

The l3pdfmeta module

PDF standards

LaTeX PDF management bundle

The LaTeX Project^{*}

Version 0.97d, released 2026-10-06

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 a colorprofile and an /OutputIntent, some other things are forbidden or restricted, e.g. the action dictionary of an annotation should not contain Javascript.

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

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

^{*}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 {<family>} {<code>}
```

This command allows to change the family for which requirements should be checked. *<family>* should be one of A, X or UA. The function switches to the requirements of this family, then executes *<code>* 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). *<code>* can do testing or retrieve values or other things but it shouldn't contain another `\pdfmeta_standard_family:nn`.

```
\pdfmeta_standard_verify:nTF * \pdfmeta_standard_verify:nTF {<requirement>} {<true code>} {<false code>}
```

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

```
\pdfmeta_standard_verify:nnTF \pdfmeta_standard_verify:nnTF {<requirement>} {<value>} {<true code>} {<false code>}
```

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

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

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

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

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 {<requirement>} <tl var>`

This retrieves the value of `<requirement>` and stores it in the `<tl var>`. If the `<requirement>` is not found the special value `\q_no_value` is used. The `<tl var>` 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 been 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 $\text{\LaTeX}$ 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 fixed 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 “grÃ¼ÃŸe”. As XMP-metadata are in XML format special chars like `<`, `>`, and `&` and „ must be escaped.

`hyperxmp` hooks into `hyperref` and passes all input through `\pdfstringdef`. This means a word like “hallo” is first converted by `\pdfstringdef` into `\376\377\000h\000a\0001\0001\000o` and then back to UTF-8 by `hyperxmp` and in the course of this action the XML-escapings are applied. `pdfx` uses `\pdfstringdef` together with a special fontencoding (similar to the PU-encoding of `hyperref`) for a similar aim. The code here is based on `\text_purify:n` followed by a few replacements for the escaping.

User data should normally be declared in the preamble (or even in the `\DocumentMetadata` command), and consist of rather simple text; `&` can be entered as `\&` (but directly `&` will normally work too), babel shorthands should not be used. Some data, e.g., `pdftitle`, `pdfsubtitle`, `pdfcopyright`, 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:ennnn
  {https://pdfa.org/declarations\c_hash_str wcag21A}{2023-11-20}{}{}
\pdfmeta_xmp_add_declaration:nnnnn
  {https://github.com/TikZlings/no-duck-harmed}
  {Ulrike-Fischer}{2023-11-20}{Bär}{https://github.com/u-fischer/bearwear}
\pdfmeta_xmp_add_declaration:nnnnn
  {https://github.com/TikZlings/no-duck-harmed}
  {Ulrike-Fischer}{2023-11-20}{Paulo}{https://github.com/cereda/sillypage}
\ExplSyntaxOff
\begin{document}
  text
\end{document}
```

2.3.3 Dates

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

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

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

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

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

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

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

2.4 Language

The code assumes that a default language is always declared (as the `pdfmanagement` gives the `/Lang` entry in the catalog a default value) This language can be changed with the 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`.

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

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

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

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

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

<code>\pdfmeta_xmp_add_declaration:n</code> <code>\pdfmeta_xmp_add_declaration:e</code>	<code>\pdfmeta_xmp_add_declaration:n {<uri>}</code>
--	---

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

<code>\pdfmeta_xmp_add_declaration:nnnnn</code> <code>\pdfmeta_xmp_add_declaration:(ennnn eeenn)</code>	<code>\pdfmeta_xmp_add_declaration:nnnnn</code> <code>{<uri>} {<By>} {<Date>} {<Credentials>} {<Report>}</code>
--	--

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

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

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

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

\pdfmeta_xmp_property_new:nnnnn {\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}{2026-10-06}{0.97d}{
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_tmpp_tl      19 \tl_new:N \l__pdfmeta_tmpa_tl  
\l__pdfmeta_tmpa_str     20 \tl_new:N \l__pdfmeta_tmpp_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_tmpp_seq     23 \seq_new:N \l__pdfmeta_tmpa_seq  
                          24 \seq_new:N \l_pdfmeta_tmpp seq
```

(End of definition for `\l__pdfmeta_tpa_t1` 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      25 \prop_new:N \g__pdfmeta_standard_prop
\g__pdfmeta_standard_UA_prop     26 \prop_new:N \g__pdfmeta_standard_A_prop
\g__pdfmeta_standard_X_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_prop
\g_pdfmeta_standard_pdf/A-2U_prop
\g_pdfmeta_standard_pdf/A-3A_prop
\g_pdfmeta_standard_pdf/A-3B_prop
\g_pdfmeta_standard_pdf/A-3U_prop
\g_pdfmeta_standard_pdf/A-4_prop
\g_pdfmeta_standard_pdf/A-4F_prop
\g_pdfmeta_standard_pdf/A-4E_prop
138 \prop_new:c { g__pdfmeta_standard_pdf/A-1B_prop }
139 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/A-1B_prop }
140 {
141   ,name           = pdf/A-1B
142   ,type           = A
143   ,level          = 1
144   ,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}
329 %A-4e =====
330 \prop_new:c { g__pdfmeta_standard_pdf/A-4E_prop }
331 \prop_gset_eq:cc
332 { g__pdfmeta_standard_pdf/A-4E_prop }
333 { g__pdfmeta_standard_pdf/A-4_prop }
334 \prop_gput:cnn
335 { g__pdfmeta_standard_pdf/A-4E_prop }{conformance}{E}

```

(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
336 \prop_new:c { g__pdfmeta_standard_pdf/UA-1_prop}
337 \prop_new:c { g__pdfmeta_standard_pdf/UA-2_prop}
338 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/UA-1_prop }
339 {
340   ,name           = pdf/UA-1
341   ,level          = 1
342   ,year           = 2014
343   ,min_pdf_version = 1.4           %minimum
344   ,max_pdf_version = 1.7           %maximum
345   ,tagged         =
346   ,link_Contents  =
347 }
348 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/UA-2_prop }
349 {
350   ,name           = pdf/UA-2
351   ,level          = 2
352   ,year           = 2024
353   ,min_pdf_version = 2.0           %minimum
354   ,tagged         =
355 }

```

(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_prop
\g__pdfmeta_standard_pdf/X-5N_prop
\g__pdfmeta_standard_pdf/X-5PG_prop
\g__pdfmeta_standard_pdf/X-6_prop
\g__pdfmeta_standard_pdf/X-6N_prop
\g__pdfmeta_standard_pdf/X-6P_prop
356 \prop_new:c { g__pdfmeta_standard_pdf/X-4_prop}
357 \prop_new:c { g__pdfmeta_standard_pdf/X-4P_prop}
358 \prop_new:c { g__pdfmeta_standard_pdf/X-5G_prop}

```

```

359 \prop_new:c { g__pdfmeta_standard_pdf/X-5N_prop}
360 \prop_new:c { g__pdfmeta_standard_pdf/X-5PG_prop}
361 \prop_new:c { g__pdfmeta_standard_pdf/X-6_prop}
362 \prop_new:c { g__pdfmeta_standard_pdf/X-6N_prop}
363 \prop_new:c { g__pdfmeta_standard_pdf/X-6P_prop}
364
365 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/X-4_prop }
366 {
367   ,name          = PDF/X-4
368 }
369 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/X-4P_prop }
370 {
371   ,name          = PDF/X-4p
372 }
373 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/X-5G_prop }
374 {
375   ,name          = PDF/X-5g
376 }
377 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/X-5N_prop }
378 {
379   ,name          = PDF/X-5n
380 }
381 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/X-5PG_prop }
382 {
383   ,name          = PDF/X-5pg
384 }
385 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/X-6_prop }
386 {
387   ,name          = PDF/X-6
388 }
389 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/X-6N_prop }
390 {
391   ,name          = PDF/X-6n
392 }
393 \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/X-6P_prop }
394 {
395   ,name          = PDF/X-6p
396 }

```

(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

```

397 \AddToHook{begindocument/end}
398 {
399   \pdfmeta_standard_verify:nF{Catalog_EmbeddedFiles}
400   {
401     \tl_gput_right:Nn\g__kernel_pdfmanagement_end_run_code_tl
402     {
403       \pdfdict_if_empty:nT { g__pdf_Core/Catalog/Names/EmbeddedFiles }
404       {

```

```

405 \group_begin:
406 \pdfdict_put:nne {l_pdffile/Filespec} {Desc}{(note~about~PDF/A-4F)}
407 \pdfdict_put:nnn { l_pdffile/Filespec }{AFRelationship} { /Unspecified }
408 \pdffile_embed_stream:nnN
409 {The~document~was~declared~to~be~of~type~PDF/A-4f~but~hasn't~any~attachments.~
410 LaTeX~therefore~added~this~dummy~file.}
411 {pdf-A4f.txt}
412 \l__pdfmeta_tmpa_tl
413 \exp_args:Nne \__pdf_backend_Names_gpush:nn{EmbeddedFiles}{(pdf-A4f)~\l__pdfmeta_t
414 \group_end:
415 }
416 }
417 }
418 }

```

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.

```

419 \AddToHook{enddocument/end}
420 {
421   \pdfmeta_standard_verify:nF{no_embed_content}
422   {
423     \bool_lazy_or:nnT
424     { ! \int_if_zero_p:n { \g_pdffile_embed_pdfa_int } }
425     { ! \int_if_zero_p:n { \g_pdffile_embed_nonpdfa_int } }
426     {
427       \prop_get:NnNT\g__pdfmeta_standard_prop { name } \l__pdfmeta_tmpa_tl
428       {
429         \msg_warning:nne { pdf } { validation-failure }
430         {
431           Embedded-files-detected.\iow_newline:
432           This-is-not-allowed-in-standard~\l__pdfmeta_tmpa_tl
433         }
434       }
435     }
436   }
437   \pdfmeta_standard_verify:nF {only_pdfa_embed_content}
438   {
439     \int_if_zero:nF { \g_pdffile_embed_nonpdfa_int }
440     {
441       \prop_get:NnNT\g__pdfmeta_standard_prop { name } \l__pdfmeta_tmpa_tl
442       {
443         \str_if_eq:VnTF {\l__pdfmeta_tmpa_tl} { pdf/A-4 }
444         {
445           \msg_note:nne { pdf } { validation-failure }
446           {
447             Embedded-non-PDF-files-detected.\iow_newline:
448             Check-if-the-standard-must-be-changed~to~PDF/A-4F
449           }
450         }
451         {
452           \msg_warning:nne { pdf } { validation-failure }
453           {
454             Embedded-non-PDF-files-detected.\iow_newline:
455             This-is-not-allowed-in-standard~\l__pdfmeta_tmpa_tl

```

```

456     }
457   }
458 }
459 }
460 }
461 }

```

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 only color profile.

```
462 \prop_new:N \g_pdfmeta_outputintents_prop
```

(End of definition for `\g_pdfmeta_outputintents_prop`.)

Some keys to fill the property.

```

463 \keys_define:nn { document / metadata }
464 {
465   colorprofiles .code:n =
466   {
467     \keys_set:nn { document / metadata / colorprofiles }{#1}
468   }
469 }
470 \keys_define:nn { document / metadata / colorprofiles }
471 {
472   ,A .code:n =

```

```

473     {
474         \tl_if_blank:nF {#1}
475         {
476             \prop_gput:Nnn \g__pdfmeta_outputintents_prop
477             { GTS_PDFA1 } {#1}
478         }
479     }
480 ,a .code:n =
481     {
482         \tl_if_blank:nF {#1}
483         {
484             \prop_gput:Nnn \g__pdfmeta_outputintents_prop
485             { GTS_PDFA1 } {#1}
486         }
487     }
488 ,X .code:n =
489     {
490         \tl_if_blank:nF {#1}
491         {
492             \prop_gput:Nnn \g__pdfmeta_outputintents_prop
493             { GTS_PDFX } {#1}
494         }
495     }
496 ,x .code:n =
497     {
498         \tl_if_blank:nF {#1}
499         {
500             \prop_gput:Nnn \g__pdfmeta_outputintents_prop
501             { GTS_PDFX } {#1}
502         }
503     }
504 ,unknown .code:n =
505     {
506         \tl_if_blank:nF {#1}
507         {
508             \exp_args:NNo
509             \prop_gput:Nnn \g__pdfmeta_outputintents_prop
510             { \l_keys_key_str } {#1}
511         }
512     }
513 }

```

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

```

514 \pdfdict_new:n {l_pdfmeta/outputintent}
515 \pdfdict_put:nnn {l_pdfmeta/outputintent}
516 {Type}{/OutputIntent}
517 \prop_const_from_keyval:cn { c__pdfmeta_colorprofile_sRGB.icc}
518 {
519     ,OutputConditionIdentifier=IEC~sRGB
520     ,Info=IEC~61966-2.1~Default~RGB~colour~space~~~sRGB
521     ,RegistryName=http://www.iec.ch
522     ,N = 3
523 }

```

```

524 \prop_const_from_keyval:cn { c__pdfmeta_colorprofile_FOGRA39L_coated.icc}
525 {
526   ,OutputConditionIdentifier=FOGRA39L~Coated
527   ,Info={Offset~printing,~according~to~ISO~12647-2:2004/Amd~1,~OFCOM,~ %
528     paper~type~1~or~2~==~coated~art,~115~g/m2,~tone~value~increase~
529     curves~A~(CMY)~and~B~(K)}
530   ,RegistryName=http://www.fogra.org
531   ,N = 4
532 }

```

__pdfmeta_embed_colorprofile:n
 __pdfmeta_write_outputintent:nn

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

```

533 \cs_new_protected:Npn \__pdfmeta_embed_colorprofile:n #1#1 file name
534 {
535   \pdf_object_if_exist:nF { __color_icc_ #1 }
536   {
537     \pdf_object_new:n { __color_icc_ #1 }
538     \pdf_object_write:nne { __color_icc_ #1 } { fstream }
539     {
540       {/N\c_space_tl
541         \prop_item:cn{c__pdfmeta_colorprofile_#1}{N}
542       }
543       {#1}
544     }
545   }
546 }
547
548 \cs_new_protected:Npn \__pdfmeta_write_outputintent:nn #1 #2 % #1 file name, #2 subtype
549 {
550   \group_begin:
551   \pdfdict_put:nne {l_pdfmeta/outputintent}{S}{/\str_convert_pdfname:n{#2}}
552   \pdfdict_put:nne {l_pdfmeta/outputintent}
553     {DestOutputProfile}
554   {\pdf_object_ref:n{ __color_icc_ #1 }}
555   \clist_map_inline:nn { OutputConditionIdentifier, Info, RegistryName }
556   {
557     \prop_get:cnNT
558     { c__pdfmeta_colorprofile_#1 }
559     { ##1 }
560     \l__pdfmeta_tmpa_tl
561     {
562       \pdf_string_from_unicode:nVN {utf8/string}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_st
563       \pdfdict_put:nne
564       {l_pdfmeta/outputintent}{##1}{\l__pdfmeta_tmpa_str}
565     }
566   }
567   \pdf_object_unnamed_write:ne {dict}{\pdfdict_use:n {l_pdfmeta/outputintent} }
568   \pdfmanagement_add:nne {Catalog}{OutputIntents}{\pdf_object_ref_last:}
569   \group_end:
570 }

```

(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

```

571 \AddToHook{begindocument/end}
572 {
573   \pdfmeta_standard_verify:nTF {outputintent_A}
574   {
575     \prop_map_inline:Nn \g__pdfmeta_outputintents_prop
576     {
577       \prop_if_exist:cTF {c__pdfmeta_colorprofile_#2}
578       {
579         \__pdfmeta_embed_colorprofile:n
580         {#2}
581         \__pdfmeta_write_outputintent:nn
582         {#2}
583         {#1}
584       }
585       {
586         \msg_warning:nnn{pdfmeta}{colorprofile-undefined}{#2}
587       }
588     }
589   }

```

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.

```

590   {
591     \exp_args:NNe
592     \prop_if_in:NnF
593     \g__pdfmeta_outputintents_prop
594     { \pdfmeta_standard_item:n { outputintent_A } }
595     {
596       \exp_args:NNe
597       \prop_gput:Nnn
598       \g__pdfmeta_outputintents_prop
599       { \pdfmeta_standard_item:n { outputintent_A } }
600       { sRGB.icc }
601     }
602     \exp_args:NNe
603     \prop_get:NnN
604     \g__pdfmeta_outputintents_prop
605     { \pdfmeta_standard_item:n { outputintent_A } }
606     \l__pdfmeta_tmpb_tl
607     \prop_if_exist:cTF {c__pdfmeta_colorprofile_\l__pdfmeta_tmpb_tl}
608     {
609       \exp_args:NV \__pdfmeta_embed_colorprofile:n \l__pdfmeta_tmpb_tl
610       \prop_map_inline:Nn \g__pdfmeta_outputintents_prop
611       {
612         \exp_args:NV
613         \__pdfmeta_write_outputintent:nn
614         \l__pdfmeta_tmpb_tl
615         { #1 }
616       }
617     }
618     {
619       \msg_warning:nne{pdfmeta}{colorprofile-undefined}{\l__pdfmeta_tmpb_tl}

```

```

620     }
621   }
622 }

```

3.2 Regression test

This is simply a copy of the backend function.

```

623 \cs_new_protected:Npn \pdfmeta_set_regression_data:
624   { \__pdf_backend_set_regression_data: }

```

4 XMP-Metadata implementation

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

```

625 \bool_new:N\g__pdfmeta_xmp_bool
626 \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.

```

627 \hook_gput_code:nnn{pdfmanagement/add}{pdfmanagement}
628   {
629     \pdfmanagement_add:nne {Info}{Producer}{(\c_sys_engine_exec_str-\c_sys_engine_version_str)}
630     \pdfmanagement_add:nne {Info}{Creator}{(LaTeX)}
631   }

```

4.1 New document keys

```

632 \cs_generate_variant:Nn\pdf_version_gset:n{e}

```

if the pdf version is wrong for the standard we force the highest possible.

```

633 \cs_new_protected:Nn\__pdfmeta_force_standard_pdfversion:
634   {
635     \pdfmeta_standard_verify:nnF { min_pdf_version }
636     { \pdf_version: }
637     {
638       \pdfmeta_standard_verify:nTF { max_pdf_version }
639       {
640         \pdf_version_gset:n { 2.0 }
641       }
642       {
643         \pdf_version_gset:e{ \pdfmeta_standard_item:n{ max_pdf_version } }
644       }
645     }
646     \pdfmeta_standard_verify:nnF { max_pdf_version }
647     { \pdf_version: }
648     {
649       \pdf_version_gset:e{ \pdfmeta_standard_item:n{ max_pdf_version } }
650     }
651   }

```

At begin document we then want to test if there is a version problem.

```

652 \cs_new_protected:Npn \__pdfmeta_check_standard_pdfversion:
653 {
654   \clist_map_inline:nn{A,UA,X}
655   {
656     \pdfmeta_standard_family:nn { ##1 }
657     {
658       \pdfmeta_standard_verify:nnF { min_pdf_version }
659       { \pdf_version: }
660       { \msg_warning:nneee {pdf}{wrong-pdfversion} %TODO make error
661         {\pdf_version:}{low}
662         {
663           \pdfmeta_standard_item:n{name}
664         }
665       }
666       \pdfmeta_standard_verify:nnF { max_pdf_version }
667       { \pdf_version: }
668       { \msg_warning:nneee {pdf}{wrong-pdfversion}
669         {\pdf_version:}{high}
670         {
671           \pdfmeta_standard_item:n{name}
672         }
673       }
674     }
675   }
676 }

677 \keys_define:nn { document / metadata }
678 {
679   _pdfstandard .choices:nn =
680   {A-1B,A-2A,A-2B,A-2U,A-3A,A-3B,A-3U,A-4,A-4F,A-4E}
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}{#1}
686   },
687   _pdfstandard / unknown .code:n =
688   {
689     \msg_error:nnn{pdf}{unknown-standard}{#1}
690   },
691   _pdfstandard / X-4 .code:n =
692   {
693     \prop_gset_eq:Nc \g__pdfmeta_standard_X_prop { g__pdfmeta_standard_pdf/X-
694     4_prop }
695     \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-4}
696     \__pdfmeta_xmp_add_pdfxid:
697   },
698   _pdfstandard / X-4P .code:n =
699   {
700     \prop_gset_eq:Nc \g__pdfmeta_standard_X_prop { g__pdfmeta_standard_pdf/X-
701     4P_prop }
702     \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-4p}
703     \__pdfmeta_xmp_add_pdfxid:

```

```

702     },
703     _pdfstandard / X-5G .code:n =
704     {
705         \prop_gset_eq:Nc \g__pdfmeta_standard_X_prop { g__pdfmeta_standard_pdf/X-
5G_prop }
706         \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5g}
707         \__pdfmeta_xmp_add_pdfxid:
708     },
709     _pdfstandard / X-5N .code:n =
710     {
711         \prop_gset_eq:Nc \g__pdfmeta_standard_X_prop { g__pdfmeta_standard_pdf/X-
5N_prop }
712         \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5n}
713         \__pdfmeta_xmp_add_pdfxid:
714     },
715     _pdfstandard / X-5PG .code:n =
716     {
717         \prop_gset_eq:Nc \g__pdfmeta_standard_X_prop { g__pdfmeta_standard_pdf/X-
5PG_prop }
718         \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5pg}
719         \__pdfmeta_xmp_add_pdfxid:
720     },
721     _pdfstandard / X-6 .code:n =
722     {
723         \prop_gset_eq:Nc \g__pdfmeta_standard_X_prop { g__pdfmeta_standard_pdf/X-
6_prop }
724         \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6}
725         \__pdfmeta_xmp_add_pdfxid:
726     },
727     _pdfstandard / X-6N .code:n =
728     {
729         \prop_gset_eq:Nc \g__pdfmeta_standard_X_prop { g__pdfmeta_standard_pdf/X-
6N_prop }
730         \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6n}
731         \__pdfmeta_xmp_add_pdfxid:
732     },
733     _pdfstandard / X-6P .code:n =
734     {
735         \prop_gset_eq:Nc \g__pdfmeta_standard_X_prop { g__pdfmeta_standard_pdf/X-
6P_prop }
736         \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6p}
737         \__pdfmeta_xmp_add_pdfxid:
738     },
739     _pdfstandard / UA-1 .code:n =
740     {
741         \prop_gset_eq:Nc \g__pdfmeta_standard_UA_prop { g__pdfmeta_standard_pdf/UA-
1_prop }
742         \pdfmeta_standard_family:nn{UA}{ \__pdfmeta_force_standard_pdfversion: }
743         \AddToDocumentProperties [document]{pdfstandard-UA}{1}{1}{}
744         \AddToHook{begindocument/before}
745         {
746             \prop_gput:Nnn \g__pdfmeta_standard_prop {omit_CID}{}
747         }
748     },

```

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.

```

749   _pdfstandard / UA-2 .code:n =
750   {
751     \prop_gset_eq:Nc \g__pdfmeta_standard_UA_prop { g__pdfmeta_standard_pdf/UA-
2_prop }
752     \pdfmeta_standard_family:nn{UA}{\_pdfmeta_force_standard_pdfversion:}
753     \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.

```

754   %\AddToHook{begindocument/before}
755   %{\prop_gput:Nnn \g__pdfmeta_standard_prop {Trailer_no_Info}{}}
756   \AddToHook{begindocument/before}
757   {
758     \__pdfmeta_xmp_wtpdf_accessibility_declaration:
759     \__pdfmeta_xmp_wtpdf_reuse_declaration:
760     \cs_set_eq:NN\__pdfmeta_xmp_wtpdf_accessibility_declaration:\prg_do_nothing:
761     \cs_set_eq:NN\__pdfmeta_xmp_wtpdf_reuse_declaration:\prg_do_nothing:
762   }
763   },
764   xmp .choice:,
765   xmp / true .code:n = { \bool_gset_true:N \g__pdfmeta_xmp_bool },
766   xmp / false .code:n = { \bool_gset_false:N \g__pdfmeta_xmp_bool},
767   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.

```

768   xmp / wtpdf .code:n =
769   {
770     \keys_set:nn {__pdfmeta/xmp}{#1}
771   },
772   }
773   \keys_define:nn {__pdfmeta/xmp}
774   {
775     reuse .choice:,
776     reuse / true .code:n = \__pdfmeta_xmp_wtpdf_reuse_declaration:,
777     reuse / false .code:n =
778     {
779       \cs_set_eq:NN \__pdfmeta_xmp_wtpdf_reuse_declaration: \prg_do_nothing:
780     },
781     accessibility .choice:,
782     accessibility / true .code:n = \__pdfmeta_xmp_wtpdf_accessibility_declaration:,
783     accessibility /false .code:n =
784     {
785       \cs_set_eq:NN \__pdfmeta_xmp_wtpdf_accessibility_declaration: \prg_do_nothing:
786     },
787   }

```

XMP debugging option

```

788   \bool_new:N \g__pdfmeta_xmp_export_bool
789   \str_new:N \g__pdfmeta_xmp_export_str
790

```

```

791 \keys_define:nn { document / metadata }
792 {
793   ,debug .code:n =
794   {
795     \keys_set:nn { document / metadata / debug } {#1}
796   }
797   ,debug / xmp-export .choice:
798   ,debug / xmp-export / true .code:n=
799   {
800     \bool_gset_true:N \g__pdfmeta_xmp_export_bool
801     \str_gset_eq:NN \g__pdfmeta_xmp_export_str \c_sys_jobname_str
802   }
803   ,debug / xmp-export / false .code:n =
804   {
805     \bool_gset_false:N \g__pdfmeta_xmp_export_bool
806   }
807   ,debug / xmp-export / unknown .code:n =
808   {
809     \bool_gset_true:N \g__pdfmeta_xmp_export_bool
810     \str_gset:Nn \g__pdfmeta_xmp_export_str { #1 }
811   }
812   ,debug / xmp-export .default:n = true
813 }

```

4.2 Messages

```

814 \msg_new:nnn{pdfmeta}{xmp-defined}{The~XMP~#1~`#2`~is~already~declared}
815 \msg_new:nnn{pdfmeta}{xmp-undefined}{The~XMP~#1~`#2`~is~undefined}
816 \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:`

```

817 \bool_lazy_or:nnTF
818 { \sys_if_engine luatex_p: }
819 { \sys_if_engine xetex_p: }
820 {
821   \cs_new:Npn \__pdfmeta_xmp_generate_bom:
822   { \char_generate:nn {"FEFF"}{12} }
823 }
824 {
825   \cs_new:Npn \__pdfmeta_xmp_generate_bom:
826   {
827     \char_generate:nn {"EF"}{12}
828     \char_generate:nn {"BB"}{12}
829     \char_generate:nn {"BF"}{12}
830   }
831 }

```

(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
832 \int_new:N \l__pdfmeta_xmp_indent_int
(End of definition for \l__pdfmeta_xmp_indent_int.)

__pdfmeta_xmp_indent:
__pdfmeta_xmp_indent:n 833 \cs_new:Npn __pdfmeta_xmp_indent:
__pdfmeta_xmp_incr_indent: 834 {
__pdfmeta_xmp_decr_indent: 835   \iow_newline:
836   \prg_replicate:nn {\l__pdfmeta_xmp_indent_int}{\c_space_tl}
837 }
838
839 \cs_new:Npn __pdfmeta_xmp_indent:n #1
840 {
841   \iow_newline:
842   \prg_replicate:nn {#1}{\c_space_tl}
843 }
844
845 \cs_new_protected:Npn __pdfmeta_xmp_incr_indent:
846 {
847   \int_incr:N \l__pdfmeta_xmp_indent_int
848 }
849
850 \cs_new_protected:Npn __pdfmeta_xmp_decr_indent:
851 {
852   \int_decr:N \l__pdfmeta_xmp_indent_int
853 }
(End of definition for __pdfmeta_xmp_indent: and others.)
```

4.3.3 Date and time handling

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

```
\l__pdfmeta_xmp_date_regex
854 \regex_new:N \l__pdfmeta_xmp_date_regex
855 \regex_set:Nn \l__pdfmeta_xmp_date_regex
856 {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.

```
857 \cs_new_protected:Npn __pdfmeta_xmp_date_split:nN #1 #2 %#1 date, #2 seq
858 {
859   \regex_split:NnN \l__pdfmeta_xmp_date_regex {#1} #2
860 }
861 \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.

```

862 \cs_new:Npn \_pdfmeta_xmp_print_date:N #1 % seq
863 {
864   \tl_if_blank:eTF { \seq_item:Nn #1 {1} }
865   {
866     \seq_item:Nn #1 {2} %year
867     -
868     \seq_item:Nn #1 {3} %month
869     -
870     \seq_item:Nn #1 {4} % day
871     \tl_if_blank:eF
872     { \seq_item:Nn #1 {5} }
873     { T \seq_item:Nn #1 {5} } %hour
874     \tl_if_blank:eF
875     { \seq_item:Nn #1 {6} }
876     { : \seq_item:Nn #1 {6} } %minutes
877     \tl_if_blank:eF
878     { \seq_item:Nn #1 {7} }
879     { : \seq_item:Nn #1 {7} } %seconds
880     \seq_item:Nn #1 {8} %Z,+,-
881     \seq_item:Nn #1 {9}
882     \tl_if_blank:eF
883     { \seq_item:Nn #1 {10} }
884     { : \seq_item:Nn #1 {10} }
885   }
886   {
887     \seq_item:Nn #1 {1}
888   }
889 }

```

(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 the regex.

```

890 \tl_new:N \l_pdfmeta_xmp_currentdate_tl
891 \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.

```

892 \cs_new_protected:Npn \_pdfmeta_xmp_date_get:nNN #1 #2 #3
893   % #1 property, #2 tl var with PDF date, #3 seq for split date
894   {
895     \tl_set:Ne #2 { \pdfmanagement_get_documentproperty:n{#1} }
896     \tl_if_blank:VTF #2
897     {
898       \seq_set_eq:NN #3 \l_pdfmeta_xmp_currentdate_seq
899       \tl_set_eq:NN #2 \l_pdfmeta_xmp_currentdate_tl
900     }
901     {
902       \_pdfmeta_xmp_date_split:VN #2 #3
903     }
904   }

```

(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
905 \cs_new_protected:Npn \_pdfmeta_xmp_create_uuid:nN #1 #2
906 {
907   \str_set:Ne#2 {\str_lowercase:f{\tex_mdffivesum:D{#1}}}
908   \str_set:Ne#2
909     {
910       \str_range:Nnn #2{1}{8}
911       -\str_range:Nnn#2{9}{12}
912       -4\str_range:Nnn#2{13}{15}
913       -8\str_range:Nnn#2{16}{18}
914       -\str_range:Nnn#2{19}{30}
915     }
916 }
```

(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.

```
917 \cs_new_protected:Npn \_pdfmeta_xmp_sanitize:nN #1 #2
918 %#1 input string, #2 str with the output
919 {
920   \group_begin:
921   \text_declare_purify_equivalent:Nn \& {\tl_to_str:N & }
922   \text_declare_purify_equivalent:Nn \texttilde {\c_tilde_str}
923   \tl_set:Ne \l__pdfmeta_tmpa_tl { \text_purify:n {#1} }
924   \str_gset:Ne \g__pdfmeta_tmpa_str { \tl_to_str:N \l__pdfmeta_tmpa_tl }
925   \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {\&}{\&amp;}
926   \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {<}{\&lt;}
927   \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {>}{\&gt;}
928   \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {"}{\&quot;}
929   \group_end:
930   \str_set_eq:NN #2 \g__pdfmeta_tmpa_str
931 }
932
933 \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
934 \tl_new:N \l__pdfmeta_xmp_doclang_tl
935 \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

```
936 \regex_new:N\l__pdfmeta_xmp_lang_regex
937 \regex_set:Nn\l__pdfmeta_xmp_lang_regex {\A\[([A-Za-z-]+\)](.*)}
(End of definition for \l__pdfmeta_xmp_lang_regex.)

938 \cs_new_protected:Npn \__pdfmeta_xmp_lang_get:nNN #1 #2 #3
939 % #1 text, #2 t1 var for lang match (or default), #3 t1 var for text
940 {
941   \regex_extract_once:NnN \l__pdfmeta_xmp_lang_regex {#1}\l__pdfmeta_tmpa_seq
942   \seq_if_empty:NTF \l__pdfmeta_tmpa_seq
943   {
944     \tl_set:Nn #2 \l__pdfmeta_xmp_metalang_t1
945     \tl_set:Nn #3 {#1}
946   }
947   {
948     \tl_set:Ne #2 {\seq_item:Nn\l__pdfmeta_tmpa_seq{2}}
949     \tl_set:Ne #3 {\seq_item:Nn\l__pdfmeta_tmpa_seq{3}}
950   }
951 }
952 \cs_generate_variant:Nn \__pdfmeta_xmp_lang_get:nNN {eNN,VNN}
```

4.5 Filling the packet

This t1 var that holds the whole packet

\g__pdfmeta_xmp_packet_t1

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

4.5.1 Helper commands to add lines and lists

__pdfmeta_xmp_add_packet_chunk:n

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

```
954 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_chunk:n #1
955 {
956   \tl_gput_right:Ne\g__pdfmeta_xmp_packet_t1
957   {
958     \__pdfmeta_xmp_indent: \exp_not:n{#1}
959   }
960 }
961 \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.

```
962 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_chunk:nN #1 #2
963 {
964   \tl_put_right:Ne#2
965   {
966     \__pdfmeta_xmp_indent: \exp_not:n{#1}
967   }
968 }
969 \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.

```

970 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_open:nn #1 #2 %1 prefix #2 name
971 {
972   \__pdfmeta_xmp_add_packet_chunk:n {<#1:#2>}
973   \__pdfmeta_xmp_incr_indent:
974 }
975 \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.

```

976 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_open_attr:nnn #1 #2 #3
977 %1 prefix #2 name #3 attr
978 {
979   \__pdfmeta_xmp_add_packet_chunk:n {<#1:#2~#3>}
980   \__pdfmeta_xmp_incr_indent:
981 }
982 \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.

```

983 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_close:nn #1 #2 %1 prefix #2:name
984 {
985   \__pdfmeta_xmp_decr_indent:
986   \__pdfmeta_xmp_add_packet_chunk:n {</#1:#2>}
987 }

```

(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.

```

988 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_line:nnn #1 #2 #3
989 %1 prefix #2 name #3 content
990 {
991   \tl_if_blank:nF {#3}
992   {
993     \__pdfmeta_xmp_sanitizize:nN {#3}\l__pdfmeta_tmpa_str
994     \__pdfmeta_xmp_add_packet_chunk:e {<#1:#2>\l__pdfmeta_tmpa_str</#1:#2>}
995   }
996 }
997 \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.

```

998 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_line:nnnN #1 #2 #3 #4
999 %1 prefix #2 name #3 content #4 tl_var to prebuilt.
1000 {
1001   \tl_if_blank:nF {#3}
1002   {

```

```

1003     \__pdfmeta_xmp_sanitiz:eN {#3}\l__pdfmeta_tmpa_str
1004     \__pdfmeta_xmp_add_packet_chunk:eN {<#1:#2>\l__pdfmeta_tmpa_str</#1:#2>} #4
1005   }
1006 }
1007 \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

```

1008 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_line_attr:nnnn #1 #2 #3 #4
1009 % #1 prefix #2 name #3 attribute #4 content
1010 {
1011   \tl_if_blank:nF {#4}
1012   {
1013     \__pdfmeta_xmp_sanitiz:eN {#4}\l__pdfmeta_tmpa_str
1014     \__pdfmeta_xmp_add_packet_chunk:e {<#1:#2~#3>\l__pdfmeta_tmpa_str</#1:#2>}
1015   }
1016 }
1017 \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

```

1018 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_line_default:nnnn #1 #2 #3 #4
1019 % #1 prefix #2 name #3 default #4 content
1020 {
1021   \tl_if_blank:nTF { #4 }
1022   {
1023     \tl_set:Nn \l__pdfmeta_tmpa_tl {#3}
1024   }
1025   {
1026     \tl_set:Nn \l__pdfmeta_tmpa_tl {#4}
1027   }
1028   \__pdfmeta_xmp_add_packet_line:nnV {#1}{#2}\l__pdfmeta_tmpa_tl
1029 }
1030 \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:

```

1031 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_list_simple:nnnn #1 #2 #3 #4
1032 % #1 prefix, #2 name, #3 type (Seq/Bag/Alt) #4 a clist
1033 {
1034   \clist_if_empty:nF { #4 }
1035   {
1036     \__pdfmeta_xmp_add_packet_open:nn {#1}{#2}
1037     \__pdfmeta_xmp_add_packet_open:nn {rdf}{#3}
1038     \clist_map_inline:nn {#4}
1039     {
1040       \__pdfmeta_xmp_add_packet_line:nnn
1041       {rdf}{li}{##1}
1042     }
1043     \__pdfmeta_xmp_add_packet_close:nn{rdf}{#3}
1044     \__pdfmeta_xmp_add_packet_close:nn {#1}{#2}

```

```

1045     }
1046   }
1047   \cs_generate_variant:Nn \__pdfmeta_xmp_add_packet_list_simple:nnnn {nnnV,nnne}

```

Here we check also for the language.

```

1048   \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_list:nnnn #1 #2 #3 #4
1049     {%#1 prefix, #2 name, #3 type (Seq/Bag/Alt) #4 a clist
1050     {
1051       \clist_if_empty:nF { #4 }
1052       {
1053         \__pdfmeta_xmp_add_packet_open:nn {#1}{#2}
1054         \__pdfmeta_xmp_add_packet_open:nn {rdf}{#3}
1055         \clist_map_inline:nn {#4}
1056         {
1057           \__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.

```

1058           \tl_if_eq:eeTf{\l__pdfmeta_tmpa_tl}{\l__pdfmeta_xmp_metalang_tl}
1059           {
1060             \__pdfmeta_xmp_add_packet_line_attr:nneV
1061             {rdf}{li}{xml:lang="x-default" }\l__pdfmeta_tmpb_tl
1062             \__pdfmeta_xmp_add_packet_line_attr:nneV
1063             {rdf}{li}{xml:lang="\l__pdfmeta_tmpa_tl" }\l__pdfmeta_tmpb_tl
1064           }
1065         }
1066       \clist_map_inline:nn {#4}
1067       {
1068         \__pdfmeta_xmp_lang_get:nNN {##1}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl
1069         \tl_if_eq:eeFf{\l__pdfmeta_tmpa_tl}{\l__pdfmeta_xmp_metalang_tl}
1070         {
1071           \__pdfmeta_xmp_add_packet_line_attr:nneV
1072           {rdf}{li}{xml:lang="\l__pdfmeta_tmpa_tl" }\l__pdfmeta_tmpb_tl
1073         }
1074       }
1075       \__pdfmeta_xmp_add_packet_close:nn{rdf}{#3}
1076       \__pdfmeta_xmp_add_packet_close:nn {#1}{#2}
1077     }
1078   }
1079   \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.

```

1080   \cs_new_protected:Npn \__pdfmeta_xmp_build_packet:
1081     {

```

Get the main languages

```

1082     \tl_set:Ne \l__pdfmeta_xmp_doclang_tl {\pdfmanagement_get_documentproperty:n{document/la
1083     \tl_set:Ne \l__pdfmeta_xmp_metalang_tl {\pdfmanagement_get_documentproperty:n{hyperref/pd

```

```

1084 \tl_if_blank:VT \l__pdfmeta_xmp_metalang_tl
1085 { \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

```

1086 \__pdfmeta_xmp_build_iptc_data:N \l__pdfmeta_xmp_iptc_data_tl
1087 \tl_if_empty:NT \l__pdfmeta_xmp_iptc_data_tl
1088 {
1089   \seq_remove_all:Nn \l__pdfmeta_xmp_schema_seq { Iptc4xmpCore }
1090 }

```

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

```

1091 \__pdfmeta_xmp_add_packet_chunk:e
1092 {<?xpacket~begin="\__pdfmeta_xmp_generate_bom:"~id="W5MOMpCehiHzreSzNTczkc9d"?>}
1093 \__pdfmeta_xmp_add_packet_open:nn{x}{xmpmeta~xmlns:x="adobe:ns:meta/"}
1094 \__pdfmeta_xmp_add_packet_open:ne{rdf}
1095 {RDF~xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns:c_hash_str"}

```

The rdf namespaces

```

1096 \__pdfmeta_xmp_add_packet_open_attr:nne
1097 {rdf}{Description}{rdf:about="" \g__pdfmeta_xmp_xmlns_tl}

```

The extensions

```

1098 \__pdfmeta_xmp_add_packet_open:nn{pdfaExtension}{schemas}
1099 \__pdfmeta_xmp_add_packet_open:nn {rdf}{Bag}
1100 \seq_map_inline:Nn \l__pdfmeta_xmp_schema_seq
1101 {
1102   \tl_use:c { g__pdfmeta_xmp_schema_##1_tl }
1103 }
1104 \__pdfmeta_xmp_add_packet_close:nn {rdf}{Bag}
1105 \__pdfmeta_xmp_add_packet_close:nn {pdfaExtension}{schemas}

```

Now starts the part with the data.

```

1106 % data
1107 \__pdfmeta_xmp_build_pdf:
1108 \__pdfmeta_xmp_build_xmpRights:
1109 \__pdfmeta_xmp_build_standards: %pdfaid,pdfxid,pdfuaid
1110 \__pdfmeta_xmp_build_pdfd:
1111 \__pdfmeta_xmp_build_dc:
1112 \__pdfmeta_xmp_build_photoshop:
1113 \__pdfmeta_xmp_build_xmp:
1114 \__pdfmeta_xmp_build_xmpMM:
1115 \__pdfmeta_xmp_build_prism:
1116 \__pdfmeta_xmp_build_iptc:
1117 \__pdfmeta_xmp_build_tdm:
1118 \__pdfmeta_xmp_build_user: %user additions
1119 % end
1120 \__pdfmeta_xmp_add_packet_close:nn {rdf}{Description}
1121 \__pdfmeta_xmp_add_packet_close:nn {rdf}{RDF}
1122 \__pdfmeta_xmp_add_packet_close:nn {x}{xmpmeta}
1123 \int_set:Nn \l__pdfmeta_xmp_indent_int{20}
1124 \prg_replicate:nn{10}{\__pdfmeta_xmp_add_packet_chunk:n {}}
1125 \int_zero:N \l__pdfmeta_xmp_indent_int
1126 \__pdfmeta_xmp_add_packet_chunk:n {<?xpacket~end="w"?>}
1127 }

```

(End of definition for _pdfmeta_xmp_build_packet:.)

4.6 Building the chunks: rdf namespaces

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

```
\g__pdfmeta_xmp_xmlns_tl  
\g__pdfmeta_xmp_xmlns_prop
```

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

```
1128 \str_new:N \g__pdfmeta_xmp_xmlns_tl  
1129 \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
```

```
1130 \cs_new_protected:Npn \_pdfmeta_xmp_xmlns_new:nn #1 #2  
1131 {  
1132   \prop_gput:Nnn \g__pdfmeta_xmp_xmlns_prop {#1}{#2}  
1133   \tl_gput_right:Ne \g__pdfmeta_xmp_xmlns_tl  
1134   {  
1135     \_pdfmeta_xmp_indent:n{4} xmlns:\exp_not:n{#1="#2"}  
1136   }  
1137 }
```

(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.

```
1138 \_pdfmeta_xmp_xmlns_new:nn {pdf}      {http://ns.adobe.com/pdf/1.3/}  
1139 \_pdfmeta_xmp_xmlns_new:nn {xmpRights}{http://ns.adobe.com/xap/1.0/rights/}  
1140 \_pdfmeta_xmp_xmlns_new:nn {dc}       {http://purl.org/dc/elements/1.1/}  
1141 \_pdfmeta_xmp_xmlns_new:nn {photoshop}{http://ns.adobe.com/photoshop/1.0/}  
1142 \_pdfmeta_xmp_xmlns_new:nn {xmp}     {http://ns.adobe.com/xap/1.0/}  
1143 \_pdfmeta_xmp_xmlns_new:nn {xmpMM}   {http://ns.adobe.com/xap/1.0/mm/}  
1144 \_pdfmeta_xmp_xmlns_new:nn {stEvt}   {http://ns.adobe.com/xap/1.0/sType/ResourceEvent\c_hash_str}  
1145 \_pdfmeta_xmp_xmlns_new:nn {pdfaid}   {http://www.aiim.org/pdfa/ns/id/}  
1146 \_pdfmeta_xmp_xmlns_new:nn {pdfuaid}  {http://www.aiim.org/pdfua/ns/id/}  
1147 \_pdfmeta_xmp_xmlns_new:nn {pdfx}    {http://ns.adobe.com/pdfx/1.3/}  
1148 \_pdfmeta_xmp_xmlns_new:nn {pdfxid}   {http://www.npes.org/pdfx/ns/id/}  
1149 \_pdfmeta_xmp_xmlns_new:nn {prism}    {http://prismstandard.org/namespaces/basic/3.0/}  
1150 \_pdfmeta_xmp_xmlns_new:nn {jav}      {http://www.niso.org/schemas/jav/1.0/}  
1151 \_pdfmeta_xmp_xmlns_new:nn {xmpTPg}   {http://ns.adobe.com/xap/1.0/t/pg/}  
1152 \_pdfmeta_xmp_xmlns_new:nn {stFnt}    {http://ns.adobe.com/xap/1.0/sType/Font\c_hash_str}  
1153 \_pdfmeta_xmp_xmlns_new:nn {Iptc4xmpCore}{http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}  
1154 \_pdfmeta_xmp_xmlns_new:nn {pdfaExtension}{http://www.aiim.org/pdfa/ns/extension/}  
1155 \_pdfmeta_xmp_xmlns_new:nn {pdfaSchema}{http://www.aiim.org/pdfa/ns/schema\c_hash_str}  
1156 \_pdfmeta_xmp_xmlns_new:nn {pdfaProperty}{http://www.aiim.org/pdfa/ns/property\c_hash_str}  
1157 \_pdfmeta_xmp_xmlns_new:nn {pdfaType} {http://www.aiim.org/pdfa/ns/type\c_hash_str}  
1158 \_pdfmeta_xmp_xmlns_new:nn {pdfaField}{http://www.aiim.org/pdfa/ns/field\c_hash_str}  
1159 \_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

```
1161 \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.

```
1162 \cs_new_protected:Npn \__pdfmeta_xmp_schema_new:nnn #1 #2 #3
1163   {%#1 name #2 prefix, #3 text
1164   {
1165     \tl_if_exist:cTF { g__pdfmeta_xmp_schema_#2_tl }
1166     {
1167       \msg_warning:nnnn{pdfmeta}{xmp-defined}{schema}{#2}
1168     }
1169     {
1170       \seq_put_right:Nn \l__pdfmeta_xmp_schema_seq { #2 }
1171       \tl_new:c { g__pdfmeta_xmp_schema_#2_tl }
1172       \tl_new:c { g__pdfmeta_xmp_schema_#2_properties_tl }
1173       \tl_gput_right:cn { g__pdfmeta_xmp_schema_#2_tl }
1174       {
1175         \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1176         \__pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{schema}{#1}
1177         \__pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{prefix}{#2}
1178         \__pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{namespaceURI}{#3}
1179         \__pdfmeta_xmp_add_packet_open:nn {pdfaSchema}{property}
1180         \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1181         \tl_use:c { g__pdfmeta_xmp_schema_#2_properties_tl }
1182         \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1183         \__pdfmeta_xmp_add_packet_close:nn {pdfaSchema}{property}
1184         \cs_if_exist_use:c {__pdfmeta_xmp_schema_#2_additions:}
1185         \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1186       }
1187     }
1188   }
```

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

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

```
1189 \prop_new:N\g__pdfmeta_xmp_schema_property_prop
1190 \cs_new_protected:Npn \__pdfmeta_xmp_property_new:nnnnn #1 #2 #3 #4 #5 %
1191   {%#1 schema #2 name, #3 type, #4 category #5 description
1192   {
1193     \tl_if_exist:cTF { g__pdfmeta_xmp_schema_#1_properties_tl }
1194     {
1195       \prop_get:NeNF \g__pdfmeta_xmp_schema_property_prop {#1:#2}\l__pdfmeta_tmpa_tl
```

```

1196     {
1197         \prop_gput:Nee \g__pdfmeta_xmp_schema_property_prop {#1:#2}{#3}
1198         \tl_gput_right:cn { g__pdfmeta_xmp_schema_#1_properties_tl }
1199         {
1200             \__pdfmeta_xmp_add_packet_open:nn {rdf}{li~rdf:parseType="Resource"}
1201             \__pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{name}{#2}
1202             \__pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{valueType}{#3}
1203             \__pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{category}{#4}
1204             \__pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{description}{#5}
1205             \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1206         }
1207     }
1208 }
1209 {
1210     \msg_warning:nnnn{pdfmeta}{xmp-undefined}{schema}{#1}
1211 }
1212 }

```

(End of definition for __pdfmeta_xmp_property_new:nnnnn.)

__pdfmeta_xmp_add_packet_field:nnn This adds a field to a schema.

```

1213 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_field:nnn #1 #2 #3 %
1214 %#1 name #2 valuetype #3 description
1215 {
1216     \__pdfmeta_xmp_add_packet_open_attr:nnn {rdf}{li}{rdf:parseType="Resource"}
1217     \__pdfmeta_xmp_add_packet_line:nnn {pdfaField}{name}{#1}
1218     \__pdfmeta_xmp_add_packet_line:nnn {pdfaField}{valueType}{#2}
1219     \__pdfmeta_xmp_add_packet_line:nnn {pdfaField}{description}{#3}
1220     \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1221 }

```

(End of definition for __pdfmeta_xmp_add_packet_field:nnn.)

4.7.1 The extension data

The list of extension has been reviewed and compared with the list of namespaces which can be used in pdf/A-1⁷

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

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

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

```

1222 \__pdfmeta_xmp_schema_new:nnn
1223 {XMP~Media~Management~Schema}
1224 {xmpMM}
1225 {http://ns.adobe.com/xap/1.0/mm/}

```

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.

```

1226 \_pdfmeta_xmp_property_new:nnnnn
1227 {xmpMM}
1228 {OriginalDocumentID}
1229 {URI}
1230 {internal}
1231 {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)

```

1232 \_pdfmeta_xmp_schema_new:nnn
1233 {PDF/A~Identification~Schema}
1234 {pdfaid}
1235 {http://www.aiim.org/pdfa/ns/id/}
1236 \_pdfmeta_xmp_property_new:nnnnn
1237 {pdfaid}
1238 {year}
1239 {Integer}
1240 {internal}
1241 {Year~of~standard}
1242 \_pdfmeta_xmp_property_new:nnnnn
1243 {pdfaid}
1244 {rev}
1245 {Integer}
1246 {internal}
1247 {Revision~year~of~standard}

```

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

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

pdfuaid~(schema)

```

1248 \_pdfmeta_xmp_schema_new:nnn
1249 {PDF/UA~Universal~Accessibility~Schema}
1250 {pdfuaid}
1251 {http://www.aiim.org/pdfua/ns/id/}
1252 \_pdfmeta_xmp_property_new:nnnnn
1253 {pdfuaid}
1254 {part}
1255 {Integer}
1256 {internal}
1257 {Part~of~ISO~14289~standard}
1258 \_pdfmeta_xmp_property_new:nnnnn
1259 {pdfuaid}
1260 {rev}
1261 {Integer}
1262 {internal}
1263 {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)

```

1264 \cs_new_protected:Npn \__pdfmeta_xmp_add_pdfxid:
1265 {
1266   \__pdfmeta_xmp_xmlns_new:nn {pdfxid}    {http://www.npes.org/pdfx/ns/id/}
1267   \__pdfmeta_xmp_schema_new:nnn
1268     {PDF/X-ID-Schema}
1269     {pdfxid}
1270     {http://www.npes.org/pdfx/ns/id/}
1271   \__pdfmeta_xmp_property_new:nnnnn
1272     {pdfxid}
1273     {GTS_PDFXVersion}
1274     {Text}
1275     {internal}
1276     {ID-of-PDF/X-standard}
1277 }

```

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

prism~(schema)

```

1278 \__pdfmeta_xmp_schema_new:nnn
1279   {PRISM-Basic-Metadata}
1280   {prism}
1281   {http://prismstandard.org/namespaces/basic/3.0/}
1282 \__pdfmeta_xmp_property_new:nnnnn
1283   {prism}
1284   {complianceProfile}
1285   {Text}
1286   {internal}
1287   {PRISM-specification-compliance-profile-to-which-this-document-adheres}
1288 \__pdfmeta_xmp_property_new:nnnnn
1289   {prism}
1290   {publicationName}
1291   {Text}
1292   {external}
1293   {Publication-name}
1294 \__pdfmeta_xmp_property_new:nnnnn
1295   {prism}
1296   {aggregationType}
1297   {Text}
1298   {external}
1299   {Publication-type}
1300 \__pdfmeta_xmp_property_new:nnnnn
1301   {prism}
1302   {bookEdition}
1303   {Text}

```

```

1304 {external}
1305 {Edition~of~the~book~in~which~the~document~was~published}
1306 \_pdfmeta_xmp_property_new:nnnnn
1307 {prism}
1308 {volume}
1309 {Text}
1310 {external}
1311 {Publication~volume~number}
1312 \_pdfmeta_xmp_property_new:nnnnn
1313 {prism}
1314 {number}
1315 {Text}
1316 {external}
1317 {Publication~issue~number~within~a~volume}
1318 \_pdfmeta_xmp_property_new:nnnnn
1319 {prism}
1320 {pageRange}
1321 {Text}
1322 {external}
1323 {Page~range~for~the~document~within~the~print~version~of~its~publication}
1324 \_pdfmeta_xmp_property_new:nnnnn
1325 {prism}
1326 {issn}
1327 {Text}
1328 {external}
1329 {ISSN~for~the~printed~publication~in~which~the~document~was~published}
1330 \_pdfmeta_xmp_property_new:nnnnn
1331 {prism}
1332 {eIssn}
1333 {Text}
1334 {external}
1335 {ISSN~for~the~electronic~publication~in~which~the~document~was~published}
1336 \_pdfmeta_xmp_property_new:nnnnn
1337 {prism}
1338 {isbn}
1339 {Text}
1340 {external}
1341 {ISBN~for~the~publication~in~which~the~document~was~published}
1342 \_pdfmeta_xmp_property_new:nnnnn
1343 {prism}
1344 {doi}
1345 {Text}
1346 {external}
1347 {Digital~Object~Identifier~for~the~document}
1348 \_pdfmeta_xmp_property_new:nnnnn
1349 {prism}
1350 {url}
1351 {URL}
1352 {external}
1353 {URL~at~which~the~document~can~be~found}
1354 \_pdfmeta_xmp_property_new:nnnnn
1355 {prism}
1356 {byteCount}
1357 {Integer}

```

```

1358 {internal}
1359 {Approximate~file~size~in~octets}
1360 \_pdfmeta_xmp_property_new:nnnnn
1361 {prism}
1362 {pageCount}
1363 {Integer}
1364 {internal}
1365 {Number~of~pages~in~the~print~version~of~the~document}
1366 \_pdfmeta_xmp_property_new:nnnnn
1367 {prism}
1368 {subtitle}
1369 {Text}
1370 {external}
1371 {Document's~subtitle}

```

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

iptc (schema)

```

1372 \_pdfmeta_xmp_schema_new:nnn
1373 {IPTC~Core~Schema}
1374 {Iptc4xmpCore}
1375 {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
1376 \_pdfmeta_xmp_property_new:nnnnn
1377 {Iptc4xmpCore}
1378 {CreatorContactInfo}
1379 {ContactInfo}
1380 {external}
1381 {Document~creator's~contact~information}
1382 \cs_new_protected:cpn { \_pdfmeta_xmp_schema_Iptc4xmpCore_additions: }
1383 {
1384   \_pdfmeta_xmp_add_packet_open:nn{pdfaSchema}{valueType}
1385   \_pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1386   \_pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1387   \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{ContactInfo}
1388   \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
1389   {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
1390   \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{Iptc4xmpCore}
1391   \_pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
1392   {Basic~set~of~information~to~get~in~contact~with~a~person}
1393   \_pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
1394   \_pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1395   \_pdfmeta_xmp_add_packet_field:nnn{CiAdrCity}{Text}
1396   {Contact~information~city}
1397   \_pdfmeta_xmp_add_packet_field:nnn{CiAdrCtry}{Text}
1398   {Contact~information~country}
1399   \_pdfmeta_xmp_add_packet_field:nnn{CiAdrExtadr}{Text}
1400   {Contact~information~address}
1401   \_pdfmeta_xmp_add_packet_field:nnn{CiAdrPcode}{Text}
1402   {Contact~information~local~postal~code}
1403   \_pdfmeta_xmp_add_packet_field:nnn{CiAdrRegion}{Text}
1404   {Contact~information~regional~information~such~as~state~or~province}
1405   \_pdfmeta_xmp_add_packet_field:nnn{CiEmailWork}{Text}
1406   {Contact~information~email~address(es)}
1407   \_pdfmeta_xmp_add_packet_field:nnn{CiTelWork}{Text}

```

```

1408         {Contact~information~telephone-number(s)}
1409         \_pdfmeta_xmp_add_packet_field:nnn{CiUrlWork}{Text}
1410         {Contact~information~Web~URL(s)}
1411         \_pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1412         \_pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
1413         \_pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1414         \_pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1415         \_pdfmeta_xmp_add_packet_close:nn{pdfaSchema}{valueType}
1416     }

```

(End of definition for iptc (schema).)

jav : currently ignored

tdmrep (schema)

```

1417 \_pdfmeta_xmp_schema_new:nnn
1418 {TDMRep}
1419 {tdm}
1420 {http://www.w3.org/ns/tdmrep/}
1421 \_pdfmeta_xmp_property_new:nnnnn
1422 {tdm}
1423 {reservation}
1424 {Closed~Choice~of~Integer}
1425 {internal}
1426 {TDM~rights~are~reserved~(1)~or~not~reserved~(0)}
1427 \_pdfmeta_xmp_property_new:nnnnn
1428 {tdm}
1429 {policy}
1430 {URI}
1431 {internal}
1432 {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.)

```

1433 \cs_new_protected:Npn \_pdfmeta_xmp_schema_enable_pdfd:
1434 {
1435   \_pdfmeta_xmp_xmlns_new:nn {pdfd}{http://pdfa.org/declarations/}
1436   \_pdfmeta_xmp_schema_new:nnn
1437     {PDF~Declarations~Schema}
1438     {pdfd}
1439     {http://pdfa.org/declarations/}
1440   \_pdfmeta_xmp_property_new:nnnnn
1441     {pdfd}
1442     {declarations}

```

```

1443     {Bag-declaration}
1444     {external}
1445     {An-unordered-array-of-PDF-Declaration-entries,~where~each~PDF-Declaration-represen

```

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

```

1446     \cs_new_protected:cpn { __pdfmeta_xmp_schema_pdfd_additions: }
1447     {
1448         \__pdfmeta_xmp_add_packet_open:nn{pdfaSchema}{valueType}
1449         \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1450         \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1451         \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{claim}
1452         \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
1453             {http://pdfa.org/declarations/}
1454         \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{pdfd}
1455         \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
1456             {A-structure-describing-properties-of-an-individual claim.}
1457         \__pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
1458         \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1459         \__pdfmeta_xmp_add_packet_field:nnn{claimReport}{Text}
1460             {A~URL~to~a~report~containing~details~of~the~specific~conformance~claim}
1461         \__pdfmeta_xmp_add_packet_field:nnn{claimCredentials}{Text}
1462             {The~claimant's~credentials.}
1463         \__pdfmeta_xmp_add_packet_field:nnn{claimDate}{Text}
1464             {A~date~identifying~when~the~claim~was~made.}
1465         \__pdfmeta_xmp_add_packet_field:nnn{claimBy}{Text}
1466             {The~name~of~the~organization~and/or~individual~and/or~software~making}
1467         \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1468         \__pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
1469         \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1470         \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1471         \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{declaration}
1472         \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
1473             {http://pdfa.org/declarations/}
1474         \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{pdfd}
1475         \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
1476             {A-structure-describing-a~single~PDF~ Declaration~asserting~conformance~
1477             identified-standard-or~ profile.}
1478         \__pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
1479         \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1480         \__pdfmeta_xmp_add_packet_field:nnn{conformsTo}{Text}
1481             {A~property~containing~a~URI~specifying~the~standard~or~profile~by~the}
1482         \__pdfmeta_xmp_add_packet_field:nnn{claimData}{Bag-claim}
1483             {An-unordered-array-of~claim~data,~where~each~claim~identifies~the~natu}
1484         \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1485         \__pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
1486         \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1487         \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1488         \__pdfmeta_xmp_add_packet_close:nn{pdfaSchema}{valueType}
1489     }

```

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

```

1489     \cs_gset_eq:NN \__pdfmeta_xmp_schema_enable_pdfd: \prg_do_nothing:
1490 }

```

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      1491 \cs_new_protected:Npn \__pdfmeta_xmp_build_pdf:
    PDFversion              1492 {
```

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

```
1493   \__pdfmeta_xmp_add_packet_line_default:nnee
1494     {pdf}{Producer}
1495     { \c_sys_engine_exec_str- \c_sys_engine_version_str }
1496     { \pdfmanagement_get_documentproperty:n{hyperref/pdfproducer} }
```

Now the PDF version

```
1497   \__pdfmeta_xmp_add_packet_line:nne{pdf}{PDFVersion}{ \pdf_version: }
1498 }
```

(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    1499 \cs_new_protected:Npn \__pdfmeta_xmp_build_xmp:
    BaseUrl/baseurl         1500 {
```

The creator

```
1501   \__pdfmeta_xmp_add_packet_line_default:nnee
1502     {xmp}{CreatorTool}
1503     {LaTeX}
1504     { \pdfmanagement_get_documentproperty:n{hyperref/pdfcreator} }
```

The baseurl

```
1505   \__pdfmeta_xmp_add_packet_line_default:nnee
1506     {xmp}{BaseUrl}{ }
1507     { \pdfmanagement_get_documentproperty:n{hyperref/baseurl} }
```

CreationDate

```
1508   \__pdfmeta_xmp_date_get:nNN
1509     {document/creationdate} \l__pdfmeta_tmpa_tl \l__pdfmeta_tmpa_seq
1510   \__pdfmeta_xmp_add_packet_line:nne{xmp}{CreateDate}{ \__pdfmeta_xmp_print_date:N \l__pdfme
1511   \pdfmanagement_add:nne{Info}{CreateDate}{ ( \l__pdfmeta_tmpa_tl ) }
```

ModifyDate

```
1512   \__pdfmeta_xmp_date_get:nNN
1513     {document/moddate} \l__pdfmeta_tmpa_tl \l__pdfmeta_tmpa_seq
1514   \__pdfmeta_xmp_add_packet_line:nne{xmp}{ModifyDate}{ \__pdfmeta_xmp_print_date:N \l__pdfme
1515   \pdfmanagement_add:nne{Info}{ModDate}{ ( \l__pdfmeta_tmpa_tl ) }
```

MetadataDate

```
1516     \__pdfmeta_xmp_date_get:nNN
1517         {hyperref/pdfmetadate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1518     \__pdfmeta_xmp_add_packet_line:nne{xmp}{MetadataDate}{\__pdfmeta_xmp_print_date:N\l__pdf
1519 }
```

(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:

```
1520 \cs_new_protected:Npn \__pdfmeta_xmp_build_standards:
1521 {
1522     \__pdfmeta_xmp_add_packet_line:nne {pdfaid}{part}{\pdfmeta_standard_item:n{level}}
1523     \__pdfmeta_xmp_add_packet_line:nne
1524         {pdfaid}{conformance}{\pdfmeta_standard_item:n{conformance}}
1525     \int_compare:nNnTF {0\pdfmeta_standard_item:n{level}}<{4}
1526         {\__pdfmeta_xmp_add_packet_line:nne {pdfaid}{year} {\pdfmeta_standard_item:n{year}}}
1527         {\__pdfmeta_xmp_add_packet_line:nne {pdfaid}{rev} {\pdfmeta_standard_item:n{year}}}
1528     \__pdfmeta_xmp_add_packet_line:nne
1529         {pdfxid}{GTS_PDFXVersion}{\pdfmanagement_get_documentproperty:n{document/pdfstandard-
X}}
1530     \pdfmanagement_get_documentproperty:nNT {document/pdfstandard-UA}\l__pdfmeta_tmpa_tl
1531     {
1532         \__pdfmeta_xmp_add_packet_line:nne
1533             {pdfuaid}{part}{\exp_last_unbraced:No\use_i:nn \l__pdfmeta_tmpa_tl}
1534         \__pdfmeta_xmp_add_packet_line:nne
1535             {pdfuaid}{rev}{\exp_last_unbraced:No\use_ii:nn \l__pdfmeta_tmpa_tl}
1536     }
1537 }
```

(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.

```
1538 \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:

```
1539 \cs_new_protected:Npn \__pdfmeta_xmp_build_pdfd:
1540 {
1541     \prop_if_empty:NF\g__pdfmeta_xmp_pdfd_data_prop
1542     {
```

```

1543     \__pdfmeta_xmp_add_packet_open:nn{pdfd}{declarations}
1544     \__pdfmeta_xmp_add_packet_open:nn{rdf}{Bag}
1545     \prop_map_inline:Nn \g__pdfmeta_xmp_pdfd_data_prop
1546     {
1547         \__pdfmeta_xmp_build_pdfd_claim:nn{##1}{##2}
1548     }
1549     \__pdfmeta_xmp_add_packet_close:nn{rdf}{Bag}
1550     \__pdfmeta_xmp_add_packet_close:nn{pdfd}{declarations}
1551 }
1552 }

```

(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.

```

1553 \cs_new_protected:Npn \__pdfmeta_xmp_build_pdfd_claim:nn #1#2
1554 {
1555     \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1556     \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{conformsTo}{#1}
1557     \tl_if_empty:nF {#2}
1558     {
1559         \__pdfmeta_xmp_add_packet_open:nn{pdfd}{claimData}
1560         \__pdfmeta_xmp_add_packet_open:nn{rdf}{Bag}
1561         #2
1562         \__pdfmeta_xmp_add_packet_close:nn{rdf}{Bag}
1563         \__pdfmeta_xmp_add_packet_close:nn{pdfd}{claimData}
1564     }
1565     \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1566 }

```

(End of definition for __pdfmeta_xmp_build_pdfd_claim:nn.)

4.10 Photoshop

__pdfmeta_xmp_build_photoshop:

```

1567 \cs_new_protected:Npn \__pdfmeta_xmp_build_photoshop:
1568 {
1569     pdfauthortitle/photoshop:AuthorsPosition
1570     { \pdfmanagement_get_documentproperty:n{hyperref/pdfauthortitle} }
1571     pdfcaptionwriter/photoshop:CaptionWriter
1572     { \pdfmanagement_get_documentproperty:n{hyperref/pdfcaptionwriter} }
1573 }

```

(End of definition for __pdfmeta_xmp_build_photoshop:.)

4.11 XMP Media Management

_pdfmeta_xmp_build_xmpMM:

```
1574 \cs_new_protected:Npn \_pdfmeta_xmp_build_xmpMM:
1575 {
```

pdfdocumentid / xmpMM:DocumentID

```
1576 \str_set:N\l__pdfmeta_tmpa_str {\pdfmanagement_get_documentproperty:n{hyperref/pdfdocumentid}}
1577 \str_if_empty:NT \l__pdfmeta_tmpa_str
1578 {
1579 \_pdfmeta_xmp_create_uuid:nN
1580 {\jobname\_pdfmanagement_get_documentproperty:n{hyperref/pdfdocumentid}}
1581 \l__pdfmeta_tmpa_str
1582 }
1583 \_pdfmeta_xmp_add_packet_line:nnV{xmpMM}{DocumentID}
1584 \l__pdfmeta_tmpa_str
```

pdfinstanceid / xmpMM:InstanceID

```
1585 \str_set:N\l__pdfmeta_tmpa_str {\pdfmanagement_get_documentproperty:n{hyperref/pdfinstanceid}}
1586 \str_if_empty:NT \l__pdfmeta_tmpa_str
1587 {
1588 \_pdfmeta_xmp_create_uuid:nN
1589 {\jobname\_pdfmeta_xmp_currentdate_tl}
1590 \l__pdfmeta_tmpa_str
1591 }
1592 \_pdfmeta_xmp_add_packet_line:nnV{xmpMM}{InstanceID}
1593 \l__pdfmeta_tmpa_str
```

pdfversionid/xmpMM:VersionID

```
1594 \_pdfmeta_xmp_add_packet_line:nne{xmpMM}{VersionID}
1595 { \pdfmanagement_get_documentproperty:n{hyperref/pdfversionid} }
```

pdfrendition/xmpMM:RenditionClass

```
1596 \_pdfmeta_xmp_add_packet_line:nne{xmpMM}{RenditionClass}
1597 { \pdfmanagement_get_documentproperty:n{hyperref/pdfrendition} }

1598 }
```

(End of definition for _pdfmeta_xmp_build_xmpMM:.)

4.12 Rest of dublin Core data

_pdfmeta_xmp_build_dc:

```
dc:creator/pdfauthor 1599 \cs_new_protected:Npn \_pdfmeta_xmp_build_dc:
dc:subject/pdfkeywords 1600 {
```

dc:type/pdftype

pdfauthor/dc:creator

```
1601 \_pdfmeta_xmp_add_packet_list_simple:nne {dc}{creator}{Seq}
1602 { \pdfmanagement_get_documentproperty:n{hyperref/pdfauthor} }
1603 \int_compare:nNnT {0\pdfmeta_standard_item:n{level}}={1}
1604 { \pdfmanagement_remove:nn{Info}{Author} }
```

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

```
1605 \_pdfmeta_xmp_add_packet_list:nne {dc}{title}{Alt}
1606 { \pdfmanagement_get_documentproperty:n{hyperref/pdftitle} }
```

pdfkeywords/dc:subject

```
1607     \__pdfmeta_xmp_add_packet_list_simple:nnne {dc}{subject}{Bag}
1608     { \pdfmanagement_get_documentproperty:n{hyperref/pdfkeywords} }
1609     \int_compare:nNnT {0\pdfmeta_standard_item:n{level}}={1}
1610     { \pdfmanagement_remove:nn{Info}{Keywords} }
```

pdftype/dc:type

```
1611     \pdfmanagement_get_documentproperty:nNTF { hyperref/pdftype } \l__pdfmeta_tmpa_tl
1612     {
1613         \__pdfmeta_xmp_add_packet_list_simple:nnnV {dc}{type}{Bag}\l__pdfmeta_tmpa_tl
1614     }
1615     {
1616         \__pdfmeta_xmp_add_packet_list_simple:nnnn {dc}{type}{Bag}{Text}
1617     }
```

pdfpublisher/dc:publisher

```
1618     \__pdfmeta_xmp_add_packet_list_simple:nnne {dc}{publisher}{Bag}
1619     { \pdfmanagement_get_documentproperty:n{hyperref/pdfpublisher} }
```

pdfsubject/dc:description

```
1620     \__pdfmeta_xmp_add_packet_list:nnne
1621     {dc}{description}{Alt}
1622     { \pdfmanagement_get_documentproperty:n{hyperref/pdfsubject} }
```

lang/pdflang/dc:language

```
1623     \__pdfmeta_xmp_add_packet_list_simple:nnnV
1624     {dc}{language}{Bag}\l__pdfmeta_xmp_doclang_tl
```

pdfidentifier/dc:identifier

```
1625     \__pdfmeta_xmp_add_packet_line:nne{dc}{identifier}
1626     { \pdfmanagement_get_documentproperty:n{hyperref/pdfidentifier} }
```

pdfdate/dc:date

```
1627     \__pdfmeta_xmp_date_get:nNN {hyperref/pdfdate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1628     \__pdfmeta_xmp_add_packet_list_simple:nnne
1629     {dc}{date}{Seq}{\__pdfmeta_xmp_print_date:N\l__pdfmeta_tmpa_seq}
```

The file format

```
1630     \__pdfmeta_xmp_add_packet_line:nnn{dc}{format}{application/pdf}
```

The source

```
1631     \__pdfmeta_xmp_add_packet_line_default:nnee
1632     {dc}{source}
1633     { \c_sys_jobname_str.tex }
1634     { \pdfmanagement_get_documentproperty:n{hyperref/pdfsource} }

1635     \__pdfmeta_xmp_add_packet_list:nnne{dc}{rights}{Alt}
1636     { \pdfmanagement_get_documentproperty:n{hyperref/pdfcopyright} }

1637 }
```

(End of definition for __pdfmeta_xmp_build_dc: and others.)

4.13 xmpRights

`\_pdfmeta_xmp_build_xmpRights:`

```

1638 \cs_new_protected:Npn \_pdfmeta_xmp_build_xmpRights:
1639 {
1640   \_pdfmeta_xmp_add_packet_line:nne
1641     {xmpRights}
1642     {WebStatement}
1643     {\pdfmanagement_get_documentproperty:n{hyperref/pdflicenseurl}}
1644   \_pdfmeta_xmp_add_packet_line:nne
1645     {xmpRights}
1646     {Marked}
1647   {
1648     \str_case:en {\pdfmanagement_get_documentproperty:n{document/copyright}}
1649     {
1650       {true}{True}
1651       {false}{False}
1652     }
1653   }
1654 }

```

(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`

```

1655 \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`

```

1656 \cs_new_protected:Npn \_pdfmeta_xmp_build_iptc_data:N #1
1657 {
1658   \tl_clear:N #1
1659   \_pdfmeta_xmp_incr_indent:\_pdfmeta_xmp_incr_indent:\_pdfmeta_xmp_incr_indent:\_pdf
1660   \_pdfmeta_xmp_add_packet_line:nneN
1661     {Iptc4xmpCore}{CiAdrExtadr}
1662     {\pdfmanagement_get_documentproperty:n{hyperref/pdfcontactaddress}}
1663     #1
1664   \_pdfmeta_xmp_add_packet_line:nneN
1665     {Iptc4xmpCore}{CiAdrCity}
1666     {\pdfmanagement_get_documentproperty:n{hyperref/pdfcontactcity}}
1667     #1
1668   \_pdfmeta_xmp_add_packet_line:nneN
1669     {Iptc4xmpCore}{CiAdrPcode}
1670     {\pdfmanagement_get_documentproperty:n{hyperref/pdfcontactpostcode}}
1671     #1
1672   \_pdfmeta_xmp_add_packet_line:nneN
1673     {Iptc4xmpCore}{CiAdrCtry}
1674     {\pdfmanagement_get_documentproperty:n{hyperref/pdfcontactcountry}}
1675     #1
1676   \_pdfmeta_xmp_add_packet_line:nneN

```

```

1677     {Iptc4xmpCore}{CiTelWork}
1678     {\pdfmanagement_get_documentproperty:n{hyperref/pdfcontactphone}}
1679     #1
1680     \__pdfmeta_xmp_add_packet_line:nneN
1681     {Iptc4xmpCore}{CiEmailWork}
1682     {\pdfmanagement_get_documentproperty:n{hyperref/pdfcontactemail}}
1683     #1
1684     \__pdfmeta_xmp_add_packet_line:nneN
1685     {Iptc4xmpCore}{CiUrlWork}
1686     {\pdfmanagement_get_documentproperty:n{hyperref/pdfcontacturl}}
1687     #1
1688     \__pdfmeta_xmp_decr_indent:\__pdfmeta_xmp_decr_indent:\__pdfmeta_xmp_decr_indent:\__pdf
1689   }

```

(End of definition for __pdfmeta_xmp_build_iptc_data:N.)

__pdfmeta_xmp_build_iptc:

```

1690 \cs_new_protected:Npn \__pdfmeta_xmp_build_iptc:
1691 {
1692   \tl_if_empty:NF\l__pdfmeta_xmp_iptc_data_tl
1693   {
1694     \__pdfmeta_xmp_add_packet_open_attr:nnn
1695     {Iptc4xmpCore}{CreatorContactInfo}{rdf:parseType="Resource"}
1696     \tl_gput_right:Ne\g__pdfmeta_xmp_packet_tl { \l__pdfmeta_xmp_iptc_data_tl }
1697     \__pdfmeta_xmp_add_packet_close:nn
1698     {Iptc4xmpCore}{CreatorContactInfo}
1699   }
1700 }

```

(End of definition for __pdfmeta_xmp_build_iptc:.)

4.15 Prism

__pdfmeta_xmp_build_prism:
 complianceProfile
 prism:subtitle/pdfsubtitle

```

1701 \cs_new_protected:Npn \__pdfmeta_xmp_build_prism:
1702 {

```

The compliance profile is a fix value taken from hyperxmp

```

1703   \__pdfmeta_xmp_add_packet_line:nnn
1704   {prism}{complianceProfile}
1705   {three}

```

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

```

1706   \__pdfmeta_xmp_lang_get:eNN
1707   {\pdfmanagement_get_documentproperty:n{hyperref/pdfsubtitle}}
1708   \l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl
1709   \__pdfmeta_xmp_add_packet_line_attr:nneV
1710   {prism}{subtitle}
1711   {xml:lang="\l__pdfmeta_tmpa_tl"}
1712   \l__pdfmeta_tmpb_tl

```

Then publicationName

```

1713   \__pdfmeta_xmp_lang_get:eNN
1714   {\pdfmanagement_get_documentproperty:n{hyperref/pdfpublication}}
1715   \l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl

```

```

1716 \_pdfmeta_xmp_add_packet_line_attr:nneV
1717 {prism}{publicationName}
1718 {xml:lang="\l__pdfmeta_tmpa_tl"}
1719 \l__pdfmeta_tmpb_tl

```

Now the rest

```

1720 \_pdfmeta_xmp_add_packet_line:nne
1721 {prism}{bookEdition}
1722 {\pdfmanagement_get_documentproperty:n{hyperref/pdfbookedition}}
1723 \_pdfmeta_xmp_add_packet_line:nne
1724 {prism}{aggregationType}
1725 {\pdfmanagement_get_documentproperty:n{hyperref/pdfpubtype}}
1726 \_pdfmeta_xmp_add_packet_line:nne
1727 {prism}{volume}
1728 {\pdfmanagement_get_documentproperty:n{hyperref/pdfvolumenum}}
1729 \_pdfmeta_xmp_add_packet_line:nne
1730 {prism}{number}
1731 {\pdfmanagement_get_documentproperty:n{hyperref/pdfissuenum}}
1732 \_pdfmeta_xmp_add_packet_line:nne
1733 {prism}{pageRange}
1734 {\pdfmanagement_get_documentproperty:n{hyperref/pdfpagerange}}
1735 \_pdfmeta_xmp_add_packet_line:nne
1736 {prism}{issn}
1737 {\pdfmanagement_get_documentproperty:n{hyperref/pdfissn}}
1738 \_pdfmeta_xmp_add_packet_line:nne
1739 {prism}{eIssn}
1740 {\pdfmanagement_get_documentproperty:n{hyperref/pdfeissn}}
1741 \_pdfmeta_xmp_add_packet_line:nne
1742 {prism}{doi}
1743 {\pdfmanagement_get_documentproperty:n{hyperref/pdfdoi}}
1744 \_pdfmeta_xmp_add_packet_line:nne
1745 {prism}{url}
1746 {\pdfmanagement_get_documentproperty:n{hyperref/pdfurl}}

```

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

```

1747 \tl_set:Nx \l__pdfmeta_tmpa_tl { \pdfmanagement_get_documentproperty:n{hyperref/pdfnumpages}}
1748 \_pdfmeta_xmp_add_packet_line:nne
1749 {prism}{pageCount}
1750 {\tl_if_blank:VTF \l__pdfmeta_tmpa_tl {\PreviousTotalPages}{\l__pdfmeta_tmpa_tl}}
1751 }

```

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

4.16 TDM

_pdfmeta_xmp_build_tdm:

```

1752 \cs_new_protected:Npn \_pdfmeta_xmp_build_tdm:
1753 {

```

pdftdmreservation/tdm:reservation

```

1754 \_pdfmeta_xmp_add_packet_line:nne{tdm}{reservation}
1755 { \pdfmanagement_get_documentproperty:n{hyperref/pdftdmreservation} }

```

pdftdmpolicy/tdm:policy

```
1756 \__pdfmeta_xmp_add_packet_line:nne{tdm}{policy}  
1757 { \pdfmanagement_get_documentproperty:n{hyperref/pdftdmpolicy} }  
1758 }
```

(End of definition for __pdfmeta_xmp_build_tdm:.)

4.17 User additions

\g_pdfmeta_xmp_user_packet_str

```
1759 \tl_new:N \g__pdfmeta_xmp_user_packet_tl
```

(End of definition for \g__pdfmeta_xmp_user_packet_str.)

__pdfmeta_xmp_build_user:

```
1760 \cs_new_protected:Npn \__pdfmeta_xmp_build_user:  
1761 {  
1762 \int_zero:N \l__pdfmeta_xmp_indent_int  
1763 \g__pdfmeta_xmp_user_packet_tl  
1764 \int_set:Nn \l__pdfmeta_xmp_indent_int {3}  
1765 }
```

(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

```
1766 \hook_new:n { pdfmeta/xmp }  
1767 \AddToHook{shipout/lastpage}[pdfmanagement-testphase]  
1768 {  
1769 \bool_if:NT\g__pdfmeta_xmp_bool  
1770 {  
1771 \str_if_exist:NTF\c_sys_timestamp_str  
1772 {  
1773 \tl_set_eq:NN \l__pdfmeta_xmp_currentdate_tl \c_sys_timestamp_str  
1774 }  
1775 {  
1776 \file_get_timestamp:nN{\jobname.log}\l__pdfmeta_xmp_currentdate_tl  
1777 }  
1778 \__pdfmeta_xmp_date_split:VN\l__pdfmeta_xmp_currentdate_tl\l__pdfmeta_xmp_currentdate  
1779 \hook_use:n { pdfmeta/xmp }  
1780 \__pdfmeta_xmp_build_packet:  
1781 \pdf_object_new:n {__pdfmeta/xmp}  
1782 \exp_args:No  
1783 \__pdf_backend_metadata_stream:n {\g__pdfmeta_xmp_packet_tl}  
1784 \pdfmanagement_add:nne {Catalog} {Metadata}{\pdf_object_ref:n{__pdfmeta/xmp}}  
1785 \bool_if:NT \g__pdfmeta_xmp_export_bool  
1786 {  
1787 \iow_open:Nn\g_tmpa_iow{\g__pdfmeta_xmp_export_str.xmpi}  
1788 \exp_args:NNo\iow_now:Nn\g_tmpa_iow{\g__pdfmeta_xmp_packet_tl}  
1789 \iow_close:N\g_tmpa_iow  
1790 }  
1791 }  
1792 }
```

4.19 User commands

`\pdfmeta_xmp_add:n`

```
1793 \cs_new_protected:Npn \pdfmeta_xmp_add:n #1
1794 {
1795   \tl_gput_right:Nn \g__pdfmeta_xmp_user_packet_tl
1796   {
1797     \__pdfmeta_xmp_add_packet_chunk:n { #1 }
1798   }
1799 }
```

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

`\pdfmeta_xmp_xmlns_new:nn`

```
1800 \cs_new_protected:Npn \pdfmeta_xmp_xmlns_new:nn #1 #2
1801 {
1802   \prop_if_in:NnTF \g__pdfmeta_xmp_xmlns_prop {#1}
1803   {\msg_warning:nnnn{pdfmeta}{xmp-defined}{xmlns-namespace}{#1}}
1804   {\__pdfmeta_xmp_xmlns_new:nn {#1}{#2}}
1805 }
```

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

`\pdfmeta_xmp_schema_new:nnn`

```
1806 \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:nnnnn`

```
1807 \cs_set_eq:NN \pdfmeta_xmp_property_new:nnnnn \__pdfmeta_xmp_property_new:nnnnn
```

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

`\pdfmeta_xmp_add_declaration:n`

`\pdfmeta_xmp_add_declaration:e`

```
1808 \cs_new_protected:Npn \pdfmeta_xmp_add_declaration:n #1 %conformsTo uri
1809 {
1810   \__pdfmeta_xmp_schema_enable_pdfd:
1811   \prop_gput:Nnn\g__pdfmeta_xmp_pdfd_data_prop{#1}{ }
1812 }
1813 \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:ennnn`

```
1814 \cs_new_protected:Npn \pdfmeta_xmp_add_declaration:nnnnn #1#2#3#4#5
1815   % #1=conformsTo uri, #2 claimBy, #3 claimDate #4 claimCredentials #5 claimReport
1816   {
1817     \__pdfmeta_xmp_schema_enable_pdfd:
1818     \tl_set:Nn \l__pdfmeta_tmpa_tl
1819     {
1820       \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1821       \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimBy}{#2}
1822       \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimDate}{#3}
1823       \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimCredentials}{#4}
1824       \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimReport}{#5}
1825       \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
```

```

1826     }
1827     \prop_get:NnNT \g__pdfmeta_xmp_pdfd_data_prop {#1}\l__pdfmeta_tmpl_t1
1828     {
1829         \tl_concat:NNN \l__pdfmeta_tmpl_t1 \l__pdfmeta_tmpl_t1 \l__pdfmeta_tmpl_t1
1830     }
1831     \prop_gput:Nno\g__pdfmeta_xmp_pdfd_data_prop{#1}
1832     {
1833         \l__pdfmeta_tmpl_t1
1834     }
1835 }
1836 \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:
1837 \cs_new:Npn \__pdfmeta_xmp_iso_today:
1838 {
1839     \int_use:N\c_sys_year_int-
1840     \int_compare:nNnT {\c_sys_month_int} < {10}{0} \int_use:N\c_sys_month_int -
1841     \int_compare:nNnT {\c_sys_day_int} < {10}{0} \int_use:N\c_sys_day_int
1842 }
1843 \cs_new_protected:Npn \__pdfmeta_xmp_wtpdf_reuse_declaration:
1844 {
1845     \pdfmeta_xmp_add_declaration:eeenn
1846     {http://pdfa.org/declarations/wtpdf\c_hash_str reuse1.0}
1847     {LaTeX~Project}
1848     {\__pdfmeta_xmp_iso_today:}{}{}
1849 }
1850 \cs_new_protected:Npn \__pdfmeta_xmp_wtpdf_accessibility_declaration:
1851 {
1852     \pdfmeta_xmp_add_declaration:ennnn
1853     {http://pdfa.org/declarations/wtpdf\c_hash_str accessibility1.0}
1854     {LaTeX~Project}
1855     {\__pdfmeta_xmp_iso_today:}{}{}
1856 }

```

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

```

1857 </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	\'	856
\&	921	\+ 856

\-	856, 937	\cs_set_eq:NN	760, 761, 779, 785, 1085, 1806, 1807
\[	937		
\\	11, 12, 16		
\]	937		
A			
\A	937		
\AddToDocumentProperties	685, 694, 700, 706, 712, 718, 724, 730, 736, 743, 753		
\AddToHook	397, 419, 571, 744, 754, 756, 1767		
B			
BaseUrl/baseurl	1499		
bitset commands:			
\bitset_set_false:Nn	117, 118, 119		
\bitset_set_true:Nn	116		
\bitset_to_arabic:N	120, 121, 122, 123, 124		
bool commands:			
\bool_gset_false:N	766, 805		
\bool_gset_true:N	626, 765, 800, 809		
\bool_if:NTF	1769, 1785		
\bool_lazy_or:nnTF	423, 817		
\bool_new:N	625, 788		
C			
char commands:			
\char_generate:nn	822, 827, 828, 829		
clist commands:			
\clist_if_empty:nTF	1034, 1051		
\clist_map_inline:nn	555, 654, 1038, 1055, 1066		
complianceProfile	1701		
CreatorTool/pdfcreator	1499		
cs commands:			
\cs_generate_variant:Nn	632, 861, 933, 952, 961, 969, 975, 982, 997, 1007, 1017, 1030, 1047, 1079, 1813, 1836		
\cs_gset_eq:NN	1489		
\cs_if_exist:NTF	64		
\cs_if_exist_use:N	1184		
\cs_new:Npn	42, 821, 825, 833, 839, 862, 1837		
\cs_new_protected:Nn	633		
\cs_new_protected:Npn	29, 46, 81, 89, 96, 102, 108, 114, 533, 548, 623, 652, 845, 850, 857, 892, 905, 917, 938, 954, 962, 970, 976, 983, 988, 998, 1008, 1018, 1031, 1048, 1080, 1130, 1162, 1190, 1213, 1264, 1382, 1433, 1446, 1491, 1499, 1520, 1539, 1553, 1567, 1574, 1599, 1638, 1656, 1690, 1701, 1752, 1760, 1793, 1800, 1808, 1814, 1843, 1850		
D			
\d	856		
dc commands:			
dc:description/pdfsubject	1599		
dc:identifier/pdfidentifier	1599		
dc:language/lang/pdflang	1599		
dc:Ncreator/pdfauthor	1599		
dc:publisher/pdfpublisher	1599		
dc:subject/pdfkeywords	1599		
dc:type/pdftype	1599		
\DocumentMetadata	3, 4		
E			
exp commands:			
\exp_args:Nne	591, 596, 602		
\exp_args:Nne	413		
\exp_args:Nnne	66		
\exp_args:Nno	508, 1788		
\exp_args:No	1782		
\exp_args:Nv	609, 612		
\exp_last_unbraced:No	1533, 1535		
\exp_not:n	958, 966, 1135		
F			
file commands:			
\file_get_timestamp:nN	1776		
G			
group commands:			
\group_begin:	405, 550, 920		
\group_end:	414, 569, 929		
H			
hook commands:			
\hook_gput_code:nnn	126, 627		
\hook_new:n	1766		
\hook_use:n	1779		
I			
int commands:			
\int_compare:nNnTF	1525, 1603, 1609, 1840, 1841		
\int_decr:N	852		
\int_if_zero:nTF	439		
\int_if_zero_p:n	424, 425		
\int_incr:N	847		
\int_new:N	832		
\int_set:Nn	1123, 1764		
\int_use:N	1839, 1840, 1841		
\int_zero:N	1125, 1762		
iow commands:			
\iow_close:N	1789		

\iow_newline: ..	431, 447, 454, 835, 841	pdfdict commands:	
\iow_now:Nn	1788	\pdfdict_if_empty:nTF	403
\iow_open:Nn	1787	\pdfdict_new:n	514
\g_tmpa_iow	1787, 1788, 1789	\pdfdict_put:nnn	
iptc_L(schema)	1372		406, 407, 515, 551, 552, 563
J		\pdfdict_use:n	567
\jobname	1580, 1589, 1776	pdffile commands:	
K		\g_pdffile_embed_nonpdfa_int	425, 439
kernel internal commands:		\g_pdffile_embed_pdfa_int	424
\g__kernel_pdfmanagement_end_-		\pdffile_embed_stream:nnN	408
run_code_tl	401	pdfmanagement commands:	
keys commands:		\pdfmanagement_add:nnn	
\keys_define:nn	463, 470, 677, 773, 791		568, 629, 630, 1511, 1515, 1784
\l_keys_key_str	510	\pdfmanagement_get_documentproperty:n	
\keys_set:nn	467, 770, 795		895, 1082, 1083, 1496, 1504, 1507, 1529, 1570, 1572, 1576, 1580, 1585, 1595, 1597, 1602, 1606, 1608, 1619, 1622, 1626, 1634, 1636, 1643, 1648, 1662, 1666, 1670, 1674, 1678, 1682, 1686, 1707, 1714, 1722, 1725, 1728, 1731, 1734, 1737, 1740, 1743, 1746, 1747, 1755, 1757
M		\pdfmanagement_get_documentproperty:nNTF	
msg commands:			1530, 1611
\msg_error:nnn	689	\pdfmanagement_remove:nn	1604, 1610
\msg_new:nnn . 7, 9, 10, 14, 814, 815, 816		pdfmeta commands:	
\msg_note:nnn	445	\pdfmeta_set_regression_data:	6, 623
\msg_warning:nnn	39, 429, 452, 586, 619	\pdfmeta_standard_family:nn	
\msg_warning:nnnn ..	1167, 1210, 1803		2, 29, 29, 656, 742, 752
\msg_warning:nnnnn	660, 668	\pdfmeta_standard_get:nN	3, 46, 46
P		\pdfmeta_standard_item:n	
pdf commands:			2, 42, 42, 594, 599, 605, 643, 649, 663, 671, 1522, 1524, 1525, 1526, 1527, 1603, 1609
\pdf_object_if_exist:nTF	535	\pdfmeta_standard_verify:n	2, 50
\pdf_object_new:n	537, 1781	\pdfmeta_standard_verify:nn	3, 60
\pdf_object_ref:n	554, 1784	\pdfmeta_standard_verify:nnN	3
\pdf_object_ref_last:	568	\pdfmeta_standard_verify:nnTF	
\pdf_object_unnamed_write:nn	567		2, 60, 635, 646, 658, 666
\pdf_object_write:nnn	538	\pdfmeta_standard_verify:nTF	
\pdf_string_from_unicode:nnN	562		2, 50, 128, 130, 132, 134, 399, 421, 437, 573, 638
\pdf_version:		\pdfmeta_standard_verify_p:n	50
.....	4, 636, 647, 659, 661, 667, 669, 1497	\pdfmeta_xmp_add:n	10, 1793, 1793
\pdf_version_compare:NnTF	83, 91	\pdfmeta_xmp_add_declaration:n	
\pdf_version_gset:n	632, 640, 643, 649		10, 1808, 1808, 1813
pdf internal commands:		\pdfmeta_xmp_add_declaration:nnnnn	
__pdf_backend_metadata_stream:n			10, 1814, 1814, 1836, 1845, 1852
.....	1783	\pdfmeta_xmp_property_new:nnnnn	
__pdf_backend_Names_gpush:nn	413		11, 1807, 1807
__pdf_backend_omit_charset:n	133	\pdfmeta_xmp_schema_new:nnn	
__pdf_backend_omit_cidset:n	135		11, 1806, 1806
__pdf_backend_omit_info:n	131	\pdfmeta_xmp_xmlns_new:nn	
__pdf_backend_set_regression_-			10, 1800, 1800
data:	624		
pdfaid~(schema)	1232		
pdfannot commands:			
\pdfannot_dict_put:nnn			
.....	120, 121, 122, 123, 124		
\l_pdfannot_F_bitset	116,		
117, 118, 119, 120, 121, 122, 123, 124			

pdfmeta internal commands:

__pdfmeta_check_standard_-
 pdfversion: 136, 652
 __pdfmeta_embed_colorprofile:n
 533, 533, 579, 609
 __pdfmeta_force_standard_-
 pdfversion: 633, 684, 742, 752
 \g__pdfmeta_outputintents_prop .
 462, 476, 484,
 492, 500, 509, 575, 593, 598, 604, 610
 \g__pdfmeta_standard_A_prop
 25, 33, 36, 682, 683
 \g__pdfmeta_standard_pdf/A-1B_-
 prop 138
 \g__pdfmeta_standard_pdf/A-2A_-
 prop 138
 \g__pdfmeta_standard_pdf/A-2B_-
 prop 138
 \g__pdfmeta_standard_pdf/A-2U_-
 prop 138
 \g__pdfmeta_standard_pdf/A-3A_-
 prop 138
 \g__pdfmeta_standard_pdf/A-3B_-
 prop 138
 \g__pdfmeta_standard_pdf/A-3U_-
 prop 138
 \g__pdfmeta_standard_pdf/A-4_-
 prop 138
 \g__pdfmeta_standard_pdf/A-4E_-
 prop 138
 \g__pdfmeta_standard_pdf/A-4F_-
 prop 138
 \g__pdfmeta_standard_pdf/UA-1_-
 prop 336
 \g__pdfmeta_standard_pdf/UA-2_-
 prop 336
 \g__pdfmeta_standard_pdf/X-4_-
 prop 356
 \g__pdfmeta_standard_pdf/X-4P_-
 prop 356
 \g__pdfmeta_standard_pdf/X-5G_-
 prop 356
 \g__pdfmeta_standard_pdf/X-5N_-
 prop 356
 \g__pdfmeta_standard_pdf/X-5PG_-
 prop 356
 \g__pdfmeta_standard_pdf/X-6_-
 prop 356
 \g__pdfmeta_standard_pdf/X-6N_-
 prop 356
 \g__pdfmeta_standard_pdf/X-6P_-
 prop 356

\g__pdfmeta_standard_prop
 12, 25, 33, 34, 36, 44,
 48, 52, 62, 70, 427, 441, 683, 746, 755
 \g__pdfmeta_standard_UA_prop ...
 25, 741, 751
 __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,
 693, 699, 705, 711, 717, 723, 729, 735
 \l__pdfmeta_tmpa_seq
 19, 941, 942, 948, 949, 1509, 1510,
 1513, 1514, 1517, 1518, 1627, 1629
 \g__pdfmeta_tmpa_str
 22, 924, 925, 926, 927, 928, 930
 \l__pdfmeta_tmpa_str
 19, 562, 564, 993, 994,
 1003, 1004, 1013, 1014, 1576, 1577,
 1581, 1584, 1585, 1586, 1590, 1593
 \l__pdfmeta_tmpa_tl
 19, 412, 413, 427, 432,
 441, 443, 455, 560, 562, 923, 924,
 1023, 1026, 1028, 1057, 1058, 1063,
 1068, 1069, 1072, 1195, 1509, 1511,
 1513, 1515, 1517, 1530, 1533, 1535,
 1611, 1613, 1627, 1708, 1711, 1715,
 1718, 1747, 1750, 1818, 1829, 1833
 \l__pdfmeta_tmpb_seq 19
 \l__pdfmeta_tmpb_tl
 19, 606, 607, 609, 614,
 619, 1057, 1061, 1063, 1068, 1072,
 1708, 1712, 1715, 1719, 1827, 1829
 __pdfmeta_verify_pdfa_annot_-
 flags: 114, 129
 __pdfmeta_write_outputintent:nn
 533, 548, 581, 613
 __pdfmeta_xmp_add_packet_-
 chunk:n . 954, 954, 961, 972, 979,
 986, 994, 1014, 1091, 1124, 1126, 1797
 __pdfmeta_xmp_add_packet_-
 chunk:nN 962, 962, 969, 1004
 __pdfmeta_xmp_add_packet_-
 close:nn 983,
 983, 1043, 1044, 1075, 1076, 1104,
 1105, 1120, 1121, 1122, 1182, 1183,
 1185, 1205, 1220, 1411, 1412, 1413,

1414, 1415, 1467, 1468, 1469, 1483,
 1484, 1485, 1486, 1487, 1549,
 1550, 1562, 1563, 1565, 1697, 1825
 _pdfmeta_xmp_add_packet_-
 field:nnn 1213, 1213, 1395, 1397,
 1399, 1401, 1403, 1405, 1407, 1409,
 1459, 1461, 1463, 1465, 1479, 1481
 _pdfmeta_xmp_add_packet_-
 line:nnn 988,
 988, 997, 1028, 1040, 1176, 1177,
 1178, 1201, 1202, 1203, 1204, 1217,
 1218, 1219, 1387, 1388, 1390, 1391,
 1451, 1452, 1454, 1455, 1471, 1472,
 1474, 1475, 1497, 1510, 1514, 1518,
 1522, 1523, 1526, 1527, 1528, 1532,
 1534, 1556, 1569, 1571, 1583, 1592,
 1594, 1596, 1625, 1630, 1640, 1644,
 1703, 1720, 1723, 1726, 1729, 1732,
 1735, 1738, 1741, 1744, 1748,
 1754, 1756, 1821, 1822, 1823, 1824
 _pdfmeta_xmp_add_packet_-
 line:nnnN . 998, 998, 1007, 1660,
 1664, 1668, 1672, 1676, 1680, 1684
 _pdfmeta_xmp_add_packet_line_-
 attr:nnnn 1008, 1008,
 1017, 1060, 1062, 1071, 1709, 1716
 _pdfmeta_xmp_add_packet_line_-
 default:nnnn 1018,
 1018, 1030, 1493, 1501, 1505, 1631
 _pdfmeta_xmp_add_packet_-
 list:nnnn
 1048, 1079, 1605, 1620, 1635
 _pdfmeta_xmp_add_packet_list_-
 simple:nnnn . 1031, 1047, 1601,
 1607, 1613, 1616, 1618, 1623, 1628
 _pdfmeta_xmp_add_packet_-
 open:nn 970, 970, 975, 1036,
 1037, 1053, 1054, 1093, 1094, 1098,
 1099, 1179, 1180, 1200, 1384, 1385,
 1393, 1394, 1448, 1449, 1457, 1458,
 1477, 1478, 1543, 1544, 1559, 1560
 _pdfmeta_xmp_add_packet_open_-
 attr:nnn
 .. 976, 976, 982, 1096, 1175, 1216,
 1386, 1450, 1470, 1555, 1694, 1820
 _pdfmeta_xmp_add_pdfxid: . 695,
 701, 707, 713, 719, 725, 731, 737, 1264
 \g_pdfmeta_xmp_bool
 625, 765, 766, 1769
 _pdfmeta_xmp_build_dc:
 1111, 1599, 1599
 _pdfmeta_xmp_build_iptc:
 1116, 1690, 1690
 _pdfmeta_xmp_build_iptc_data:N
 1086, 1656, 1656
 _pdfmeta_xmp_build_packet: ...
 1080, 1080, 1780
 _pdfmeta_xmp_build_pdf:
 1107, 1491, 1491
 _pdfmeta_xmp_build_pdfd:
 1110, 1539, 1539
 _pdfmeta_xmp_build_pdfd_-
 claim:nn 1547, 1553, 1553
 _pdfmeta_xmp_build_photoshop:
 1112, 1567, 1567
 _pdfmeta_xmp_build_prism:
 1115, 1701, 1701
 _pdfmeta_xmp_build_standards:
 1109, 1520, 1520
 _pdfmeta_xmp_build_tdm:
 1117, 1752, 1752
 _pdfmeta_xmp_build_user:
 1118, 1760, 1760
 _pdfmeta_xmp_build_xmp:
 1113, 1499, 1499
 _pdfmeta_xmp_build_xmpMM:
 1114, 1574, 1574
 _pdfmeta_xmp_build_xmpRights:
 1108, 1638, 1638
 _pdfmeta_xmp_create_uuid:nN ..
 905, 905, 1579, 1588
 \l_pdfmeta_xmp_currentdate_seq
 890, 898, 1778
 \l_pdfmeta_xmp_currentdate_tl .
 890, 899, 1589, 1773, 1776, 1778
 _pdfmeta_xmp_date_get:nNN
 892, 892, 1508, 1512, 1516, 1627
 \l_pdfmeta_xmp_date_regex 854, 859
 _pdfmeta_xmp_date_split:nN ...
 857, 857, 861, 902, 1778
 _pdfmeta_xmp_decr_indent:
 833, 850, 985, 1688
 \l_pdfmeta_xmp_doclang_tl
 934, 1082, 1085, 1624
 \g_pdfmeta_xmp_export_bool
 788, 800, 805, 809, 1785
 \g_pdfmeta_xmp_export_str
 789, 801, 810, 1787
 _pdfmeta_xmp_generate_bom: ...
 817, 821, 825, 1092
 _pdfmeta_xmp_incr_indent:
 833, 845, 973, 980, 1659
 _pdfmeta_xmp_indent:
 833, 833, 958, 966
 _pdfmeta_xmp_indent:n 833, 839, 1135
 \l_pdfmeta_xmp_indent_int . 832,
 836, 847, 852, 1123, 1125, 1762, 1764

_pdfmeta_xmp_iptc_data_tl . . .	PDFVersion	1491
. 1086, 1087, 1655, 1692, 1696	pdfxid~(schema)	1264
_pdfmeta_xmp_iso_today:	photoshop commands:	
. 1837, 1848, 1855	photoshop:AuthorsPosition/pdfauthortitle	
_pdfmeta_xmp_lang_get:nNN		1599
. 938, 952, 1057, 1068, 1706, 1713	photoshop:CaptionWriter/pdfcaptionwriter	
_pdfmeta_xmp_lang_regex 936, 941		1599
_pdfmeta_xmp_metalang_tl	\PreviousTotalPages	1750
934, 944, 1058, 1069, 1083, 1084, 1085	prg commands:	
_g_pdfmeta_xmp_packet_tl	_prg_do_nothing:	
. 953, 956, 1696, 1783, 1788		760, 761, 779, 785, 1489
_g_pdfmeta_xmp_pdfd_data_prop	_prg_new_conditional:Npnn	50
. 1538, 1541, 1545, 1811, 1827, 1831	_prg_new_protected_conditional:Npnn	
_pdfmeta_xmp_print_date:N		60
. 862, 862, 1510, 1514, 1518, 1629	_prg_replicate:nn	836, 842, 1124
_pdfmeta_xmp_property_new:nnnnn	_prg_return_false:	
. 1189, 1190, 1226, 1236,		54, 73, 85, 93, 100, 106, 112
1242, 1252, 1258, 1271, 1282, 1288,	_prg_return_true:	
1294, 1300, 1306, 1312, 1318, 1324,		57, 77, 86, 94, 99, 105, 111
1330, 1336, 1342, 1348, 1354, 1360,	prism commands:	
1366, 1376, 1421, 1427, 1440, 1807	prism:subtitle/pdfsubtitle	1701
_pdfmeta_xmp_sanitize:nN	prism~(schema)	1278
. 917, 917, 933, 993, 1003, 1013	Producer/pdfproducer	1491
_pdfmeta_xmp_schema_enable_-	prop commands:	
pdfd: 1433, 1489, 1810, 1817	_prop_const_from_keyval:Nn	517, 524
_pdfmeta_xmp_schema_new:nnn	_prop_get:NnN	48, 603
. 1162, 1162, 1222, 1232, 1248,	_prop_get:NnNTF	
1267, 1278, 1372, 1417, 1436, 1806		427, 441, 557, 1195, 1827
_g_pdfmeta_xmp_schema_property_-	_prop_gput:Nnn	204,
prop 1189, 1195, 1197	206, 208, 214, 218, 220, 232, 234,	
_l_pdfmeta_xmp_schema_seq	236, 244, 246, 248, 257, 259, 261,	
. 1089, 1100, 1161, 1170	272, 274, 276, 284, 286, 288, 296,	
_g_pdfmeta_xmp_user_packet_str 1759	298, 300, 302, 304, 306, 308, 321,	
_g_pdfmeta_xmp_user_packet_tl	324, 334, 476, 484, 492, 500, 509,	
. 1759, 1763, 1795	597, 746, 755, 1132, 1197, 1811, 1831	
_pdfmeta_xmp_wtpdf_accessibility_-	_prop_gremove:Nn	
declaration:		211, 223, 264, 310, 312, 314, 327
. 758, 760, 782, 785, 1837, 1850	_prop_gset_eq:NN	33,
_pdfmeta_xmp_wtpdf_reuse_-	34, 36, 201, 229, 241, 254, 269, 281,	
declaration:	293, 318, 331, 682, 683, 693, 699,	
. 759, 761, 776, 779, 1837, 1843	705, 711, 717, 723, 729, 735, 741, 751	
_pdfmeta_xmp_xmlns_new:nn	_prop_gset_from_keyval:Nn	
. 1130, 1130, 1138,		139, 338, 348,
1139, 1140, 1141, 1142, 1143, 1144,	365, 369, 373, 377, 381, 385, 389, 393	
1146, 1147, 1148, 1149, 1150, 1151,	_prop_if_empty:NTF	1541
1152, 1153, 1154, 1155, 1156, 1157,	_prop_if_exist:NTF	31, 577, 607
1158, 1159, 1160, 1266, 1435, 1804	_prop_if_in:NnNTF	52, 62, 592, 1802
_g_pdfmeta_xmp_xmlns_prop	_prop_item:Nn	44, 70, 541
. 1128, 1132, 1802	_prop_map_inline:Nn	575, 610, 1545
_g_pdfmeta_xmp_xmlns_tl	_prop_new:N	
. 1097, 1128, 1133		25, 26, 27, 28, 138, 200, 228,
pdfmetatmpa internal commands:	240, 253, 268, 280, 292, 317, 330,	
_g_pdfmetatmpa_str	336, 337, 356, 357, 358, 359, 360,	
pdfuaid~(schema)	361, 362, 363, 462, 1129, 1189, 1538	
1248		

<code>\ProvidesExplPackage</code>	3	<code>\sys_if_engine_xetex_p:</code>	819
R		<code>\c_sys_jobname_str</code>	801, 1633
regex commands:		<code>\c_sys_month_int</code>	1840
<code>\regex_extract_once:NnN</code>	941	<code>\c_sys_timestamp_str</code>	6, 1771, 1773
<code>\regex_new:N</code>	854, 936	<code>\c_sys_year_int</code>	1839
<code>\regex_set:Nn</code>	855, 937	T	
<code>\regex_split:NnN</code>	859	<code>tdmrep_□(schema)</code>	1417
S		tex commands:	
seq commands:		<code>\tex_mdffivesum:D</code>	907
<code>\seq_if_empty:NTF</code>	942	text commands:	
<code>\seq_item:Nn</code>	864, 866, 868, 870, 872, 873, 875, 876, 878, 879, 880, 881, 883, 884, 887, 948, 949	<code>\text_declare_purify_equivalent:Nn</code>	921, 922
<code>\seq_map_inline:Nn</code>	1100	<code>\text_purify:n</code>	923
<code>\seq_new:N</code>	23, 24, 891, 1161	<code>\texttilde</code>	922
<code>\seq_put_right:Nn</code>	1170	tl commands:	
<code>\seq_remove_all:Nn</code>	1089	<code>\c_space_tl</code>	540, 836, 842
<code>\seq_set_eq:NN</code>	898	<code>\tl_clear:N</code>	1658
str commands:		<code>\tl_concat:NNN</code>	1829
<code>\c_hash_str</code>	10, 1095, 1145, 1153, 1156, 1157, 1158, 1159, 1846, 1853	<code>\tl_gput_right:Nn</code>	401, 956, 1133, 1173, 1198, 1696, 1795
<code>\str_case:nn</code>	1648	<code>\tl_if_blank:nTF</code>	474, 482, 490, 498, 506, 864, 871, 874, 877, 882, 896, 991, 1001, 1011, 1021, 1084, 1750
<code>\str_convert_pdfname:n</code>	551	<code>\tl_if_empty:nTF</code>	1087, 1692
<code>\str_greplace_all:Nnn</code>	925, 926, 927, 928	<code>\tl_if_empty:nTF</code>	1557
<code>\str_gset:Nn</code>	810, 924	<code>\tl_if_eq:nnTF</code>	110, 1058, 1069
<code>\str_gset_eq:NN</code>	801	<code>\tl_if_exist:nTF</code>	1165, 1193
<code>\str_if_empty:NTF</code>	1577, 1586	<code>\tl_if_in:nnTF</code>	98, 104
<code>\str_if_eq:nnTF</code>	443	<code>\tl_new:N</code>	19, 20, 890, 934, 935, 953, 1171, 1172, 1655, 1759
<code>\str_if_exist:NTF</code>	1771	<code>\tl_put_right:Nn</code>	964
<code>\str_lowercase:n</code>	907	<code>\tl_set:Nn</code>	895, 923, 944, 945, 948, 949, 1023, 1026, 1082, 1083, 1747, 1818
<code>\str_new:N</code>	21, 22, 789, 1128	<code>\tl_set_eq:NN</code>	899, 1773
<code>\str_range:Nnn</code>	910, 911, 912, 913, 914	<code>\tl_to_str:N</code>	921, 924
<code>\str_set:Nn</code>	907, 908, 1576, 1585	<code>\tl_use:N</code>	1102, 1181
<code>\str_set_eq:NN</code>	930	U	
<code>\c_tilde_str</code>	922	use commands:	
sys commands:		<code>\use:N</code>	67
<code>\c_sys_day_int</code>	1841	<code>\use_i:nn</code>	1533
<code>\c_sys_engine_exec_str</code>	629, 1495	<code>\use_ii:nn</code>	1535
<code>\c_sys_engine_version_str</code>	629, 1495		
<code>\sys_if_engine luatex_p:</code>	818		