_int
1847 }
1848 \cs_new_protected:Npn \__pdfmeta_xmp_wtpdf_reuse_declaration:
1849 {
1850   \pdfmeta_xmp_add_declaration:eeenn

```

```

1851     {http://pdfa.org/declarations/wtpdf\c_hash_str reuse1.0}
1852     {LaTeX~Project}
1853     {\_pdfmeta_xmp_iso_today:}{}{}
1854 }
1855 \cs_new_protected:Npn \_pdfmeta_xmp_wtpdf_accessibility_declaration:
1856 {
1857   \pdfmeta_xmp_add_declaration:enonn
1858   {http://pdfa.org/declarations/wtpdf\c_hash_str accessibility1.0}
1859   {LaTeX~Project}
1860   {\_pdfmeta_xmp_iso_today:}{}{}
1861 }

```

(End of definition for _pdfmeta_xmp_wtpdf_reuse_declaration: and _pdfmeta_xmp_wtpdf-accessibility_declaration:.)

```

1862 </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	C
\& 926	char commands:
\' 861	\char_generate:nn 827, 832, 833, 834
\+ 861	clist commands:
\- 861, 942	\clist_if_empty:nTF 1039, 1056
\[. 942	\clist_map_inline:nn
\\ 11, 12, 16 548, 647, 1043, 1060, 1071
\] 942	complianceProfile <u>1706</u>
	CreatorTool/pdfcreator <u>1504</u>
	cs commands:
A	\cs_generate_variant:Nn
\A 942 625, 866, 938,
\AddToDocumentProperties	957, 966, 974, 980, 987, 1002, 1012,
. 678, 685, 692, 701, 707,	1022, 1035, 1052, 1084, 1818, 1841
713, 719, 725, 731, 737, 743, 750, 760	\cs_gset_eq:NN 1494
\AddToHook 390, 412, 564, 751, 761, 763, 1772	\cs_if_exist:NTF 64
	\cs_if_exist_use:N 1189
B	\cs_new:Npn
BaseUrl/baseurl <u>1504</u> 42, 826, 830, 838, 844, 867, 1842
bitset commands:	\cs_new_protected:Nn 626
\bitset_set_false:Nn 117, 118, 119	\cs_new_protected:Npn 29,
\bitset_set_true:Nn 116	46, 81, 89, 96, 102, 108, 114, 526,
\bitset_to_arabic:N	541, 616, 645, 850, 855, 862, 897,
. 120, 121, 122, 123, 124	910, 922, 943, 959, 967, 975, 981,
bool commands:	988, 993, 1003, 1013, 1023, 1036,
\bool_gset_false:N 771, 810	1053, 1085, 1135, 1167, 1195, 1218,
\bool_gset_true:N 619, 770, 805, 814	1269, 1387, 1438, 1451, 1496, 1504,
\bool_if:NTF 1774, 1790	1525, 1544, 1558, 1572, 1579, 1604,
\bool_lazy_or:nnTF 416, 822	1643, 1661, 1695, 1706, 1757, 1765,
\bool_new:N 618, 793	1798, 1805, 1813, 1819, 1848, 1855

<code>\cs_set_eq:NN</code>	784, 790, 1090, 1811, 1812	<code>\int_incr:N</code>	852
D		<code>\int_new:N</code>	837
<code>\d</code>	861	<code>\int_set:Nn</code>	1128, 1769
dc commands:		<code>\int_use:N</code>	1844, 1845, 1846
<code>dc:description/pdfsubject</code>	1604	<code>\int_zero:N</code>	1130, 1767
<code>dc:identifier/pdfidentifier</code>	1604	iow commands:	
<code>dc:language/lang/pdflang</code>	1604	<code>\iow_close:N</code>	1794
<code>dc:Nreator/pdfauthor</code>	1604	<code>\iow_newline:</code>	424, 440, 447, 840, 846
<code>dc:publisher/pdfpublisher</code>	1604	<code>\iow_now:Nn</code>	1793
<code>dc:subject/pdfkeywords</code>	1604	<code>\iow_open:Nn</code>	1792
<code>dc:type/pdftype</code>	1604	<code>\g_tmpa_iow</code>	1792, 1793, 1794
<code>\DocumentMetadata</code>	3, 4	<code>iptc_(schema)</code>	1377
E		J	
exp commands:		<code>\jobname</code>	1585, 1594, 1781
<code>\exp_args:NNe</code>	584, 589, 595	K	
<code>\exp_args:Nne</code>	406	kernel internal commands:	
<code>\exp_args:Nnne</code>	66	<code>\g__kernel_pdfmanagement_end_</code>	
<code>\exp_args:NNo</code>	501, 1793	<code>run_code_tl</code>	394
<code>\exp_args:No</code>	1787	keys commands:	
<code>\exp_args:NV</code>	602, 605	<code>\keys_define:nn</code>	456, 463, 670, 778, 796
<code>\exp_last_unbraced:No</code>	1538, 1540	<code>\l_keys_key_str</code>	503
<code>\exp_not:n</code>	963, 971, 1140	<code>\keys_set:nn</code>	460, 775, 800
F		M	
file commands:		msg commands:	
<code>\file_get_timestamp:nN</code>	1781	<code>\msg_error:nnn</code>	696
G		<code>\msg_new:nnn</code>	7, 9, 10, 14, 819, 820, 821
<code>\GetDocumentProperties</code>	900, 1087, 1088, 1501, 1509, 1512, 1534, 1575, 1577, 1581, 1585, 1590, 1600, 1602, 1607, 1611, 1613, 1624, 1627, 1631, 1639, 1641, 1648, 1653, 1667, 1671, 1675, 1679, 1683, 1687, 1691, 1712, 1719, 1727, 1730, 1733, 1736, 1739, 1742, 1745, 1748, 1751, 1752, 1760, 1762	<code>\msg_note:nnn</code>	438
group commands:		<code>\msg_warning:nnn</code>	39, 422, 445, 579, 612
<code>\group_begin:</code>	398, 543, 925	<code>\msg_warning:nnnn</code>	1172, 1215, 1808
<code>\group_end:</code>	407, 562, 934	<code>\msg_warning:nnnnn</code>	653, 661
H		P	
hook commands:		pdf commands:	
<code>\hook_gput_code:nnn</code>	126, 620	<code>\pdf_object_if_exist:nTF</code>	528
<code>\hook_new:n</code>	1771	<code>\pdf_object_new:n</code>	530, 1786
<code>\hook_use:n</code>	1784	<code>\pdf_object_ref:n</code>	547, 1789
I		<code>\pdf_object_ref_last:</code>	561
int commands:		<code>\pdf_object_unnamed_write:nn</code>	560
<code>\int_compare:nNnTF</code>	1530, 1608, 1614, 1845, 1846	<code>\pdf_object_write:nnn</code>	531
<code>\int_decr:N</code>	857	<code>\pdf_string_from_unicode:nnN</code>	555
<code>\int_if_zero:nTF</code>	432	<code>\pdf_version:</code>	4, 629, 640, 652, 654, 660, 662, 1502
<code>\int_if_zero_p:n</code>	417, 418	<code>\pdf_version_compare:NnTF</code>	83, 91
		<code>\pdf_version_gset:n</code>	625, 633, 636, 642
		pdf internal commands:	
		<code>__pdf_backend_metadata_stream:</code>	1788
		<code>__pdf_backend_Names_gpsh:nn</code>	406
		<code>__pdf_backend_omit_charset:n</code>	133
		<code>__pdf_backend_omit_cidset:n</code>	135
		<code>__pdf_backend_omit_info:n</code>	131

__pdf_backend_set_regression_- data:	617	pdfmeta internal commands:	
pdfaid~(schema)	1237	__pdfmeta_check_standard_- pdfversion:	136, 645
pdfannot commands:		__pdfmeta_embed_colorprofile:n	526, 526, 572, 602
\pdfannot_dict_put:nnn	120, 121, 122, 123, 124	__pdfmeta_force_standard_- pdfversion:	626, 677, 684, 691, 749, 759
\l_pdfannot_F_bitset	116, 117, 118, 119, 120, 121, 122, 123, 124	\g__pdfmeta_outputintents_prop	455, 469, 477, 485, 493, 502, 568, 586, 591, 597, 603
pdfdict commands:		\g__pdfmeta_standard_A_prop . .	25, 33, 36, 675, 676, 682, 683, 689, 690
\pdfdict_if_empty:nTF	396	\g__pdfmeta_standard_pdf/A-1B_- prop	138
\pdfdict_new:n	507	\g__pdfmeta_standard_pdf/A-2A_- prop	138
\pdfdict_put:nnn	399, 400, 508, 544, 545, 556	\g__pdfmeta_standard_pdf/A-2B_- prop	138
\pdfdict_use:n	560	\g__pdfmeta_standard_pdf/A-2U_- prop	138
pdffile commands:		\g__pdfmeta_standard_pdf/A-3A_- prop	138
\g_pdffile_embed_nonpdfa_int	418, 432	\g__pdfmeta_standard_pdf/A-3B_- prop	138
\g_pdffile_embed_pdfa_int	417	\g__pdfmeta_standard_pdf/A-3U_- prop	138
\pdffile_embed_stream:nnN	401	\g__pdfmeta_standard_pdf/A-4_- prop	138
pdfmanagement commands:		\g__pdfmeta_standard_pdf/A-4F_- prop	138
\pdfmanagement_add:nnn	561, 622, 623, 1516, 1520, 1789	\g__pdfmeta_standard_pdf/UA-1_- prop	329
\pdfmanagement_get_documentproperties:nNTF	1535, 1616	\g__pdfmeta_standard_pdf/UA-2_- prop	329
\pdfmanagement_remove:nn	1609, 1615	\g__pdfmeta_standard_pdf/X-4_- prop	349
pdfmeta commands:		\g__pdfmeta_standard_pdf/X-4P_- prop	349
\pdfmeta_set_regression_data:	6, 616	\g__pdfmeta_standard_pdf/X-5G_- prop	349
\pdfmeta_standard_family:nn	2, 29, 29, 649, 749, 759	\g__pdfmeta_standard_pdf/X-5N_- prop	349
\pdfmeta_standard_get:nN	3, 46, 46	\g__pdfmeta_standard_pdf/X-5PG_- prop	349
\pdfmeta_standard_item:n	2, 42, 42, 587, 592, 598, 636, 642, 656, 664, 1527, 1529, 1530, 1531, 1532, 1608, 1614	\g__pdfmeta_standard_pdf/X-6_- prop	349
\pdfmeta_standard_verify:n	2, 50	\g__pdfmeta_standard_pdf/X-6N_- prop	349
\pdfmeta_standard_verify:nn	3, 60	\g__pdfmeta_standard_pdf/X-6P_- prop	349
\pdfmeta_standard_verify:nnN	3	\g__pdfmeta_standard_prop	12, 25, 33, 34, 36, 44, 48, 52, 62, 70, 420, 434, 676, 683, 690, 753, 762
\pdfmeta_standard_verify:nnTF	2, 60, 628, 639, 651, 659		
\pdfmeta_standard_verify:nTF	2, 50, 128, 130, 132, 134, 392, 414, 430, 566, 631		
\pdfmeta_standard_verify_p:n	50		
\pdfmeta_xmp_add:n	10, 1798, 1798		
\pdfmeta_xmp_add_declaration:n	10, 1813, 1813, 1818		
\pdfmeta_xmp_add_declaration:nnnnn	10, 1819, 1819, 1841, 1850, 1857		
\pdfmeta_xmp_property_new:nnnnn	11, 1812, 1812		
\pdfmeta_xmp_schema_new:nnn	11, 1811, 1811		
\pdfmeta_xmp_xmlns_new:nn	10, 1805, 1805		

```

\g__pdfmeta_standard_UA_prop ...
    ..... 25, 748, 758
\__pdfmeta_standard_verify_-
  handler_annot_action_A:nn 102, 102
\__pdfmeta_standard_verify_-
  handler_max_pdf_version:nn 88, 89
\__pdfmeta_standard_verify_-
  handler_min_pdf_version:nn 80, 81
\__pdfmeta_standard_verify_-
  handler_named_actions:nn .. 96, 96
\__pdfmeta_standard_verify_-
  handler_outputintent_subtype:nn
    ..... 108, 108
\g__pdfmeta_standard_X_prop .. 25,
    700, 706, 712, 718, 724, 730, 736, 742
\l__pdfmeta_tmpa_seq .....
    19, 946, 947, 953, 954, 1514, 1515,
    1518, 1519, 1522, 1523, 1632, 1634
\g__pdfmeta_tmpa_str .....
    ..... 22, 929, 930, 931, 932, 933, 935
\l__pdfmeta_tmpa_str .....
    ..... 19, 555, 557, 998, 999,
    1008, 1009, 1018, 1019, 1581, 1582,
    1586, 1589, 1590, 1591, 1595, 1598
\l__pdfmeta_tmpa_tl .....
    ..... 19, 405, 406, 420, 425,
    434, 436, 448, 553, 555, 928, 929,
    1028, 1031, 1033, 1062, 1063, 1068,
    1073, 1074, 1077, 1200, 1514, 1516,
    1518, 1520, 1522, 1535, 1538, 1540,
    1616, 1618, 1632, 1713, 1716, 1720,
    1723, 1752, 1755, 1823, 1834, 1838
\l__pdfmeta_tmpb_seq ..... 19
\l__pdfmeta_tmpb_tl .....
    ..... 19, 599, 600, 602, 607,
    612, 1062, 1066, 1068, 1073, 1077,
    1713, 1717, 1720, 1724, 1832, 1834
\__pdfmeta_verify_pdfa_annot_-
  flags: ..... 114, 129
\__pdfmeta_write_outputintent:nn
    ..... 526, 541, 574, 606
\__pdfmeta_xmp_add_packet_-
  chunk:n . 959, 959, 966, 977, 984,
    991, 999, 1019, 1096, 1129, 1131, 1802
\__pdfmeta_xmp_add_packet_-
  chunk:nN ..... 967, 967, 974, 1009
\__pdfmeta_xmp_add_packet_-
  close:nn ..... 988,
    988, 1048, 1049, 1080, 1081, 1109,
    1110, 1125, 1126, 1127, 1187, 1188,
    1190, 1210, 1225, 1416, 1417, 1418,
    1419, 1420, 1472, 1473, 1474, 1488,
    1489, 1490, 1491, 1492, 1554,
    1555, 1567, 1568, 1570, 1702, 1830
\__pdfmeta_xmp_add_packet_-
  field:nnn 1218, 1218, 1400, 1402,
    1404, 1406, 1408, 1410, 1412, 1414,
    1464, 1466, 1468, 1470, 1484, 1486
\__pdfmeta_xmp_add_packet_-
  line:nnn ..... 993,
    993, 1002, 1033, 1045, 1181, 1182,
    1183, 1206, 1207, 1208, 1209, 1222,
    1223, 1224, 1392, 1393, 1395, 1396,
    1456, 1457, 1459, 1460, 1476, 1477,
    1479, 1480, 1502, 1515, 1519, 1523,
    1527, 1528, 1531, 1532, 1533, 1537,
    1539, 1561, 1574, 1576, 1588, 1597,
    1599, 1601, 1630, 1635, 1645, 1649,
    1708, 1725, 1728, 1731, 1734, 1737,
    1740, 1743, 1746, 1749, 1753,
    1759, 1761, 1826, 1827, 1828, 1829
\__pdfmeta_xmp_add_packet_-
  line:nnnN 1003, 1003, 1012, 1665,
    1669, 1673, 1677, 1681, 1685, 1689
\__pdfmeta_xmp_add_packet_line_-
  attr:nnnn ..... 1013, 1013,
    1022, 1065, 1067, 1076, 1714, 1721
\__pdfmeta_xmp_add_packet_line_-
  default:nnnn ..... 1023,
    1023, 1035, 1498, 1506, 1510, 1636
\__pdfmeta_xmp_add_packet_-
  list:nnnn .....
    ..... 1053, 1084, 1610, 1625, 1640
\__pdfmeta_xmp_add_packet_list_-
  simple:nnnn . 1036, 1052, 1606,
    1612, 1618, 1621, 1623, 1628, 1633
\__pdfmeta_xmp_add_packet_-
  open:nn .... 975, 975, 980, 1041,
    1042, 1058, 1059, 1098, 1099, 1103,
    1104, 1184, 1185, 1205, 1389, 1390,
    1398, 1399, 1453, 1454, 1462, 1463,
    1482, 1483, 1548, 1549, 1564, 1565
\__pdfmeta_xmp_add_packet_open_-
  attr:nnn .....
    .. 981, 981, 987, 1101, 1180, 1221,
    1391, 1455, 1475, 1560, 1699, 1825
\__pdfmeta_xmp_add_pdfxid: . 702,
    708, 714, 720, 726, 732, 738, 744, 1269
\g__pdfmeta_xmp_bool .....
    ..... 618, 770, 771, 1774
\__pdfmeta_xmp_build_dc: .....
    ..... 1116, 1604, 1604
\__pdfmeta_xmp_build_iptc: ....
    ..... 1121, 1695, 1695
\__pdfmeta_xmp_build_iptc_data:N
    ..... 1091, 1661, 1661
\__pdfmeta_xmp_build_packet: ...
    ..... 1085, 1085, 1785

```

_pdfmeta_xmp_build_pdf:	_pdfmeta_xmp_lang_get:nNN
. 1112, 1496, 1496 943, 957, 1062, 1073, 1711, 1718
_pdfmeta_xmp_build_pdfd:	\l_pdfmeta_xmp_lang_regex 941, 946
. 1115, 1544, 1544	\l_pdfmeta_xmp_metalang_tl
_pdfmeta_xmp_build_pdfd_-	939, 949, 1063, 1074, 1088, 1089, 1090
claim:nn 1552, 1558, 1558	\g_pdfmeta_xmp_packet_tl
_pdfmeta_xmp_build_photshop: 958, 961, 1701, 1788, 1793
. 1117, 1572, 1572	\g_pdfmeta_xmp_pdfd_data_prop .
_pdfmeta_xmp_build_prism: 1543, 1546, 1550, 1816, 1832, 1836
. 1120, 1706, 1706	_pdfmeta_xmp_print_date:N
_pdfmeta_xmp_build_standards: 867, 867, 1515, 1519, 1523, 1634
. 1114, 1525, 1525	_pdfmeta_xmp_property_new:nnnnn
_pdfmeta_xmp_build_tdm: 1194, 1195, 1231, 1241,
. 1122, 1757, 1757	1247, 1257, 1263, 1276, 1287, 1293,
_pdfmeta_xmp_build_user:	1299, 1305, 1311, 1317, 1323, 1329,
. 1123, 1765, 1765	1335, 1341, 1347, 1353, 1359, 1365,
_pdfmeta_xmp_build_xmp:	1371, 1381, 1426, 1432, 1445, 1812
. 1118, 1504, 1504	_pdfmeta_xmp_sanitize:nN
_pdfmeta_xmp_build_xmpMM: 922, 922, 938, 998, 1008, 1018
. 1119, 1579, 1579	_pdfmeta_xmp_schema_enable_-
_pdfmeta_xmp_build_xmpRights:	pdfd: 1438, 1494, 1815, 1822
. 1113, 1643, 1643	_pdfmeta_xmp_schema_new:nnn . .
_pdfmeta_xmp_create_uid:nN 1167, 1167, 1227, 1237, 1253,
. 910, 910, 1584, 1593	1272, 1283, 1377, 1422, 1441, 1811
\l_pdfmeta_xmp_currentdate_seq	\g_pdfmeta_xmp_schema_property_-
. 895, 903, 1783	prop 1194, 1200, 1202
\l_pdfmeta_xmp_currentdate_tl	\l_pdfmeta_xmp_schema_seq
. 895, 904, 1594, 1778, 1781, 1783 1094, 1105, 1166, 1175
_pdfmeta_xmp_date_get:nNN	\g_pdfmeta_xmp_user_packet_str 1764
. 897, 897, 1513, 1517, 1521, 1632	\g_pdfmeta_xmp_user_packet_tl .
\l_pdfmeta_xmp_date_regex 859, 864 1764, 1768, 1800
_pdfmeta_xmp_date_split:nN	_pdfmeta_xmp_wtpdf_accessibility_-
. 862, 862, 866, 907, 1783	declaration:
_pdfmeta_xmp_decr_indent: 765, 787, 790, 1842, 1855
. 838, 855, 990, 1693	_pdfmeta_xmp_wtpdf_reuse_-
\l_pdfmeta_xmp_doclang_tl	declaration:
. 939, 1087, 1090, 1629 766, 781, 784, 1842, 1848
\g_pdfmeta_xmp_export_bool	_pdfmeta_xmp_xmlns_new:nn
. 793, 805, 810, 814, 1790 1135, 1135, 1143,
\g_pdfmeta_xmp_export_str	1144, 1145, 1146, 1147, 1148, 1149,
. 794, 806, 815, 1792	1151, 1152, 1153, 1154, 1155, 1156,
_pdfmeta_xmp_generate_bom:	1157, 1158, 1159, 1160, 1161, 1162,
. 822, 826, 830, 1097	1163, 1164, 1165, 1271, 1440, 1809
_pdfmeta_xmp_incr_indent:	\g_pdfmeta_xmp_xmlns_prop
. 838, 850, 978, 985, 1664 1133, 1137, 1807
_pdfmeta_xmp_indent:	\g_pdfmeta_xmp_xmlns_tl
. 838, 838, 963, 971 1102, 1133, 1138
_pdfmeta_xmp_indent:n 838, 844, 1140	pdfmetatmpa internal commands:
\l_pdfmeta_xmp_indent_int . 837,	\g_pdfmetatmpa_str 19
841, 852, 857, 1128, 1130, 1767, 1769	pdfuaid~(schema) 1253
\l_pdfmeta_xmp_iptc_data_tl	PDFversion 1496
. 1091, 1092, 1660, 1697, 1701	pdfxid~(schema) 1269
_pdfmeta_xmp_iso_today:	
. 1842, 1853, 1860	

photoshop commands:	
photoshop:AuthorsPosition/pdfauthortitle	1604
photoshop:CaptionWriter/pdfcaptionwriter	1604
\PreviousTotalPages	1755
prg commands:	
\prg_do_nothing:	784 , 790 , 1494
\prg_new_conditional:Npnn	50
\prg_new_protected_conditional:Npnn	60
\prg_replicate:nn	841 , 847 , 1129
\prg_return_false:	54 , 73 , 85 , 93 , 100 , 106 , 112
\prg_return_true:	57 , 77 , 86 , 94 , 99 , 105 , 111
prism commands:	
prism:subtitle/pdfsubtitle	1706
prism~(schema)	1283
Producer/pdfproducer	1496
prop commands:	
\prop_const_from_keyval:Nn	510 , 517
\prop_get:NnN	48 , 596
\prop_get:NnNTF	420 , 434 , 550 , 1200 , 1832
\prop_gput:Nnn	204 , 206 , 208 , 214 , 218 , 220 , 232 , 234 , 236 , 244 , 246 , 248 , 257 , 259 , 261 , 272 , 274 , 276 , 284 , 286 , 288 , 296 , 298 , 300 , 302 , 304 , 306 , 308 , 321 , 324 , 469 , 477 , 485 , 493 , 502 , 590 , 753 , 762 , 1137 , 1202 , 1816 , 1836
\prop_gremove:Nn	211 , 223 , 264 , 310 , 312 , 314 , 327
\prop_gset_eq:NN	33 , 34 , 36 , 201 , 229 , 241 , 254 , 269 , 281 , 293 , 318 , 675 , 676 , 682 , 683 , 689 , 690 , 700 , 706 , 712 , 718 , 724 , 730 , 736 , 742 , 748 , 758
\prop_gset_from_keyval:Nn	139 , 331 , 341 , 358 , 362 , 366 , 370 , 374 , 378 , 382 , 386
\prop_if_empty:NTF	1546
\prop_if_exist:NTF	31 , 570 , 600
\prop_if_in:NnTF	52 , 62 , 585 , 1807
\prop_item:Nn	44 , 70 , 534
\prop_map_inline:Nn	568 , 603 , 1550
\prop_new:N	25 , 26 , 27 , 28 , 138 , 200 , 228 , 240 , 253 , 268 , 280 , 292 , 317 , 329 , 330 , 349 , 350 , 351 , 352 , 353 , 354 , 355 , 356 , 455 , 1134 , 1194 , 1543
\ProvidesExplPackage	3
	R
regex commands:	
\regex_extract_once:NnN	946
\regex_new:N	859 , 941
\regex_set:Nn	860 , 942
\regex_split:NnN	864
	S
seq commands:	
\seq_if_empty:NTF	947
\seq_item:Nn	869 , 871 , 873 , 875 , 877 , 878 , 880 , 881 , 883 , 884 , 885 , 886 , 888 , 889 , 892 , 953 , 954
\seq_map_inline:Nn	1105
\seq_new:N	23 , 24 , 896 , 1166
\seq_put_right:Nn	1175
\seq_remove_all:Nn	1094
\seq_set_eq:NN	903
str commands:	
\c_hash_str	10 , 1100 , 1150 , 1158 , 1161 , 1162 , 1163 , 1164 , 1851 , 1858
\str_case:nn	1653
\str_convert_pdfname:n	544
\str_greplac_all:Nnn	930 , 931 , 932 , 933
\str_gset:Nn	815 , 929
\str_gset_eq:NN	806
\str_if_empty:NTF	1582 , 1591
\str_if_eq:nnTF	436
\str_if_exist:NTF	1776
\str_lowercase:n	912
\str_new:N	21 , 22 , 794 , 1133
\str_range:Nnn	915 , 916 , 917 , 918 , 919
\str_set:Nn	912 , 913 , 1581 , 1590
\str_set_eq:NN	935
\c_tilde_str	927
sys commands:	
\c_sys_day_int	1846
\c_sys_engine_exec_str	622 , 1500
\c_sys_engine_version_str	622 , 1500
\sys_if_engine luatex_p:	823
\sys_if_engine xetex_p:	824
\c_sys_jobname_str	806 , 1638
\c_sys_month_int	1845
\c_sys_timestamp_str	6 , 1776 , 1778
\c_sys_year_int	1844
	T
tdmrep␣(schema)	1422
tex commands:	
\tex_mdfivesum:D	912
text commands:	
\text_declare_purify_equivalent:Nn	926 , 927

<code>\text_purify:n</code>	928	<code>\tl_if_in:nnTF</code>	98, 104
<code>\texttilde</code>	927	<code>\tl_new:N</code>	19, 20, 895,
tl commands:		939, 940, 958, 1176, 1177, 1660, 1764	
<code>\c_space_tl</code>	533, 841, 847	<code>\tl_put_right:Nn</code>	969
<code>\tl_clear:N</code>	1663	<code>\tl_set:Nn</code> 900, 928, 949, 950, 953,	
<code>\tl_concat:NNN</code>	1834	954, 1028, 1031, 1087, 1088, 1752, 1823	
<code>\tl_gput_right:Nn</code>		<code>\tl_set_eq:NN</code>	904, 1778
394, 961, 1138, 1178, 1203, 1701, 1800		<code>\tl_to_str:N</code>	926, 929
<code>\tl_if_blank:nTF</code> ... 467, 475, 483,		<code>\tl_use:N</code>	1107, 1186
491, 499, 869, 876, 879, 882, 887,			
901, 996, 1006, 1016, 1026, 1089, 1755			
<code>\tl_if_empty:NTF</code>	1092, 1697		
<code>\tl_if_empty:nTF</code>	1562		
<code>\tl_if_eq:nnTF</code>	110, 1063, 1074		
<code>\tl_if_exist:NTF</code>	1170, 1198		

U

use commands:	
<code>\use:N</code>	67
<code>\use_i:nn</code>	1538
<code>\use_ii:nn</code>	1540