The l3pdfmeta module
PDF standards
LaTeX PDF management bundle

The LaTeX Project*

Version 0.96s, released 2025-06-23

1 l3pdfmeta documentation

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

1.1 Verifying requirements of PDF standards

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

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

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

Verifying against a PDF-standard involves two different task:

- Check if you are allowed to ignore the requirement.

- Decide which action to take if the answer to the first question is NO.

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

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

---

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

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

`\pdfmeta_standard_verify_p:n` ⋆ `\pdfmeta_standard_verify:n{⟨requirement⟩}`
`\pdfmeta_standard_verify:n`*TF* ⋆

> 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:nn`*TF* `\pdfmeta_standard_verify:nn{⟨requirement⟩}{⟨value⟩}`

> This checks if ⟨`requirement`⟩ is listed in the standard, if yes it tries to find a pre-defined test handler for the requirement and passes ⟨`value`⟩ and the value recorded in the standard to it. The handler returns `FALSE` if some special action is needed (e.g. if ⟨`value`⟩ violates the rule) and `TRUE` if no special action is needed. If no handler exists this commands works like `\pdfmeta_standard_verify:n`.
>
> In some cases one needs to query the value in the standard, e.g. to correct a wrong minimal PDF version you need to know which version is required by `min_pdf_version`. For this two commands to access the value are provided:

`\pdfmeta_standard_item:n` ⋆ `\pdfmeta_standard_item:n{⟨requirement⟩}`

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

`\pdfmeta_standard_get:nN` `\pdfmeta_standard_get:nN{⟨requirement⟩}` ⟨`tl var`⟩

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

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

1.1.1 Simple tests without handler

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

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

`no_encryption` don't encrypt

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

no_embed_content no `/EF` key in filespec, no `/Type/EmbeddedFiles`. *This will be checked in future by l3pdffiles 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.

1.1.2 Tests with values and special handlers

`min_pdf_version` stores the minimal PDF version needed for a standard. It should be checked against the current PDF version (`\pdf_version:`). A failure means that the version should be changed. Currently there is only one hard requirement which leads to a failure in a validator like verapdf: The A-4 standard should use PDF 2.0. As PDF A-1 is based on PDF 1.4 and PDF A-2 and A-3 are based on PDF 1.7 l3pdfmeta also sets these versions also as requirements. These requirements are checked by l3pdfmeta when the version is set with `\DocumentMetadata` and a warning is issued (but the version is not changed). More checks are only needed if the version is changed later.

`max_pdf_version` stores the maximal PDF version. It should be checked against the current PDF version (`\pdf_version:`). A failure means that the version should be changed. The check is currently relevant only for the A-1 to A-3 standards: PDF 2.0 leads to a failure in a validator like verapdf so the maximal version should be PDF 1.7. This requirement is checked by l3pdfmeta when the version is set with `\DocumentMetadata` and a warning is issued (but the version is not changed). More checks are only needed if the version is changed later.

`named_actions` this requirement restricts the list of allowed named actions to `NextPage`, `PrevPage`, `FirstPage`, `LastPage`. The check should supply the named action without slash (e.g. `View` (failure) or `NextPage` (pass)).

`annot_action_A` (rule 6.6.1-1) this requirement restricts the allowed subtypes of the `/A` dictionary of an action. The check should supply the user subtype without slash e.g. as `GoTo` (pass) or `Movie` (failure).

1.2 Colorprofiles and OutputIntent

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

The problem is that the pdf/A standards also require, that if the PDF has more then one entry in the `/OutputIntent` array (which is allowed), their `/DestOutputProfile` should all reference the same color profile[2].

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

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

The interface looks like this

```
\DocumentMetadata
  {
    %other options for example pdfstandard
    colorprofiles=
      {
        A = sRGB.icc, %or a or longer GTS_PDFA1 = sRGB.icc
```

---

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

```
          X = FOGRA39L_coated.icc, % or x or longer GTS_PDFX
          ISO_PDFE1 = whatever.icc
       }

    }
```

`sRGB.icc` and `FOGRA39L_coated.icc` (from the **colorprofiles** package are predefined and will work directly[3]. `whatever.icc` will need special setup in the document preamble to declare the values for the `OutputIntent` dictionary, but the interface hasn't be added yet. This will be decided later.

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

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

    }
```

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

1.3 Regression tests

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

`\pdfmeta_set_regression_data:` `\pdfmeta_set_regression_data:`

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

---

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

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

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

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

hyperxmp uses `\hypersetup` as user interface to enter the XMP-metadata. This syntax is also supported by the new code[5], so if hyperref has been loaded, e.g. `pdftitle=xxx` can be used to set the title. But XMP-metadata shouldn't require to use hyperref and in a future version an interface without hyperref will be added.

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

2.1 Debug option

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

```
\DocumentMetadata{debug={xmp-export}}
%or
\DocumentMetadata{debug={xmp-export=true}}
%or
\DocumentMetadata{debug={xmp-export=filename}}
```

By default the data are written to `\jobname.xmpi`, if a `filename` is given, then `filename.xmpi` is used instead. `xmp-export=false` deactivates the export.

2.2 Encoding and escaping

XMP-metadata are stored as UTF-8 in the PDF. This mean if you open a PDF in an editor a content like "grüße" will be shown probably as "grÃ¼ÃŸe". As XMP-metadata are in XML format special chars like <, >, and & and „ must be escaped.

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

User data should normally be declared in the preamble (or even in the `\DocumentMetadata` command), and consist of rather simple text; `&` can be entered as `\&` (but directly `&` will normally work too), babel shorthands should not be used. Some data are interpreted as comma lists, in this cases commas which are part of the text should be protected by braces. In some cases a text in brackets like `[en]` is interpreted as language tag, if they are part of a text they should be protected by braces too. XMP-metadata are

---

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

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

stored uncompressed in the PDF so if you are unsure if a value has been passed correctly, open the PDF in an editor, copy the whole block and pass it to a validator, e.g. https://www.w3.org/RDF/Validator/.

2.3 User interfaces and differences to **hyperxmp**

2.3.1 PDF standards

The hyperxmp/hyperref keys `pdfapart`, `pdfaconformance`, `pdfuapart`, `pdfxstandard` and `pdfa` are ignored by this code. Standards must be set with the `pdfstandard` key of `\DocumentMetadata`. This key can be used more than once, e.g.
`pdfstandard=A-2b,pdfstandard=X-4,pdfstandard=UA-1`.
Note that using these keys doesn't mean that the document actually follows the standard. LaTeX can neither ensure nor check all requirements of a standard, and not everything it can do theoretically has already been implemented. When setting an `A` standard, the code will e.g. insert a color profile and warn if the PDF version doesn't fit, but `X` and `UA` currently only adds the relevant declarations to the XMP-metadata. It is up to the author to ensure and validate that the document actually follows the standard.

2.3.2 Declarations

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

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

2.3.3 Dates

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

can be set with `\hypersetup` with the keys `pdfcreationdate`, `pdfmoddate` and `pdfmetadate`.

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

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

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

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

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

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

2.4 Language

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

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

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

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

2.5 Rights

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

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

---

[6]Extracting 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 `\DocumentMetadata` key xmp.

`\pdfmeta_xmp_add:n`   `\pdfmeta_xmp_add:n{⟨XML⟩}`

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

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

With this command a xmlns name space can be added. 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).

`\pdfmeta_xmp_add_declaration:n` `\pdfmeta_xmp_add_declaration:n{⟨uri⟩}`
`\pdfmeta_xmp_add_declaration:e`

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

`\pdfmeta_xmp_add_declaration:nnnnn`   `\pdfmeta_xmp_add_`
`\pdfmeta_xmp_add_declaration:(ennnn|eeenn)` `declaration:nnnnn{⟨uri⟩}{⟨By⟩}{⟨Date⟩}{⟨Credentials⟩}{⟨Report⟩}`

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

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/Factur X for invoices. A schema declaration should be added only once but as this task is probably not needed frequently only light guards are there to avoid duplicated entries.

`\pdfmeta_xmp_schema_new:nnn`	`\pdfmeta_xmp_schema_new:nnn{⟨text⟩}{⟨prefix⟩}{⟨uri⟩}`

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

`\pdfmeta_xmp_property_new:nnnnn`	`\pdfmeta_xmp_property_new:nnnnn{⟨schema prefix⟩}{⟨name⟩}{⟨type⟩}{⟨category⟩}{⟨description⟩}`

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

3 **l3pdfmeta** implementation

```
1 ⟨@@=pdfmeta⟩
2 ⟨*header⟩
3 \ProvidesExplPackage{l3pdfmeta}{2025-06-23}{0.96s}
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 not fitting pdf version
```
8 \msg_new:nnn  {pdf }{wrong-pdfversion}
9   {PDF~version~#1~is~too~#2~for~standard~'#3'.}
```

Messages for embedded files
```
10 \msg_new:nnn  {pdf }{validation-failure}
11   {
12     PDF~standard~validation~failure.\\
13     #1
14   }
```

`\l__pdfmeta_tmpa_tl`	
`\l__pdfmeta_tmpb_tl`	`15 \tl_new:N  \l__pdfmeta_tmpa_tl`
`\l__pdfmeta_tmpa_str`	`16 \tl_new:N  \l__pdfmeta_tmpb_tl`
`\g__pdfmetatmpa_str`	`17 \str_new:N \l__pdfmeta_tmpa_str`
`\l__pdfmeta_tmpa_seq`	`18 \str_new:N \g__pdfmeta_tmpa_str`
`\l__pdfmeta_tmpb_seq`	`19 \seq_new:N \l__pdfmeta_tmpa_seq`
	`20 \seq_new:N \l__pdfmeta_tmpb_seq`

(*End of definition for* `\l__pdfmeta_tmpa_tl` *and others.*)

3.1 **Standards (work in progress)**

3.1.1 **Tools and tests**

This internal property will contain for now the settings for the document.

`\g__pdfmeta_standard_prop`

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

(*End of definition for* `\g__pdfmeta_standard_prop`.)

3.1.2 Functions to check a requirement

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

`\pdfmeta_standard_item:n`

```
22 \cs_new:Npn \pdfmeta_standard_item:n #1
23   {
24     \prop_item:Nn \g__pdfmeta_standard_prop {#1}
25   }
```

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

`\pdfmeta_standard_get:nN`

```
26 \cs_new_protected:Npn \pdfmeta_standard_get:nN #1 #2
27   {
28     \prop_get:NnN \g__pdfmeta_standard_prop {#1} #2
29   }
```

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

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

`\pdfmeta_standard_verify_p:n`
`\pdfmeta_standard_verify:n`*TF*

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

```
30 \prg_new_conditional:Npnn \pdfmeta_standard_verify:n #1 {T,F,TF}
31   {
32     \prop_if_in:NnTF \g__pdfmeta_standard_prop {#1}
33       {
34         \prg_return_false:
35       }
36       {
37         \prg_return_true:
38       }
39   }
```

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

`\pdfmeta_standard_verify:nn`*TF*

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

```
40 \prg_new_protected_conditional:Npnn \pdfmeta_standard_verify:nn #1 #2  {T,F,TF}
41   {
42     \prop_if_in:NnTF \g__pdfmeta_standard_prop {#1}
43       {
44         \cs_if_exist:cTF {__pdfmeta_standard_verify_handler_#1:nn}
45           {
46             \exp_args:Nnne
47             \use:c
```

```
48                {__pdfmeta_standard_verify_handler_#1:nn}
49                { #2 }
50                { \prop_item:Nn \g__pdfmeta_standard_prop {#1} }
51          }
52          {
53            \prg_return_false:
54          }
55      }
56      {
57        \prg_return_true:
58      }
59    }
```

(*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.

```
60 %
61 \cs_new_protected:Npn \__pdfmeta_standard_verify_handler_min_pdf_version:nn #1 #2
62  {
63    \pdf_version_compare:NnTF <
64      { #2 }
65      {\prg_return_false:}
66      {\prg_return_true:}
67  }
```

(*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

```
68 %
69 \cs_new_protected:Npn \__pdfmeta_standard_verify_handler_max_pdf_version:nn #1 #2
70  {
71    \pdf_version_compare:NnTF >
72      { #2 }
73      {\prg_return_false:}
74      {\prg_return_true:}
75  }
```

(*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.

```
76 \cs_new_protected:Npn \__pdfmeta_standard_verify_handler_named_actions:nn #1 #2
77  {
78    \tl_if_in:nnTF { #2 }{ #1 }
79      {\prg_return_true:}
80      {\prg_return_false:}
81  }
```

(*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.

```
82 \cs_new_protected:Npn \__pdfmeta_standard_verify_handler_annot_action_A:nn #1 #2
83   {
84     \tl_if_in:nnTF { #2 }{ #1 }
85       {\prg_return_true:}
86       {\prg_return_false:}
87   }
```

(*End of definition for* __pdfmeta_standard_verify_handler_annot_action_A:nn.)

This check is probably not needed, but for completeness

```
88 \cs_new_protected:Npn \__pdfmeta_standard_verify_handler_outputintent_subtype:nn #1 #2
89   {
90     \tl_if_eq:nnTF { #2 }{ #1 }
91       {\prg_return_true:}
92       {\prg_return_false:}
93   }
```

(*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:

```
94 \cs_new_protected:Npn \__pdfmeta_verify_pdfa_annot_flags:
95   {
96     \bitset_set_true:Nn  \l_pdfannot_F_bitset {Print}
97     \bitset_set_false:Nn \l_pdfannot_F_bitset {Hidden}
98     \bitset_set_false:Nn \l_pdfannot_F_bitset {Invisible}
99     \bitset_set_false:Nn \l_pdfannot_F_bitset {NoView}
100    \pdfannot_dict_put:nnn {link/URI}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
101    \pdfannot_dict_put:nnn {link/GoTo}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
102    \pdfannot_dict_put:nnn {link/GoToR}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
103    \pdfannot_dict_put:nnn {link/Launch}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
104    \pdfannot_dict_put:nnn {link/Named}{F}{ \bitset_to_arabic:N \l_pdfannot_F_bitset }
105  }
```

At begin document this should be checked:

```
106 \hook_gput_code:nnn {begindocument} {pdf}
107   {
108     \pdfmeta_standard_verify:nF { annot_flags }
109       { \__pdfmeta_verify_pdfa_annot_flags: }
110     \pdfmeta_standard_verify:nF { Trailer_no_Info }
111       { \__pdf_backend_omit_info:n {1} }
```

13

```
112    \pdfmeta_standard_verify:nF { no_CharSet }
113     { \__pdf_backend_omit_charset:n {1} }
114    \pdfmeta_standard_verify:nF { omit_CID }
115     { \__pdf_backend_omit_cidset:n {1} }
116    \pdfmeta_standard_verify:nnF { min_pdf_version }
117     { \pdf_version: }
118     { \msg_warning:nneee {pdf}{wrong-pdfversion}
119       {\pdf_version:}{low}
120       {
121        \pdfmeta_standard_item:n{type}
122        -
123        \pdfmeta_standard_item:n{level}
124       }
125     }
126    \pdfmeta_standard_verify:nnF { max_pdf_version }
127     { \pdf_version: }
128     { \msg_warning:nneee {pdf}{wrong-pdfversion}
129       {\pdf_version:}{high}
130       {
131        \pdfmeta_standard_item:n{type}
132        -
133        \pdfmeta_standard_item:n{level}
134       }
135     }
136    }
```

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.

<div style="float:left">

\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

</div>

```
137  \prop_new:c { g__pdfmeta_standard_pdf/A-1B_prop }
138  \prop_gset_from_keyval:cn { g__pdfmeta_standard_pdf/A-1B_prop }
139    {
140      ,name               = pdf/A-1B
141      ,type               = A
142      ,level              = 1
143      ,conformance        = B
144      ,year               = 2005
145      ,min_pdf_version = 1.4          %minimum
146      ,max_pdf_version = 1.4          %minimum
147      ,no_encryption      =
148      ,no_external_content =  % no F, FFilter, or FDecodeParms in stream dicts
149      ,no_embed_content = % no EF key in filespec, no /Type/EmbeddedFiles
150      ,max_string_size  = 65535
151      ,max_array_size   = 8191
152      ,max_dict_size    = 4095
153      ,max_obj_num      = 8388607
154      ,max_nest_qQ      = 28
155      ,named_actions    = {NextPage, PrevPage, FirstPage, LastPage}
156      ,annot_flags      =
157    %booleans. Only the existence of the key matter.
```

```
158    %If the entry is added it means a requirements is there
159    %(in most cases "don't use ...")
160    %
161    %===============
162    % Rule 6.1.13-1 CosDocument, isOptionalContentPresent == false
163    ,Catalog_no_OCProperties =
164    % Rule 6.9-4 The AS key shall not appear in any optional content configuration dictionar
165    % actually only starting with A-2 but doesn't harm here either
166    ,Catalog_OCProperties_no_AS=
167    %===============
168    % Rule 6.6.1-1: PDAction, S == "GoTo" || S == "GoToR" || S == "Thread"
169    %                || S == "URI" || S == "Named" || S == "SubmitForm"
170    % means: no /S/Launch, /S/Sound, /S/Movie, /S/ResetForm, /S/ImportData,
171    %        /S/JavaScript, /S/Hide
172    ,annot_action_A        = {GoTo,GoToR,Thread,URI,Named,SubmitForm}
173    %===============
174    % Rule 6.6.2-1: PDAnnot, Subtype != "Widget" || AA_size == 0
175    % means: no AA dictionary
176    ,annot_widget_no_AA      =
177    %===============
178    % Rule 6.9-2: PDAnnot, Subtype != "Widget" || (A_size == 0 && AA_size == 0)
179    % (looks like a tightening of the previous rule)
180    ,annot_widget_no_A_AA     =
181    %===============
182    % Rule 6.9-1 PDAcroForm, NeedAppearances == null || NeedAppearances == false
183    ,form_no_NeedAppearances =
184    %===============
185    %Rule 6.9-3 PDFormField, AA_size == 0
186    ,form_no_AA             =
187    %===============
188    % to be continued https://docs.verapdf.org/validation/pdfa-part1/
189    % - Outputintent/colorprofiles requirements
190    % an outputintent should be loaded and is unique.
191    ,outputintent_A         = {GTS_PDFA1}
192    % - no Alternates key in image dictionaries
193    % - no OPI, Ref, Subtype2 with PS key in xobjects
194    % - Interpolate  = false in images
195    % - no TR, TR2 in ExtGstate
196  }
197
198  %A-2b ==============
199  \prop_new:c { g__pdfmeta_standard_pdf/A-2B_prop }
200  \prop_gset_eq:cc
201    { g__pdfmeta_standard_pdf/A-2B_prop }
202    { g__pdfmeta_standard_pdf/A-1B_prop }
203  \prop_gput:cnn
204    { g__pdfmeta_standard_pdf/A-2B_prop }{name}{pdf/A-2B}
205  \prop_gput:cnn
206    { g__pdfmeta_standard_pdf/A-2B_prop }{year}{2011}
207  \prop_gput:cnn
208    { g__pdfmeta_standard_pdf/A-2B_prop }{level}{2}
209  % embedding files is allowed (with restrictions)
210  \prop_gremove:cn
211    { g__pdfmeta_standard_pdf/A-2B_prop }
```

```
212    { no_embed_content }
213  \prop_gput:cnn
214    { g__pdfmeta_standard_pdf/A-2B_prop }
215    { only_pdfa_embed_content }
216    {}
217  \prop_gput:cnn
218    { g__pdfmeta_standard_pdf/A-2B_prop }{max_pdf_version}{1.7}
219  \prop_gput:cnn
220    { g__pdfmeta_standard_pdf/A-2B_prop }{omit_CID}{}
221  % OCG layers are allowed (with restrictions)
222  \prop_gremove:cn
223    { g__pdfmeta_standard_pdf/A-2B_prop }
224    { Catalog_no_OCProperties }
225
226    %A-2u ==============
227  \prop_new:c { g__pdfmeta_standard_pdf/A-2U_prop }
228  \prop_gset_eq:cc
229    { g__pdfmeta_standard_pdf/A-2U_prop }
230    { g__pdfmeta_standard_pdf/A-2B_prop }
231  \prop_gput:cnn
232    { g__pdfmeta_standard_pdf/A-2U_prop }{name}{pdf/A-2U}
233  \prop_gput:cnn
234    { g__pdfmeta_standard_pdf/A-2U_prop }{conformance}{U}
235  \prop_gput:cnn
236    { g__pdfmeta_standard_pdf/A-2U_prop }{unicode}{}
237
238  %A-2a ==============
239  \prop_new:c { g__pdfmeta_standard_pdf/A-2A_prop }
240  \prop_gset_eq:cc
241    { g__pdfmeta_standard_pdf/A-2A_prop }
242    { g__pdfmeta_standard_pdf/A-2B_prop }
243  \prop_gput:cnn
244    { g__pdfmeta_standard_pdf/A-2A_prop }{name}{pdf/A-2A}
245  \prop_gput:cnn
246    { g__pdfmeta_standard_pdf/A-2A_prop }{conformance}{A}
247  \prop_gput:cnn
248    { g__pdfmeta_standard_pdf/A-2A_prop }{tagged}{}
249
250
251  %A-3b ==============
252  \prop_new:c { g__pdfmeta_standard_pdf/A-3B_prop }
253  \prop_gset_eq:cc
254    { g__pdfmeta_standard_pdf/A-3B_prop }
255    { g__pdfmeta_standard_pdf/A-2B_prop }
256  \prop_gput:cnn
257    { g__pdfmeta_standard_pdf/A-3B_prop }{name}{pdf/A-3B}
258  \prop_gput:cnn
259    { g__pdfmeta_standard_pdf/A-3B_prop }{year}{2012}
260  \prop_gput:cnn
261    { g__pdfmeta_standard_pdf/A-3B_prop }{level}{3}
262  % embedding files is allowed
263  \prop_gremove:cn
264    { g__pdfmeta_standard_pdf/A-3B_prop }
265    { only_pdfa_embed_content }
```

```
266  %A-3u ==============
267  \prop_new:c { g__pdfmeta_standard_pdf/A-3U_prop }
268  \prop_gset_eq:cc
269    { g__pdfmeta_standard_pdf/A-3U_prop }
270    { g__pdfmeta_standard_pdf/A-3B_prop }
271  \prop_gput:cnn
272    { g__pdfmeta_standard_pdf/A-3U_prop }{name}{pdf/A-3U}
273  \prop_gput:cnn
274    { g__pdfmeta_standard_pdf/A-3U_prop }{conformance}{U}
275  \prop_gput:cnn
276    { g__pdfmeta_standard_pdf/A-3U_prop }{unicode}{}
277
278  %A-3a ==============
279  \prop_new:c { g__pdfmeta_standard_pdf/A-3A_prop }
280  \prop_gset_eq:cc
281    { g__pdfmeta_standard_pdf/A-3A_prop }
282    { g__pdfmeta_standard_pdf/A-3B_prop }
283  \prop_gput:cnn
284    { g__pdfmeta_standard_pdf/A-3A_prop }{name}{pdf/A-3A}
285  \prop_gput:cnn
286    { g__pdfmeta_standard_pdf/A-3A_prop }{conformance}{A}
287  \prop_gput:cnn
288    { g__pdfmeta_standard_pdf/A-3A_prop }{tagged}{}
289
290  %A-4 ==============
291  \prop_new:c { g__pdfmeta_standard_pdf/A-4_prop }
292  \prop_gset_eq:cc
293    { g__pdfmeta_standard_pdf/A-4_prop }
294    { g__pdfmeta_standard_pdf/A-3U_prop }
295  \prop_gput:cnn
296    { g__pdfmeta_standard_pdf/A-4_prop }{name}{pdf/A-4}
297  \prop_gput:cnn
298    { g__pdfmeta_standard_pdf/A-4_prop }{level}{4}
299  \prop_gput:cnn
300    { g__pdfmeta_standard_pdf/A-4_prop }{min_pdf_version}{2.0}
301  \prop_gput:cnn
302    { g__pdfmeta_standard_pdf/A-4_prop }{year}{2020}
303  \prop_gput:cnn
304    { g__pdfmeta_standard_pdf/A-4_prop }{no_CharSet}{}
305  \prop_gput:cnn
306    { g__pdfmeta_standard_pdf/A-4_prop }{Trailer_no_Info}{}
307  \prop_gput:cnn
308    { g__pdfmeta_standard_pdf/A-4_prop }{only_pdfa_embed_content}{}
309  \prop_gremove:cn
310    { g__pdfmeta_standard_pdf/A-4_prop }{conformance}
311  \prop_gremove:cn
312    { g__pdfmeta_standard_pdf/A-4_prop }{max_pdf_version}
313  \prop_gremove:cn
314    { g__pdfmeta_standard_pdf/A-4_prop }{Catalog_OCProperties_no_AS}
315  %A-4f ==============
316  \prop_new:c { g__pdfmeta_standard_pdf/A-4F_prop }
317  \prop_gset_eq:cc
318    { g__pdfmeta_standard_pdf/A-4F_prop }
319    { g__pdfmeta_standard_pdf/A-4_prop }
```

```
320 \prop_gput:cnn
321   { g__pdfmeta_standard_pdf/A-4F_prop }{conformance}{F}
322 % containsEmbeddedFiles == true ISO 19005-4:2020, Clause: 6.9, Test number: 5
323 \prop_gput:cnn
324   { g__pdfmeta_standard_pdf/A-4F_prop }{Catalog_EmbeddedFiles}{}
325 % can contain any file
326 \prop_gremove:cn
327   { g__pdfmeta_standard_pdf/A-4F_prop }{only_pdfa_embed_content}
```

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

3.1.5 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

```
328 \AddToHook{begindocument/end}
329 {
330   \pdfmeta_standard_verify:nF{Catalog_EmbeddedFiles}
331     {
332      \tl_gput_right:Nn\g__kernel_pdfmanagement_end_run_code_tl
333        {
334          \pdfdict_if_empty:nT { g__pdf_Core/Catalog/Names/EmbeddedFiles }
335            {
336              \group_begin:
337              \pdfdict_put:nne {l_pdffile/Filespec} {Desc}{(note~about~PDF/A-4F)}
338              \pdfdict_put:nnn { l_pdffile/Filespec }{AFRelationship} { /Unspecified }
339              \pdffile_embed_stream:nnN
340                {The~document~was~declared~to~be~of~type~PDF/A-4f~but~hasn't~any~attachments.~
341                 LaTeX~therefore~added~this~dummy~file.}
342                {pdf-A4f.txt}
343                \l__pdfmeta_tmpa_tl
344              \exp_args:Nne \__pdf_backend_Names_gpush:nn{EmbeddedFiles}{(pdf-A4f)~\l__pdfmeta_t
345              \group_end:
346            }
347        }
348    }
349 }
```

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.

```
350 \AddToHook{pdfmeta/xmp}
351  {
352     \pdfmeta_standard_verify:nF{no_embed_content}
353       {
354         \bool_lazy_or:nnT
355           { ! \int_if_zero_p:n { \g_pdffile_embed_pdfa_int } }
356           { ! \int_if_zero_p:n { \g_pdffile_embed_nonpdfa_int } }
357           {
358             \prop_get:NnNT\g__pdfmeta_standard_prop { name }\l__pdfmeta_tmpa_tl
359               {
360                 \msg_warning:nne { pdf } { validation-failure }
361                   {
```

18

```
362              Embedded~files~detected.\iow_newline:
363              This~is~not~allowed~in~standard~\l__pdfmeta_tmpa_tl
364            }
365          }
366        }
367      }
368    \pdfmeta_standard_verify:nF {only_pdfa_embed_content}
369      {
370        \int_if_zero:nF  { \g_pdffile_embed_nonpdfa_int }
371          {
372            \prop_get:NnNT\g__pdfmeta_standard_prop { name }\l__pdfmeta_tmpa_tl
373              {
374                \str_if_eq:VnTF {\l__pdfmeta_tmpa_tl} { pdf/A-4 }
375                  {
376                    \prop_gset_eq:cc
377                      { g__pdfmeta_standard_prop }
378                      { g__pdfmeta_standard_pdf/A-4F_prop }
379                    \msg_warning:nne { pdf } { validation-failure }
380                      {
381                        Embedded~non-PDF~files~detected.\iow_newline:
382                        Switching~standard~from~PDF/A-4~to~PDF/A-4F
383                      }
384                  }
385                  {
386                    \msg_warning:nne { pdf } { validation-failure }
387                      {
388                        Embedded~non-PDF~files~detected.\iow_newline:
389                        This~is~not~allowed~in~standard~\l__pdfmeta_tmpa_tl
390                      }
391                  }
392              }
393          }
394      }
395  }
```

3.1.6 Colorprofiles and Outputintents

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

Adding a profile and an intent is technically easy:

1. Embed the profile as stream with

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

2. Write a /OutputIntent dictionary for this

   ```
   \pdf_object_unnamed_write:ne {dict}
    {
      /Type /OutputIntent
      /S /GTS_PDFA1  % or GTS_PDFX or ISO_PDFE1 or ...
      /DestOutputProfile \pdf_object_ref_last: % ref the color profile
      /OutputConditionIdentifier ...
   ```

```
    ... %more info
  }
```

3. Reference the dictionary in the catalog:

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

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

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

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

(*End of definition for* \g__pdfmeta_outputintents_prop.)

Some keys to fill the property.

```
397 \keys_define:nn { document / metadata }
398   {
399     colorprofiles .code:n =
400       {
401         \keys_set:nn { document / metadata / colorprofiles }{#1}
402       }
403   }
404 \keys_define:nn { document / metadata / colorprofiles }
405  {
406    ,A .code:n =
407      {
408        \tl_if_blank:nF {#1}
409          {
410            \prop_gput:Nnn \g__pdfmeta_outputintents_prop
411            { GTS_PDFA1  } {#1}
412          }
413      }
414    ,a .code:n =
415      {
416        \tl_if_blank:nF {#1}
417          {
418            \prop_gput:Nnn \g__pdfmeta_outputintents_prop
419            { GTS_PDFA1  } {#1}
420          }
421      }
422    ,X .code:n =
423      {
424        \tl_if_blank:nF {#1}
425          {
426            \prop_gput:Nnn \g__pdfmeta_outputintents_prop
427            { GTS_PDFX  } {#1}
428          }
429      }
430    ,x .code:n =
431      {
```

```
432        \tl_if_blank:nF {#1}
433          {
434            \prop_gput:Nnn \g__pdfmeta_outputintents_prop
435              { GTS_PDFX  } {#1}
436          }
437      }
438    ,unknown .code:n =
439      {
440        \tl_if_blank:nF {#1}
441          {
442            \exp_args:NNo
443              \prop_gput:Nnn \g__pdfmeta_outputintents_prop
444                { \l_keys_key_str  } {#1}
445          }
446      }
447  }
```

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

```
448  \pdfdict_new:n    {l_pdfmeta/outputintent}
449  \pdfdict_put:nnn {l_pdfmeta/outputintent}
450    {Type}{/OutputIntent}
451  \prop_const_from_keyval:cn { c__pdfmeta_colorprofile_sRGB.icc}
452    {
453      ,OutputConditionIdentifier=IEC~sRGB
454      ,Info=IEC~61966-2.1~Default~RGB~colour~space~-~sRGB
455      ,RegistryName=http://www.iec.ch
456      ,N = 3
457    }
458  \prop_const_from_keyval:cn { c__pdfmeta_colorprofile_FOGRA39L_coated.icc}
459    {
460      ,OutputConditionIdentifier=FOGRA39L~Coated
461      ,Info={Offset~printing,~according~to~ISO~12647-2:2004/Amd~1,~OFCOM,~ %
462              paper~type~1~or~2~=~coated~art,~115~g/m2,~tone~value~increase~
463              curves~A~(CMY)~and~B~(K)}
464      ,RegistryName=http://www.fogra.org
465      ,N = 4
466    }
```

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

```
467  \cs_new_protected:Npn \__pdfmeta_embed_colorprofile:n #1%#1 file name
468    {
469      \pdf_object_if_exist:nF { __color_icc_ #1 }
470        {
471          \pdf_object_new:n  { __color_icc_ #1 }
472          \pdf_object_write:nne { __color_icc_ #1 } { fstream }
473            {
474              {/N\c_space_tl
475                \prop_item:cn{c__pdfmeta_colorprofile_#1}{N}
476              }
```

```
477              {#1}
478            }
479          }
480      }
481
482  \cs_new_protected:Npn \__pdfmeta_write_outputintent:nn #1 #2 %#1 file name, #2 subtype
483    {
484      \group_begin:
485      \pdfdict_put:nne {l_pdfmeta/outputintent}{S}{/\str_convert_pdfname:n{#2}}
486      \pdfdict_put:nne {l_pdfmeta/outputintent}
487        {DestOutputProfile}
488        {\pdf_object_ref:n{ __color_icc_ #1 }}
489      \clist_map_inline:nn { OutputConditionIdentifier, Info, RegistryName }
490        {
491          \prop_get:cnNT
492          { c__pdfmeta_colorprofile_#1}
493          { ##1 }
494          \l__pdfmeta_tmpa_tl
495          {
496            \pdf_string_from_unicode:nVN {utf8/string}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_st
497            \pdfdict_put:nne
498              {l_pdfmeta/outputintent}{##1}{\l__pdfmeta_tmpa_str}
499          }
500        }
501      \pdf_object_unnamed_write:ne {dict}{\pdfdict_use:n {l_pdfmeta/outputintent} }
502      \pdfmanagement_add:nne {Catalog}{OutputIntents}{\pdf_object_ref_last:}
503      \group_end:
504    }
```

(*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

```
505  \AddToHook{begindocument/end}
506    {
507      \pdfmeta_standard_verify:nTF {outputintent_A}
508        {
509          \prop_map_inline:Nn \g__pdfmeta_outputintents_prop
510            {
511              \prop_if_exist:cTF {c__pdfmeta_colorprofile_#2}
512                {
513                  \__pdfmeta_embed_colorprofile:n
514                  {#2}
515                  \__pdfmeta_write_outputintent:nn
516                  {#2}
517                  {#1}
518                }
519                {
520                  \msg_warning:nnn{pdfmeta}{colorprofile-undefined}{#2}
521                }
522            }
523        }
```

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.

```
524        {
525          \exp_args:NNe
526          \prop_if_in:NnF
527            \g__pdfmeta_outputintents_prop
528            { \pdfmeta_standard_item:n { outputintent_A } }
529            {
530              \exp_args:NNe
531              \prop_gput:Nnn
532                \g__pdfmeta_outputintents_prop
533                { \pdfmeta_standard_item:n { outputintent_A } }
534                { sRGB.icc }
535            }
536          \exp_args:NNe
537          \prop_get:NnN
538            \g__pdfmeta_outputintents_prop
539            { \pdfmeta_standard_item:n { outputintent_A } }
540            \l__pdfmeta_tmpb_tl
541          \prop_if_exist:cTF {c__pdfmeta_colorprofile_\l__pdfmeta_tmpb_tl}
542            {
543              \exp_args:NV \__pdfmeta_embed_colorprofile:n \l__pdfmeta_tmpb_tl
544              \prop_map_inline:Nn \g__pdfmeta_outputintents_prop
545                {
546                  \exp_args:NV
547                    \__pdfmeta_write_outputintent:nn
548                      \l__pdfmeta_tmpb_tl
549                      { #1 }
550                }
551            }
552            {
553              \msg_warning:nne{pdfmeta}{colorprofile-undefined}{\l__pdfmeta_tmpb_tl}
554            }
555        }
556    }
```

3.2 Regression test

This is simply a copy of the backend function.

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

4   XMP-Metadata implementation

\g__pdfmeta_xmp_bool   This boolean decides if the metadata are included

```
559 \bool_new:N\g__pdfmeta_xmp_bool
560 \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.

```
561 \hook_gput_code:nnn{pdfmanagement/add}{pdfmanagement}
562   {
563     \pdfmanagement_add:nne {Info}{Producer}{(\c_sys_engine_exec_str-\c_sys_engine_version_str
564     \pdfmanagement_add:nne {Info}{Creator}{(LaTeX)}
565   }
```

4.1 New document keys

```
566 \keys_define:nn { document / metadata }
567  {
568    _pdfstandard .choices:nn =
569      {A-1B,A-2A,A-2B,A-2U,A-3A,A-3B,A-3U,A-4}
570        {
571          \prop_gset_eq:Nc \g__pdfmeta_standard_prop { g__pdfmeta_standard_pdf/#1 _prop }
572          \AddToDocumentProperties [document]{pdfstandard}{#1}
573        },
574    _pdfstandard / A-4F .code:n =
575        {
576          \prop_gset_eq:Nc \g__pdfmeta_standard_prop { g__pdfmeta_standard_pdf/A-
   4F_prop }
577          \AddToDocumentProperties [document]{pdfstandard}{A-4F}
578        },
579    _pdfstandard / A-4E .code:n =
580        {
581          \prop_gset_eq:Nc \g__pdfmeta_standard_prop { g__pdfmeta_standard_pdf/A-
   4E_prop }
582          \AddToDocumentProperties [document]{pdfstandard}{A-4E}
583        },
584    _pdfstandard / unknown .code:n =
585        {
586          \msg_error:nnn{pdf}{unknown-standard}{#1}
587        },
588    _pdfstandard / X-4 .code:n =
589      {
590        \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-4}
591        \__pdfmeta_xmp_add_pdfxid:
592      },
593    _pdfstandard / X-4p .code:n =
594      {
595        \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-4p}
596        \__pdfmeta_xmp_add_pdfxid:
597      },
598    _pdfstandard / X-5g .code:n =
599      {
600        \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5g}
601        \__pdfmeta_xmp_add_pdfxid:
602      },
603    _pdfstandard / X-5n .code:n =
604      {
605        \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5n}
606        \__pdfmeta_xmp_add_pdfxid:
607      },
```

```
608    _pdfstandard / X-5pg .code:n =
609      {
610        \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-5pg}
611        \__pdfmeta_xmp_add_pdfxid:
612      },
613    _pdfstandard / X-6 .code:n =
614      {
615        \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6p}
616        \__pdfmeta_xmp_add_pdfxid:
617      },
618    _pdfstandard / X-6n .code:n =
619      {
620        \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6n}
621        \__pdfmeta_xmp_add_pdfxid:
622      },
623    _pdfstandard / X-6p .code:n =
624      {
625        \AddToDocumentProperties [document]{pdfstandard-X}{PDF/X-6p}
626        \__pdfmeta_xmp_add_pdfxid:
627      },
628    _pdfstandard / UA-1 .code:n =
629      {
630        \AddToDocumentProperties [document]{pdfstandard-UA}{{1}{}}
631        \AddToHook{begindocument/before}
632          {
633            \prop_gput:Nnn \g__pdfmeta_standard_prop {omit_CID}{}
634            \pdf_version_compare:NnF < {2.0}
635              {
636                \msg_warning:nneee
637                {pdf}{wrong-pdfversion}
638                {\pdf_version:}{high}{UA-1}
639              }
640          }
641      },
```

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.

```
642    _pdfstandard / UA-2 .code:n =
643      {
644        \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.

```
645        %\AddToHook{begindocument/before}
646        %{\prop_gput:Nnn \g__pdfmeta_standard_prop {Trailer_no_Info}{}}
647        \AddToHook{begindocument/before}
648          {
649            \__pdfmeta_xmp_wtpdf_accessibility_declaration:
650            \__pdfmeta_xmp_wtpdf_reuse_declaration:
651            \pdf_version_compare:NnT < {2.0}
652              {
653                \msg_warning:nneee
654                {pdf}{wrong-pdfversion}
655                {\pdf_version:}{low}{UA-2}
```

```
656                }
657            }
658        },
659        xmp  .choice:,
660        xmp / true  .code:n = { \bool_gset_true:N \g__pdfmeta_xmp_bool },
661        xmp / false .code:n = { \bool_gset_false:N \g__pdfmeta_xmp_bool},
662        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.

```
663        xmp / wtpdf .code:n =
664          {
665            \keys_set:nn {__pdfmeta/xmp}{#1}
666          },
667    }
668  \keys_define:nn {__pdfmeta/xmp}
669    {
670      reuse .choice:,
671      reuse / true .code:n = \__pdfmeta_xmp_wtpdf_reuse_declaration:,
672      reuse / false .code:n =
673        {
674          \cs_set_eq:NN \__pdfmeta_xmp_wtpdf_reuse_declaration: \prg_do_nothing:
675        },
676      accessibility .choice:,
677      accessibility / true .code:n = \__pdfmeta_xmp_wtpdf_accessibility_declaration:,
678      accessibility /false .code:n =
679        {
680          \cs_set_eq:NN \__pdfmeta_xmp_wtpdf_accessibility_declaration: \prg_do_nothing:
681        },
682    }
```

XMP debugging option

```
683  \bool_new:N \g__pdfmeta_xmp_export_bool
684  \str_new:N  \g__pdfmeta_xmp_export_str
685
686  \keys_define:nn { document / metadata }
687    {
688      ,debug / xmp-export .choice:
689      ,debug / xmp-export / true .code:n=
690        {
691          \bool_gset_true:N \g__pdfmeta_xmp_export_bool
692          \str_gset_eq:NN \g__pdfmeta_xmp_export_str \c_sys_jobname_str
693        }
694      ,debug / xmp-export / false .code:n =
695        {
696          \bool_gset_false:N \g__pdfmeta_xmp_export_bool
697        }
698      ,debug / xmp-export /unknown .code:n =
699        {
700          \bool_gset_true:N \g__pdfmeta_xmp_export_bool
701          \str_gset:Nn \g__pdfmeta_xmp_export_str { #1 }
702        }
703      ,debug / xmp-export .default:n = true
704    }
```

4.2 Messages

```
705 \msg_new:nnn{pdfmeta}{xmp-defined}{The~XMP~#1~`#2`~is~already~declared}
706 \msg_new:nnn{pdfmeta}{xmp-undefined}{The~XMP~#1~`#2`~is~undefined}
707 \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:

```
708 \bool_lazy_or:nnTF
709   { \sys_if_engine_luatex_p: }
710   { \sys_if_engine_xetex_p: }
711   {
712     \cs_new:Npn \__pdfmeta_xmp_generate_bom:
713       { \char_generate:nn {"FEFF}{12} }
714   }
715   {
716     \cs_new:Npn \__pdfmeta_xmp_generate_bom:
717       {
718         \char_generate:nn {"EF}{12}
719         \char_generate:nn {"BB}{12}
720         \char_generate:nn {"BF}{12}
721       }
722   }
```

(*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

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

(*End of definition for* \l__pdfmeta_xmp_indent_int.)

__pdfmeta_xmp_indent:
__pdfmeta_xmp_indent:n
__pdfmeta_xmp_incr_indent:
__pdfmeta_xmp_decr_indent:

```
724 \cs_new:Npn \__pdfmeta_xmp_indent:
725   {
726     \iow_newline:
727     \prg_replicate:nn {\l__pdfmeta_xmp_indent_int}{\c_space_tl}
728   }
729
730 \cs_new:Npn \__pdfmeta_xmp_indent:n #1
731   {
732     \iow_newline:
733     \prg_replicate:nn {#1}{\c_space_tl}
734   }
735
736 \cs_new_protected:Npn \__pdfmeta_xmp_incr_indent:
```

27

```
737  {
738      \int_incr:N \l__pdfmeta_xmp_indent_int
739  }
740
741  \cs_new_protected:Npn \__pdfmeta_xmp_decr_indent:
742  {
743      \int_decr:N \l__pdfmeta_xmp_indent_int
744  }
```

(*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

```
745  \regex_new:N \l__pdfmeta_xmp_date_regex
746  \regex_set:Nn \l__pdfmeta_xmp_date_regex
747  {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.

```
748  \cs_new_protected:Npn \__pdfmeta_xmp_date_split:nN #1 #2 %#1 date, #2 seq
749  {
750      \regex_split:NnN \l__pdfmeta_xmp_date_regex {#1} #2
751  }
752  \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.

```
753  \cs_new:Npn\__pdfmeta_xmp_print_date:N #1 % seq
754  {
755      \tl_if_blank:eTF { \seq_item:Nn #1 {1} }
756      {
757          \seq_item:Nn #1 {2} %year
758          -
759          \seq_item:Nn #1 {3} %month
760          -
761          \seq_item:Nn #1 {4} % day
762          \tl_if_blank:eF
763            { \seq_item:Nn #1 {5} }
764            { T \seq_item:Nn #1 {5} } %hour
765          \tl_if_blank:eF
766            { \seq_item:Nn #1 {6} }
767            { : \seq_item:Nn #1 {6} } %minutes
768          \tl_if_blank:eF
769            { \seq_item:Nn #1 {7} }
```

```
770        { : \seq_item:Nn #1 {7} } %seconds
771      \seq_item:Nn #1 {8}   %Z,+,-
772      \seq_item:Nn #1 {9}
773      \tl_if_blank:eF
774        { \seq_item:Nn #1 {10} }
775        { : \seq_item:Nn #1 {10} }
776    }
777    {
778      \seq_item:Nn #1 {1}
779    }
780  }
```

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

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

```
781 \tl_new:N  \l__pdfmeta_xmp_currentdate_tl
782 \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.

```
783 \cs_new_protected:Npn \__pdfmeta_xmp_date_get:nNN #1 #2 #3
784   %#1 property, #2 tl var with PDF date, #3 seq for split date
785   {
786     \tl_set:Ne #2 { \GetDocumentProperties{#1} }
787     \tl_if_blank:VTF #2
788       {
789         \seq_set_eq:NN #3 \l__pdfmeta_xmp_currentdate_seq
790         \tl_set_eq:NN  #2 \l__pdfmeta_xmp_currentdate_tl
791       }
792       {
793         \__pdfmeta_xmp_date_split:VN #2 #3
794       }
795   }
```

(*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`

```
796 \cs_new_protected:Npn \__pdfmeta_xmp_create_uuid:nN #1 #2
797   {
798     \str_set:Ne#2 {\str_lowercase:f{\tex_mdfivesum:D{#1}}}
799     \str_set:Ne#2
800       { uuid:
801         \str_range:Nnn #2{1}{8}
802         -\str_range:Nnn#2{9}{12}
803         -4\str_range:Nnn#2{13}{15}
804         -8\str_range:Nnn#2{16}{18}
```

```
805          -\str_range:Nnn#2{19}{30}
806        }
807    }
```

(*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.

```
808 \cs_new_protected:Npn \__pdfmeta_xmp_sanitize:nN #1 #2
809 %#1 input string, #2 str with the output
810   {
811     \group_begin:
812       \text_declare_purify_equivalent:Nn \& {\tl_to_str:N & }
813       \text_declare_purify_equivalent:Nn \texttilde {\c_tilde_str}
814       \tl_set:Ne \l__pdfmeta_tmpa_tl { \text_purify:n {#1} }
815       \str_gset:Ne \g__pdfmeta_tmpa_str { \tl_to_str:N \l__pdfmeta_tmpa_tl }
816       \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {&}{&amp;}
817       \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {<}{&lt;}
818       \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {>}{&gt;}
819       \str_greplace_all:Nnn\g__pdfmeta_tmpa_str {"}{&quot;}
820     \group_end:
821       \str_set_eq:NN #2 \g__pdfmeta_tmpa_str
822   }
823
824 \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`

```
825 \tl_new:N \l__pdfmeta_xmp_doclang_tl
826 \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`

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

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

```
829 \cs_new_protected:Npn \__pdfmeta_xmp_lang_get:nNN #1 #2 #3
830 % #1 text, #2 tl var for lang match (or default), #3 tl var for text
831   {
832     \regex_extract_once:NnN \l__pdfmeta_xmp_lang_regex {#1}\l__pdfmeta_tmpa_seq
833     \seq_if_empty:NTF \l__pdfmeta_tmpa_seq
834       {
835         \tl_set:Nn #2 \l__pdfmeta_xmp_metalang_tl
836         \tl_set:Nn #3 {#1}
837       }
838       {
839         \tl_set:Ne #2 {\seq_item:Nn\l__pdfmeta_tmpa_seq{2}}
840         \tl_set:Ne #3 {\seq_item:Nn\l__pdfmeta_tmpa_seq{3}}
841       }
842   }
843 \cs_generate_variant:Nn \__pdfmeta_xmp_lang_get:nNN {eNN,VNN}
```

4.5 Filling the packet

This tl var that holds the whole packet

`\g__pdfmeta_xmp_packet_tl`

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

(*End of definition for* `\g__pdfmeta_xmp_packet_tl`.)

4.5.1 Helper commands to add lines and lists

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

```
845 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_chunk:n #1
846   {
847     \tl_gput_right:Ne\g__pdfmeta_xmp_packet_tl
848       {
849         \__pdfmeta_xmp_indent:  \exp_not:n{#1}
850       }
851   }
852 \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.

```
853 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_chunk:nN #1 #2
854   {
855     \tl_put_right:Ne#2
856       {
857         \__pdfmeta_xmp_indent:  \exp_not:n{#1}
858       }
859   }
860 \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.

```
861 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_open:nn #1 #2 %#1 prefix #2 name
862   {
863     \__pdfmeta_xmp_add_packet_chunk:n {<#1:#2>}
864     \__pdfmeta_xmp_incr_indent:
865   }
866 \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.

```
867 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_open_attr:nnn #1 #2 #3
868   %#1 prefix #2 name #3 attr
869   {
870     \__pdfmeta_xmp_add_packet_chunk:n {<#1:#2~#3>}
871     \__pdfmeta_xmp_incr_indent:
872   }
873 \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.

```
874 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_close:nn #1 #2 %#1 prefix #2:name
875   {
876     \__pdfmeta_xmp_decr_indent:
877     \__pdfmeta_xmp_add_packet_chunk:n {</#1:#2>}
878   }
```

(*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.

```
879 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_line:nnn #1 #2 #3
880   %#1 prefix #2 name #3 content
881   {
882     \tl_if_blank:nF {#3}
883       {
884         \__pdfmeta_xmp_sanitize:nN {#3}\l__pdfmeta_tmpa_str
885         \__pdfmeta_xmp_add_packet_chunk:e {<#1:#2>\l__pdfmeta_tmpa_str</#1:#2>}
886       }
887   }
888 \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.

```
889 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_line:nnnN #1 #2 #3 #4
890   %#1 prefix #2 name #3 content #4 tl_var to prebuilt.
```

```
891   {
892     \tl_if_blank:nF {#3}
893       {
894        \__pdfmeta_xmp_sanitize:nN {#3}\l__pdfmeta_tmpa_str
895        \__pdfmeta_xmp_add_packet_chunk:eN {<#1:#2>\l__pdfmeta_tmpa_str</#1:#2>} #4
896       }
897   }
898 \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

```
899 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_line_attr:nnnn #1 #2 #3 #4
900 %#1 prefix #2 name #3 attribute #4 content
901   {
902     \tl_if_blank:nF {#4}
903       {
904        \__pdfmeta_xmp_sanitize:nN {#4}\l__pdfmeta_tmpa_str
905        \__pdfmeta_xmp_add_packet_chunk:e {<#1:#2~#3>\l__pdfmeta_tmpa_str</#1:#2>}
906       }
907   }
908 \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

```
909 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_line_default:nnnn #1 #2 #3 #4
910   % #1 prefix #2 name #3 default #4 content
911   {
912     \tl_if_blank:nTF { #4 }
913       {
914        \tl_set:Nn  \l__pdfmeta_tmpa_tl  {#3}
915       }
916       {
917        \tl_set:Nn \l__pdfmeta_tmpa_tl  {#4}
918       }
919     \__pdfmeta_xmp_add_packet_line:nnV {#1}{#2}\l__pdfmeta_tmpa_tl
920   }
921 \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:

```
922 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_list_simple:nnnn #1 #2 #3 #4
923   %#1 prefix, #2 name,  #3 type (Seq/Bag/Alt) #4 a clist
924   {
925     \clist_if_empty:nF { #4 }
926       {
927        \__pdfmeta_xmp_add_packet_open:nn {#1}{#2}
928        \__pdfmeta_xmp_add_packet_open:nn {rdf}{#3}
929         \clist_map_inline:nn {#4}
```

33

```
930              {
931                \__pdfmeta_xmp_add_packet_line:nnn
932                 {rdf}{li}{##1}
933              }
934            \__pdfmeta_xmp_add_packet_close:nn{rdf}{#3}
935            \__pdfmeta_xmp_add_packet_close:nn {#1}{#2}
936        }
937      }
938  \cs_generate_variant:Nn \__pdfmeta_xmp_add_packet_list_simple:nnnn {nnnV,nnne}
```

Here we check also for the language.

```
939  \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_list:nnnn #1 #2 #3 #4
940    %#1 prefix, #2 name,  #3 type (Seq/Bag/Alt) #4 a clist
941    {
942      \clist_if_empty:nF { #4 }
943        {
944          \__pdfmeta_xmp_add_packet_open:nn {#1}{#2}
945            \__pdfmeta_xmp_add_packet_open:nn {rdf}{#3}
946            \clist_map_inline:nn {#4}
947              {
948                \__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!

```
949                \tl_if_eq:eeTF{\l__pdfmeta_tmpa_tl}{\l__pdfmeta_xmp_metalang_tl}
950                  {
951                    \__pdfmeta_xmp_add_packet_line_attr:nneV
952                     {rdf}{li}{xml:lang="x-default" }\l__pdfmeta_tmpb_tl
953                  }
954                  {
955                    \__pdfmeta_xmp_add_packet_line_attr:nneV
956                     {rdf}{li}{xml:lang="\l__pdfmeta_tmpa_tl" }\l__pdfmeta_tmpb_tl
957                  }
958              }
959          \__pdfmeta_xmp_add_packet_close:nn{rdf}{#3}
960            \__pdfmeta_xmp_add_packet_close:nn {#1}{#2}
961        }
962    }
963  \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.

```
964  \cs_new_protected:Npn \__pdfmeta_xmp_build_packet:
965    {
```

Get the main languages

```
966      \tl_set:Ne \l__pdfmeta_xmp_doclang_tl  {\GetDocumentProperties{document/lang}}
967      \tl_set:Ne \l__pdfmeta_xmp_metalang_tl {\GetDocumentProperties{hyperref/pdfmetalang}}
```

```
968      \tl_if_blank:VT \l__pdfmeta_xmp_metalang_tl
969        { \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

```
970      \__pdfmeta_xmp_build_iptc_data:N \l__pdfmeta_xmp_iptc_data_tl
971      \tl_if_empty:NT \l__pdfmeta_xmp_iptc_data_tl
972        {
973          \seq_remove_all:Nn \l__pdfmeta_xmp_schema_seq { Iptc4xmpCore }
974        }
```

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

```
975        \__pdfmeta_xmp_add_packet_chunk:e
976        {<?xpacket~begin="\__pdfmeta_xmp_generate_bom:"~id="W5M0MpCehiHzreSzNTczkc9d"?>}
977        \__pdfmeta_xmp_add_packet_open:nn{x}{xmpmeta~xmlns:x="adobe:ns:meta/"}
978        \__pdfmeta_xmp_add_packet_open:ne{rdf}
979          {RDF~xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns\c_hash_str"}
```

The rdf namespaces

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

The extensions

```
982        \__pdfmeta_xmp_add_packet_open:nn{pdfaExtension}{schemas}
983        \__pdfmeta_xmp_add_packet_open:nn {rdf}{Bag}
984          \seq_map_inline:Nn \l__pdfmeta_xmp_schema_seq
985            {
986              \tl_use:c { g__pdfmeta_xmp_schema_##1_tl }
987            }
988        \__pdfmeta_xmp_add_packet_close:nn {rdf}{Bag}
989        \__pdfmeta_xmp_add_packet_close:nn {pdfaExtension}{schemas}
```

Now starts the part with the data.

```
990      % data
991          \__pdfmeta_xmp_build_pdf:
992          \__pdfmeta_xmp_build_xmpRights:
993          \__pdfmeta_xmp_build_standards: %pdfaid,pdfxid,pdfuaid
994          \__pdfmeta_xmp_build_pdfd:
995          \__pdfmeta_xmp_build_dc:
996          \__pdfmeta_xmp_build_photoshop:
997          \__pdfmeta_xmp_build_xmp:
998          \__pdfmeta_xmp_build_xmpMM:
999          \__pdfmeta_xmp_build_prism:
1000         \__pdfmeta_xmp_build_iptc:
1001         \__pdfmeta_xmp_build_user: %user additions
1002     % end
1003       \__pdfmeta_xmp_add_packet_close:nn {rdf}{Description}
1004       \__pdfmeta_xmp_add_packet_close:nn {rdf}{RDF}
1005     \__pdfmeta_xmp_add_packet_close:nn {x}{xmpmeta}
```

```
1006      \int_set:Nn  \l__pdfmeta_xmp_indent_int{20}
1007      \prg_replicate:nn{10}{\__pdfmeta_xmp_add_packet_chunk:n {}}
1008      \int_zero:N \l__pdfmeta_xmp_indent_int
1009      \__pdfmeta_xmp_add_packet_chunk:n {<?xpacket~end="w"?>}
1010  }
```

(*End of definition for* __pdfmeta_xmp_build_packet:.)

4.6 Building the chunks: rdf namespaces

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

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

```
1011  \str_new:N  \g__pdfmeta_xmp_xmlns_tl
1012  \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

```
1013  \cs_new_protected:Npn \__pdfmeta_xmp_xmlns_new:nn #1 #2
1014    {
1015      \prop_gput:Nnn \g__pdfmeta_xmp_xmlns_prop {#1}{#2}
1016      \tl_gput_right:Ne \g__pdfmeta_xmp_xmlns_tl
1017        {
1018          \__pdfmeta_xmp_indent:n{4} xmlns:\exp_not:n{#1="#2"}
1019        }
1020    }
```

(*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.

```
1021  \__pdfmeta_xmp_xmlns_new:nn {pdf}      {http://ns.adobe.com/pdf/1.3/}
1022  \__pdfmeta_xmp_xmlns_new:nn {xmpRights}{http://ns.adobe.com/xap/1.0/rights/}
1023  \__pdfmeta_xmp_xmlns_new:nn {dc}       {http://purl.org/dc/elements/1.1/}
1024  \__pdfmeta_xmp_xmlns_new:nn {photoshop}{http://ns.adobe.com/photoshop/1.0/}
1025  \__pdfmeta_xmp_xmlns_new:nn {xmp}      {http://ns.adobe.com/xap/1.0/}
1026  \__pdfmeta_xmp_xmlns_new:nn {xmpMM}    {http://ns.adobe.com/xap/1.0/mm/}
1027  \__pdfmeta_xmp_xmlns_new:nn {stEvt}
1028    {http://ns.adobe.com/xap/1.0/sType/ResourceEvent\c_hash_str}
1029  \__pdfmeta_xmp_xmlns_new:nn {pdfaid}   {http://www.aiim.org/pdfa/ns/id/}
1030  \__pdfmeta_xmp_xmlns_new:nn {pdfuaid}  {http://www.aiim.org/pdfua/ns/id/}
1031  \__pdfmeta_xmp_xmlns_new:nn {pdfx}     {http://ns.adobe.com/pdfx/1.3/}
1032  %\__pdfmeta_xmp_xmlns_new:nn {pdfxid}   {http://www.npes.org/pdfx/ns/id/}
1033  \__pdfmeta_xmp_xmlns_new:nn {prism}    {http://prismstandard.org/namespaces/basic/3.0/}
1034  %\__pdfmeta_xmp_xmlns_new:nn {jav}      {http://www.niso.org/schemas/jav/1.0/}
1035  %\__pdfmeta_xmp_xmlns_new:nn {xmpTPg}   {http://ns.adobe.com/xap/1.0/t/pg/}
1036  \__pdfmeta_xmp_xmlns_new:nn {stFnt}    {http://ns.adobe.com/xap/1.0/sType/Font\c_hash_str}
1037  \__pdfmeta_xmp_xmlns_new:nn {Iptc4xmpCore}{http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
```

```
1038 \__pdfmeta_xmp_xmlns_new:nn {pdfaExtension}{http://www.aiim.org/pdfa/ns/extension/}
1039 \__pdfmeta_xmp_xmlns_new:nn {pdfaSchema}{http://www.aiim.org/pdfa/ns/schema\c_hash_str}
1040 \__pdfmeta_xmp_xmlns_new:nn {pdfaProperty}{http://www.aiim.org/pdfa/ns/property\c_hash_str}
1041 \__pdfmeta_xmp_xmlns_new:nn {pdfaType} {http://www.aiim.org/pdfa/ns/type\c_hash_str}
1042 \__pdfmeta_xmp_xmlns_new:nn {pdfaField}{http://www.aiim.org/pdfa/ns/field\c_hash_str}
```

4.7 Building the chunks: Extensions

In this part local name spaces or additional names in a name space can be declared. A
"schema" declaration consist of the declaration of the name, uri and prefix which then
surrounds a bunch of property declarations. The current code doesn't support all syntax
options but sticks to what is used in hyperxmp and pdfx. If needed it can be extended
later.

\l__pdfmeta_xmp_schema_seq   This variable will hold the list of prefix so that we can loop to produce the final XML

```
1043 \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.

```
1044 \cs_new_protected:Npn \__pdfmeta_xmp_schema_new:nnn #1 #2 #3
1045   %#1 name #2 prefix, #3 text
1046   {
1047     \tl_if_exist:cTF { g__pdfmeta_xmp_schema_#2_tl }
1048       {
1049        \msg_warning:nnnn{pdfmeta}{xmp-defined}{schema}{#2}
1050       }
1051       {
1052         \seq_put_right:Nn \l__pdfmeta_xmp_schema_seq { #2 }
1053         \tl_new:c { g__pdfmeta_xmp_schema_#2_tl }
1054         \tl_new:c { g__pdfmeta_xmp_schema_#2_properties_tl }
1055         \tl_gput_right:cn { g__pdfmeta_xmp_schema_#2_tl }
1056           {
1057             \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1058              \__pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{schema}{#1}
1059              \__pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{prefix}{#2}
1060              \__pdfmeta_xmp_add_packet_line:nnn {pdfaSchema}{namespaceURI}{#3}
1061              \__pdfmeta_xmp_add_packet_open:nn {pdfaSchema}{property}
1062              \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1063                  \tl_use:c { g__pdfmeta_xmp_schema_#2_properties_tl }
1064              \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1065             \__pdfmeta_xmp_add_packet_close:nn {pdfaSchema}{property}
1066            \cs_if_exist_use:c {__pdfmeta_xmp_schema_#2_additions:}
1067            \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1068           }
1069       }
1070   }
```

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

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

```
1071 \prop_new:N\g__pdfmeta_xmp_schema_property_prop
1072 \cs_new_protected:Npn \__pdfmeta_xmp_property_new:nnnnn #1 #2 #3 #4 #5 %
1073     %#1 schema #2 name, #3 type, #4 category #5 description
1074   {
1075     \tl_if_exist:cTF { g__pdfmeta_xmp_schema_#1_properties_tl }
1076       {
1077         \prop_get:NeNF \g__pdfmeta_xmp_schema_property_prop {#1:#2}\l__pdfmeta_tmpa_tl
1078         {
1079           \prop_gput:Nee \g__pdfmeta_xmp_schema_property_prop {#1:#2}{#3}
1080           \tl_gput_right:cn { g__pdfmeta_xmp_schema_#1_properties_tl }
1081           {
1082             \__pdfmeta_xmp_add_packet_open:nn {rdf}{li~rdf:parseType="Resource"}
1083               \__pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{name}{#2}
1084               \__pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{valueType}{#3}
1085               \__pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{category}{#4}
1086               \__pdfmeta_xmp_add_packet_line:nnn {pdfaProperty}{description}{#5}
1087             \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1088           }
1089         }
1090       }
1091       {
1092         \msg_warning:nnnn{pdfmeta}{xmp-undefined}{schema}{#1}
1093       }
1094   }
```

(*End of definition for* `\__pdfmeta_xmp_property_new:nnnnn`.)

`\__pdfmeta_xmp_add_packet_field:nnn` This adds a field to a schema.

```
1095 \cs_new_protected:Npn \__pdfmeta_xmp_add_packet_field:nnn #1 #2 #3 %
1096   %#1 name #2 valuetype #3 description
1097   {
1098     \__pdfmeta_xmp_add_packet_open_attr:nnn {rdf}{li}{rdf:parseType="Resource"}
1099         \__pdfmeta_xmp_add_packet_line:nnn {pdfaField}{name}{#1}
1100         \__pdfmeta_xmp_add_packet_line:nnn {pdfaField}{valueType}{#2}
1101         \__pdfmeta_xmp_add_packet_line:nnn {pdfaField}{description}{#3}
1102     \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1103   }
```

(*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-1[7]

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

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

---

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

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.

```
1104 \__pdfmeta_xmp_schema_new:nnn
1105   {XMP~Media~Management~Schema}
1106   {xmpMM}
1107   {http://ns.adobe.com/xap/1.0/mm/}
1108 \__pdfmeta_xmp_property_new:nnnnn
1109   {xmpMM}
1110   {OriginalDocumentID}
1111   {URI}
1112   {internal}
1113   {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)

```
1114 \__pdfmeta_xmp_schema_new:nnn
1115   {PDF/A~Identification~Schema}
1116   {pdfaid}
1117   {http://www.aiim.org/pdfa/ns/id/}
1118 \__pdfmeta_xmp_property_new:nnnnn
1119   {pdfaid}
1120   {year}
1121   {Integer}
1122   {internal}
1123   {Year~of~standard}
1124 \__pdfmeta_xmp_property_new:nnnnn
1125   {pdfaid}
1126   {rev}
1127   {Integer}
1128   {internal}
1129   {Revision~year~of~standard}
```

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

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

pdfuaid~(schema)

```
1130 \__pdfmeta_xmp_schema_new:nnn
1131   {PDF/UA~Universal~Accessibility~Schema}
1132   {pdfuaid}
1133   {http://www.aiim.org/pdfua/ns/id/}
1134 \__pdfmeta_xmp_property_new:nnnnn
1135   {pdfuaid}
1136   {part}
1137   {Integer}
1138   {internal}
```

```
1139      {Part~of~ISO~14289~standard}
1140 \__pdfmeta_xmp_property_new:nnnnn
1141      {pdfuaid}
1142      {rev}
1143      {Integer}
1144      {internal}
1145      {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)`

```
1146 \cs_new_protected:Npn \__pdfmeta_xmp_add_pdfxid:
1147  {
1148   \__pdfmeta_xmp_xmlns_new:nn {pdfxid}    {http://www.npes.org/pdfx/ns/id/}
1149   \__pdfmeta_xmp_schema_new:nnn
1150       {PDF/X~ID~Schema}
1151       {pdfxid}
1152       {http://www.npes.org/pdfx/ns/id/}
1153   \__pdfmeta_xmp_property_new:nnnnn
1154       {pdfxid}
1155       {GTS_PDFXVersion}
1156       {Text}
1157       {internal}
1158       {ID~of~PDF/X~standard}
1159  }
```

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

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

```
1160 \__pdfmeta_xmp_schema_new:nnn
1161    {PRISM~Basic~Metadata}
1162    {prism}
1163    {http://prismstandard.org/namespaces/basic/3.0/}
1164 \__pdfmeta_xmp_property_new:nnnnn
1165    {prism}
1166    {complianceProfile}
1167    {Text}
1168    {internal}
1169    {PRISM~specification~compliance~profile~to~which~this~document~adheres}
1170 \__pdfmeta_xmp_property_new:nnnnn
1171    {prism}
1172    {publicationName}
1173    {Text}
1174    {external}
```

```
1175    {Publication~name}
1176  \__pdfmeta_xmp_property_new:nnnnn
1177    {prism}
1178    {aggregationType}
1179    {Text}
1180    {external}
1181    {Publication~type}
1182  \__pdfmeta_xmp_property_new:nnnnn
1183    {prism}
1184    {bookEdition}
1185    {Text}
1186    {external}
1187    {Edition~of~the~book~in~which~the~document~was~published}
1188  \__pdfmeta_xmp_property_new:nnnnn
1189    {prism}
1190    {volume}
1191    {Text}
1192    {external}
1193    {Publication~volume~number}
1194  \__pdfmeta_xmp_property_new:nnnnn
1195    {prism}
1196    {number}
1197    {Text}
1198    {external}
1199    {Publication~issue~number~within~a~volume}
1200  \__pdfmeta_xmp_property_new:nnnnn
1201    {prism}
1202    {pageRange}
1203    {Text}
1204    {external}
1205    {Page~range~for~the~document~within~the~print~version~of~its~publication}
1206  \__pdfmeta_xmp_property_new:nnnnn
1207    {prism}
1208    {issn}
1209    {Text}
1210    {external}
1211    {ISSN~for~the~printed~publication~in~which~the~document~was~published}
1212  \__pdfmeta_xmp_property_new:nnnnn
1213    {prism}
1214    {eIssn}
1215    {Text}
1216    {external}
1217    {ISSN~for~the~electronic~publication~in~which~the~document~was~published}
1218  \__pdfmeta_xmp_property_new:nnnnn
1219    {prism}
1220    {isbn}
1221    {Text}
1222    {external}
1223    {ISBN~for~the~publication~in~which~the~document~was~published}
1224  \__pdfmeta_xmp_property_new:nnnnn
1225    {prism}
1226    {doi}
1227    {Text}
1228    {external}
```

```
1229    {Digital~Object~Identifier~for~the~document}
1230 \__pdfmeta_xmp_property_new:nnnnn
1231    {prism}
1232    {url}
1233    {URL}
1234    {external}
1235    {URL~at~which~the~document~can~be~found}
1236 \__pdfmeta_xmp_property_new:nnnnn
1237    {prism}
1238    {byteCount}
1239    {Integer}
1240    {internal}
1241    {Approximate~file~size~in~octets}
1242 \__pdfmeta_xmp_property_new:nnnnn
1243    {prism}
1244    {pageCount}
1245    {Integer}
1246    {internal}
1247    {Number~of~pages~in~the~print~version~of~the~document}
1248 \__pdfmeta_xmp_property_new:nnnnn
1249    {prism}
1250    {subtitle}
1251    {Text}
1252    {external}
1253    {Document's~subtitle}
```

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

iptc (schema) **iptc**

```
1254 \__pdfmeta_xmp_schema_new:nnn
1255    {IPTC~Core~Schema}
1256    {Iptc4xmpCore}
1257    {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
1258 \__pdfmeta_xmp_property_new:nnnnn
1259    {Iptc4xmpCore}
1260    {CreatorContactInfo}
1261    {ContactInfo}
1262    {external}
1263    {Document~creator's~contact~information}
1264 \cs_new_protected:cpn { __pdfmeta_xmp_schema_Iptc4xmpCore_additions: }
1265    {
1266      \__pdfmeta_xmp_add_packet_open:nn{pdfaSchema}{valueType}
1267        \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1268          \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1269            \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{ContactInfo}
1270            \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
1271              {http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/}
1272            \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{Iptc4xmpCore}
1273            \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
1274              {Basic~set~of~information~to~get~in~contact~with~a~person}
1275            \__pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
1276              \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1277                \__pdfmeta_xmp_add_packet_field:nnn{CiAdrCity}{Text}
1278                  {Contact~information~city}
```

```
1279              \__pdfmeta_xmp_add_packet_field:nnn{CiAdrCtry}{Text}
1280                {Contact~information~country}
1281              \__pdfmeta_xmp_add_packet_field:nnn{CiAdrExtadr}{Text}
1282                {Contact~information~address}
1283              \__pdfmeta_xmp_add_packet_field:nnn{CiAdrPcode}{Text}
1284                {Contact~information~local~postal~code}
1285              \__pdfmeta_xmp_add_packet_field:nnn{CiAdrRegion}{Text}
1286                {Contact~information~regional~information~such~as~state~or~province}
1287              \__pdfmeta_xmp_add_packet_field:nnn{CiEmailWork}{Text}
1288                {Contact~information~email~address(es)}
1289              \__pdfmeta_xmp_add_packet_field:nnn{CiTelWork}{Text}
1290                {Contact~information~telephone~number(s)}
1291              \__pdfmeta_xmp_add_packet_field:nnn{CiUrlWork}{Text}
1292                {Contact~information~Web~URL(s)}
1293            \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1294          \__pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
1295        \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1296      \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1297    \__pdfmeta_xmp_add_packet_close:nn{pdfaSchema}{valueType}
1298  }
```

(*End of definition for* `iptc` (`schema`)*.*)

jav : currently ignored

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

If declarations are added to the XMP-metadata they need (for pdf/A 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.)

```
1299 \cs_new_protected:Npn \__pdfmeta_xmp_schema_enable_pdfd:
1300  {
1301   \__pdfmeta_xmp_xmlns_new:nn {pdfd}{http://pdfa.org/declarations/}
1302   \__pdfmeta_xmp_schema_new:nnn
1303     {PDF~Declarations~Schema}
1304     {pdfd}
1305     {http://pdfa.org/declarations/}
1306   \__pdfmeta_xmp_property_new:nnnnn
1307     {pdfd}
1308     {declarations}
1309     {Bag~declaration}
1310     {external}
1311     {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.

```
1312    \cs_new_protected:cpn { __pdfmeta_xmp_schema_pdfd_additions: }
1313      {
```

```
1314        \__pdfmeta_xmp_add_packet_open:nn{pdfaSchema}{valueType}
1315          \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1316            \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1317              \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{claim}
1318              \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
1319                {http://pdfa.org/declarations/}
1320              \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{pdfd}
1321              \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
1322                {A~structure~describing~properties~of~an~individual claim.}
1323              \__pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
1324                \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1325                  \__pdfmeta_xmp_add_packet_field:nnn{claimReport}{Text}
1326                    {A~URL~to~a~report~containing~details~of~the~specific~conformance~claim
1327                  \__pdfmeta_xmp_add_packet_field:nnn{claimCredentials}{Text}
1328                    {The~claimant's~credentials.}
1329                  \__pdfmeta_xmp_add_packet_field:nnn{claimDate}{Text}
1330                    {A~date~identifying~when~the~claim~was~made.}
1331                  \__pdfmeta_xmp_add_packet_field:nnn{claimBy}{Text}
1332                    {The~name~of~the~organization~and/or~individual~and/or~software~making~
1333                \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1334              \__pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
1335            \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1336            \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1337              \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{type}{declaration}
1338              \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{namespaceURI}
1339                {http://pdfa.org/declarations/}
1340              \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{prefix}{pdfd}
1341              \__pdfmeta_xmp_add_packet_line:nnn{pdfaType}{description}
1342                {A~structure~describing~a~single~PDF~ Declaration~asserting~conformance~w
     identified~standard~or~ profile.}
1343              \__pdfmeta_xmp_add_packet_open:nn{pdfaType}{field}
1344                \__pdfmeta_xmp_add_packet_open:nn{rdf}{Seq}
1345                  \__pdfmeta_xmp_add_packet_field:nnn{conformsTo}{Text}
1346                    {A~property~containing~a~URI~specifying~the~standard~or~profile~by~the~
1347                  \__pdfmeta_xmp_add_packet_field:nnn{claimData}{Bag~claim}
1348                    {An~unordered~array~of~claim~data,~where~each~claim~identifies~the~natu
1349                \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1350              \__pdfmeta_xmp_add_packet_close:nn{pdfaType}{field}
1351            \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1352          \__pdfmeta_xmp_add_packet_close:nn{rdf}{Seq}
1353        \__pdfmeta_xmp_add_packet_close:nn{pdfaSchema}{valueType}
1354      }
```

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

```
1355      \cs_gset_eq:NN \__pdfmeta_xmp_schema_enable_pdfd: \prg_do_nothing:
1356    }
```

4.8 The actual user / document data

4.8.1 pdf

This builds pdf related the data with the (prefix "pdf").

```
1357 \cs_new_protected:Npn \__pdfmeta_xmp_build_pdf:
1358   {
```

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

```
1359   \__pdfmeta_xmp_add_packet_line_default:nnee
1360     {pdf}{Producer}
1361     {\c_sys_engine_exec_str-\c_sys_engine_version_str}
1362     {\GetDocumentProperties{hyperref/pdfproducer}}
```

Now the PDF version

```
1363   \__pdfmeta_xmp_add_packet_line:nne{pdf}{PDFVersion}{\pdf_version:}
```

```
1364   }
```

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

4.8.2 xmp

This builds the data with the (prefix "xmp").

```
1365 \cs_new_protected:Npn \__pdfmeta_xmp_build_xmp:
1366   {
```

The creator

```
1367   \__pdfmeta_xmp_add_packet_line_default:nnee
1368     {xmp}{CreatorTool}
1369     {LaTeX}
1370     { \GetDocumentProperties{hyperref/pdfcreator} }
```

The baseurl

```
1371   \__pdfmeta_xmp_add_packet_line_default:nnee
1372     {xmp}{BaseURL}{}
1373     { \GetDocumentProperties{hyperref/baseurl} }
```

CreationDate

```
1374     \__pdfmeta_xmp_date_get:nNN
1375       {document/creationdate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1376     \__pdfmeta_xmp_add_packet_line:nne{xmp}{CreateDate}{\__pdfmeta_xmp_print_date:N\l__pdfme
1377     \pdfmanagement_add:nne{Info}{CreationDate}{(\l__pdfmeta_tmpa_tl)}
```

ModifyDate

```
1378     \__pdfmeta_xmp_date_get:nNN
1379       {document/moddate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1380     \__pdfmeta_xmp_add_packet_line:nne{xmp}{ModifyDate}{\__pdfmeta_xmp_print_date:N\l__pdfme
1381     \pdfmanagement_add:nne{Info}{ModDate}{(\l__pdfmeta_tmpa_tl)}
```

MetadataDate

```
1382      \__pdfmeta_xmp_date_get:nNN
1383        {hyperref/pdfmetadate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1384      \__pdfmeta_xmp_add_packet_line:nne{xmp}{MetadataDate}{\__pdfmeta_xmp_print_date:N\l__pdf

1385    }
```

(*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 \DocumentMetadata. The values for A-standards are taken from the property, X and UA are currently taken from the document container, this should be changed when merging of standards are possible.

__pdfmeta_xmp_build_standards:

```
1386  \cs_new_protected:Npn \__pdfmeta_xmp_build_standards:
1387    {
1388      \__pdfmeta_xmp_add_packet_line:nne {pdfaid}{part}{\pdfmeta_standard_item:n{level}}
1389      \__pdfmeta_xmp_add_packet_line:nne
1390        {pdfaid}{conformance}{\pdfmeta_standard_item:n{conformance}}
1391      \int_compare:nNnTF {0\pdfmeta_standard_item:n{level}}<{4}
1392        {\__pdfmeta_xmp_add_packet_line:nne {pdfaid}{year} {\pdfmeta_standard_item:n{year}}}
1393        {\__pdfmeta_xmp_add_packet_line:nne {pdfaid}{rev}  {\pdfmeta_standard_item:n{year}}}
1394      \__pdfmeta_xmp_add_packet_line:nne
1395        {pdfxid}{GTS_PDFXVersion}{\GetDocumentProperties{document/pdfstandard-X}}
1396      \pdfmanagement_get_documentproperties:nNT {document/pdfstandard-UA}\l__pdfmeta_tmpa_tl
1397        {
1398          \__pdfmeta_xmp_add_packet_line:nne
1399            {pdfuaid}{part}{\exp_last_unbraced:No\use_i:nn \l__pdfmeta_tmpa_tl}
1400          \__pdfmeta_xmp_add_packet_line:nne
1401            {pdfuaid}{rev}{\exp_last_unbraced:No\use_ii:nn \l__pdfmeta_tmpa_tl}
1402        }
1403    }
```

(*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.

```
1404  \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

```
1405 \cs_new_protected:Npn \__pdfmeta_xmp_build_pdfd:
1406   {
1407     \prop_if_empty:NF\g__pdfmeta_xmp_pdfd_data_prop
1408       {
1409         \__pdfmeta_xmp_add_packet_open:nn{pdfd}{declarations}
1410         \__pdfmeta_xmp_add_packet_open:nn{rdf}{Bag}
1411         \prop_map_inline:Nn \g__pdfmeta_xmp_pdfd_data_prop
1412           {
1413             \__pdfmeta_xmp_build_pdfd_claim:nn{##1}{##2}
1414           }
1415         \__pdfmeta_xmp_add_packet_close:nn{rdf}{Bag}
1416         \__pdfmeta_xmp_add_packet_close:nn{pdfd}{declarations}
1417       }
1418   }
```

(*End of definition for* __pdfmeta_xmp_build_pdfd:.)

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

```
1419 \cs_new_protected:Npn \__pdfmeta_xmp_build_pdfd_claim:nn #1#2
1420   {
1421     \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1422       \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{conformsTo}{#1}
1423       \tl_if_empty:nF {#2}
1424         {
1425           \__pdfmeta_xmp_add_packet_open:nn{pdfd}{claimData}
1426           \__pdfmeta_xmp_add_packet_open:nn{rdf}{Bag}
1427             #2
1428           \__pdfmeta_xmp_add_packet_close:nn{rdf}{Bag}
1429           \__pdfmeta_xmp_add_packet_close:nn{pdfd}{claimData}
1430         }
1431     \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1432   }
```

(*End of definition for* __pdfmeta_xmp_build_pdfd_claim:nn.)

4.10 Photoshop

```
1433 \cs_new_protected:Npn \__pdfmeta_xmp_build_photoshop:
1434   {
```

pdfauthortitle/photoshop:AuthorsPosition

```
1435     \__pdfmeta_xmp_add_packet_line:nne{photoshop}{AuthorsPosition}
1436       { \GetDocumentProperties{hyperref/pdfauthortitle} }
```

pdfcaptionwriter/photoshop:CaptionWriter

```
1437     \__pdfmeta_xmp_add_packet_line:nne{photoshop}{CaptionWriter}
1438       { \GetDocumentProperties{hyperref/pdfcaptionwriter} }
```

```
1439    }
```

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

4.11 XMP Media Management

`\__pdfmeta_xmp_build_xmpMM:`

```
1440 \cs_new_protected:Npn \__pdfmeta_xmp_build_xmpMM:
1441    {
```

pdfdocumentid / xmpMM:DocumentID

```
1442      \str_set:Ne\l__pdfmeta_tmpa_str {\GetDocumentProperties{hyperref/pdfdocumentid}}
1443      \str_if_empty:NT \l__pdfmeta_tmpa_str
1444        {
1445          \__pdfmeta_xmp_create_uuid:nN
1446            {\jobname\GetDocumentProperties{hyperref/pdftitle}}
1447            \l__pdfmeta_tmpa_str
1448        }
1449      \__pdfmeta_xmp_add_packet_line:nnV{xmpMM}{DocumentID}
1450        \l__pdfmeta_tmpa_str
```

pdfinstanceid / xmpMM:InstanceID

```
1451      \str_set:Ne\l__pdfmeta_tmpa_str {\GetDocumentProperties{hyperref/pdfinstanceid}}
1452      \str_if_empty:NT \l__pdfmeta_tmpa_str
1453        {
1454          \__pdfmeta_xmp_create_uuid:nN
1455            {\jobname\l__pdfmeta_xmp_currentdate_tl}
1456            \l__pdfmeta_tmpa_str
1457        }
1458      \__pdfmeta_xmp_add_packet_line:nnV{xmpMM}{InstanceID}
1459        \l__pdfmeta_tmpa_str
```

pdfversionid/xmpMM:VersionID

```
1460      \__pdfmeta_xmp_add_packet_line:nne{xmpMM}{VersionID}
1461        { \GetDocumentProperties{hyperref/pdfversionid} }
```

pdfrendition/xmpMM:RenditionClass

```
1462      \__pdfmeta_xmp_add_packet_line:nne{xmpMM}{RenditionClass}
1463        { \GetDocumentProperties{hyperref/pdfrendition} }
1464    }
```

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

4.12 Rest of dublin Core data

```
1465 \cs_new_protected:Npn \__pdfmeta_xmp_build_dc:
1466   {
```

pdfauthor/dc:creator

```
1467     \__pdfmeta_xmp_add_packet_list:nnne {dc}{creator}{Seq}
1468       { \GetDocumentProperties{hyperref/pdfauthor} }
1469     \int_compare:nNnT {0\pdfmeta_standard_item:n{level}}={1}
1470       { \pdfmanagement_remove:nn{Info}{Author} }
```

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

```
1471     \__pdfmeta_xmp_add_packet_list:nnne {dc}{title}{Alt}
1472       { \GetDocumentProperties{hyperref/pdftitle} }
```

pdfkeywords/dc:subject

```
1473     \__pdfmeta_xmp_add_packet_list:nnne {dc}{subject}{Bag}
1474       { \GetDocumentProperties{hyperref/pdfkeywords} }
1475     \int_compare:nNnT {0\pdfmeta_standard_item:n{level}}={1}
1476       { \pdfmanagement_remove:nn{Info}{Keywords} }
```

pdftype/dc:type

```
1477     \pdfmanagement_get_documentproperties:nNTF { hyperref/pdftype } \l__pdfmeta_tmpa_tl
1478       {
1479         \__pdfmeta_xmp_add_packet_list_simple:nnnV {dc}{type}{Bag}\l__pdfmeta_tmpa_tl
1480       }
1481       {
1482         \__pdfmeta_xmp_add_packet_list_simple:nnnn {dc}{type}{Bag}{Text}
1483       }
```

pdfpublisher/dc:publisher

```
1484     \__pdfmeta_xmp_add_packet_list:nnne {dc}{publisher}{Bag}
1485       { \GetDocumentProperties{hyperref/pdfpublisher} }
```

pdfsubject/dc:description

```
1486     \__pdfmeta_xmp_add_packet_list:nnne
1487     {dc}{description}{Alt}
1488     {\GetDocumentProperties{hyperref/pdfsubject}}
```

lang/pdflang/dc:language

```
1489     \__pdfmeta_xmp_add_packet_list_simple:nnnV
1490     {dc}{language}{Bag}\l__pdfmeta_xmp_doclang_tl
```

49

pdfidentifier/dc:identifier

```
1491     \__pdfmeta_xmp_add_packet_line:nne{dc}{identifier}
1492       { \GetDocumentProperties{hyperref/pdfidentifier} }
```

pdfdate/dc:date

```
1493     \__pdfmeta_xmp_date_get:nNN {hyperref/pdfdate}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_seq
1494      \__pdfmeta_xmp_add_packet_list_simple:nnne
1495       {dc}{date}{Seq}{\__pdfmeta_xmp_print_date:N\l__pdfmeta_tmpa_seq}
```

The file format

```
1496     \__pdfmeta_xmp_add_packet_line:nnn{dc}{format}{application/pdf}
```

The source

```
1497     \__pdfmeta_xmp_add_packet_line_default:nnee
1498      {dc}{source}
1499      { \c_sys_jobname_str.tex }
1500      { \GetDocumentProperties{hyperref/pdfsource} }

1501     \__pdfmeta_xmp_add_packet_list:nnne{dc}{rights}{Alt}
1502      {\GetDocumentProperties{hyperref/pdfcopyright}}

1503     }
```

(*End of definition for* __pdfmeta_xmp_build_dc: *and others.*)

4.13   xmpRights

__pdfmeta_xmp_build_xmpRights:

```
1504 \cs_new_protected:Npn \__pdfmeta_xmp_build_xmpRights:
1505   {
1506     \__pdfmeta_xmp_add_packet_line:nne
1507       {xmpRights}
1508       {WebStatement}
1509       {\GetDocumentProperties{hyperref/pdflicenseurl}}
1510     \__pdfmeta_xmp_add_packet_line:nne
1511       {xmpRights}
1512       {Marked}
1513       {
1514        \str_case:en {\GetDocumentProperties{document/copyright}}
1515          {
1516            {true}{True}
1517            {false}{False}
1518          }
1519       }
1520   }
```

(*End of definition for* __pdfmeta_xmp_build_xmpRights:*.*)

50

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

```
1521 \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

```
1522 \cs_new_protected:Npn \__pdfmeta_xmp_build_iptc_data:N #1
1523   {
1524     \tl_clear:N #1
1525     \__pdfmeta_xmp_incr_indent:\__pdfmeta_xmp_incr_indent:\__pdfmeta_xmp_incr_indent:\__pdf
1526     \__pdfmeta_xmp_add_packet_line:nneN
1527       {Iptc4xmpCore}{CiAdrExtadr}
1528       {\GetDocumentProperties{hyperref/pdfcontactaddress}}
1529       #1
1530     \__pdfmeta_xmp_add_packet_line:nneN
1531       {Iptc4xmpCore}{CiAdrCity}
1532       {\GetDocumentProperties{hyperref/pdfcontactcity}}
1533       #1
1534     \__pdfmeta_xmp_add_packet_line:nneN
1535       {Iptc4xmpCore}{CiAdrPcode}
1536       {\GetDocumentProperties{hyperref/pdfcontactpostcode}}
1537       #1
1538     \__pdfmeta_xmp_add_packet_line:nneN
1539       {Iptc4xmpCore}{CiAdrCtry}
1540       {\GetDocumentProperties{hyperref/pdfcontactcountry}}
1541       #1
1542     \__pdfmeta_xmp_add_packet_line:nneN
1543       {Iptc4xmpCore}{CiTelWork}
1544       {\GetDocumentProperties{hyperref/pdfcontactphone}}
1545       #1
1546     \__pdfmeta_xmp_add_packet_line:nneN
1547       {Iptc4xmpCore}{CiEmailWork}
1548       {\GetDocumentProperties{hyperref/pdfcontactemail}}
1549       #1
1550     \__pdfmeta_xmp_add_packet_line:nneN
1551       {Iptc4xmpCore}{CiUrlWork}
1552       {\GetDocumentProperties{hyperref/pdfcontacturl}}
1553       #1
1554     \__pdfmeta_xmp_decr_indent:\__pdfmeta_xmp_decr_indent:\__pdfmeta_xmp_decr_indent:\__pdf
1555   }
```

(*End of definition for* __pdfmeta_xmp_build_iptc_data:N.)

__pdfmeta_xmp_build_iptc:

```
1556 \cs_new_protected:Npn \__pdfmeta_xmp_build_iptc:
1557   {
1558     \tl_if_empty:NF\l__pdfmeta_xmp_iptc_data_tl
```

```
1559        {
1560          \__pdfmeta_xmp_add_packet_open_attr:nnn
1561            {Iptc4xmpCore}{CreatorContactInfo}{rdf:parseType="Resource"}
1562          \tl_gput_right:Ne\g__pdfmeta_xmp_packet_tl { \l__pdfmeta_xmp_iptc_data_tl }
1563          \__pdfmeta_xmp_add_packet_close:nn
1564            {Iptc4xmpCore}{CreatorContactInfo}
1565        }
1566     }
```

(*End of definition for* __pdfmeta_xmp_build_iptc:.)

4.15   Prism

__pdfmeta_xmp_build_prism:
          complianceProfile
 prism:subtitle/pdfsubtitle

```
1567 \cs_new_protected:Npn \__pdfmeta_xmp_build_prism:
1568     {
```

The compliance profile is a fix value taken from hyperxmp

```
1569        \__pdfmeta_xmp_add_packet_line:nnn
1570          {prism}{complianceProfile}
1571          {three}
```

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

```
1572        \__pdfmeta_xmp_lang_get:eNN
1573          {\GetDocumentProperties{hyperref/pdfsubtitle}}
1574          \l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl
1575        \__pdfmeta_xmp_add_packet_line_attr:nneV
1576          {prism}{subtitle}
1577          {xml:lang="\l__pdfmeta_tmpa_tl"}
1578          \l__pdfmeta_tmpb_tl
```

Then publicationName

```
1579        \__pdfmeta_xmp_lang_get:eNN
1580          {\GetDocumentProperties{hyperref/pdfpublication}}
1581          \l__pdfmeta_tmpa_tl\l__pdfmeta_tmpb_tl
1582        \__pdfmeta_xmp_add_packet_line_attr:nneV
1583          {prism}{publicationName}
1584          {xml:lang="\l__pdfmeta_tmpa_tl"}
1585          \l__pdfmeta_tmpb_tl
```

Now the rest

```
1586        \__pdfmeta_xmp_add_packet_line:nne
1587          {prism}{bookEdition}
1588          {\GetDocumentProperties{hyperref/pdfbookedition}}
1589        \__pdfmeta_xmp_add_packet_line:nne
1590          {prism}{aggregationType}
1591          {\GetDocumentProperties{hyperref/pdfpubtype}}
1592        \__pdfmeta_xmp_add_packet_line:nne
1593          {prism}{volume}
```

```
1594        {\GetDocumentProperties{hyperref/pdfvolumenum}}
1595     \__pdfmeta_xmp_add_packet_line:nne
1596        {prism}{number}
1597        {\GetDocumentProperties{hyperref/pdfissuenum}}
1598     \__pdfmeta_xmp_add_packet_line:nne
1599        {prism}{pageRange}
1600        {\GetDocumentProperties{hyperref/pdfpagerange}}
1601     \__pdfmeta_xmp_add_packet_line:nne
1602        {prism}{issn}
1603        {\GetDocumentProperties{hyperref/pdfissn}}
1604     \__pdfmeta_xmp_add_packet_line:nne
1605        {prism}{eIssn}
1606        {\GetDocumentProperties{hyperref/pdfeissn}}
1607     \__pdfmeta_xmp_add_packet_line:nne
1608        {prism}{doi}
1609        {\GetDocumentProperties{hyperref/pdfdoi}}
1610     \__pdfmeta_xmp_add_packet_line:nne
1611        {prism}{url}
1612        {\GetDocumentProperties{hyperref/pdfurl}}
```

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

```
1613        \tl_set:Ne \l__pdfmeta_tmpa_tl { \GetDocumentProperties{hyperref/pdfnumpages} }
1614     \__pdfmeta_xmp_add_packet_line:nne
1615        {prism}{pageCount}
1616        {\tl_if_blank:VTF \l__pdfmeta_tmpa_tl {\PreviousTotalPages}{\l__pdfmeta_tmpa_tl}}

1617   }
```

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

4.15.1 User additions

\g__pdfmeta_xmp_user_packet_str

```
1618 \tl_new:N \g__pdfmeta_xmp_user_packet_tl
```

(*End of definition for* \g__pdfmeta_xmp_user_packet_str.)

__pdfmeta_xmp_build_user:

```
1619 \cs_new_protected:Npn \__pdfmeta_xmp_build_user:
1620  {
1621    \int_zero:N \l__pdfmeta_xmp_indent_int
1622    \g__pdfmeta_xmp_user_packet_tl
1623    \int_set:Nn \l__pdfmeta_xmp_indent_int {3}
1624  }
```

(*End of definition for* __pdfmeta_xmp_build_user:.)

4.16 Activating the metadata

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

```
1625 \hook_new:n { pdfmeta/xmp }
1626 \AddToHook{shipout/lastpage}
1627   {
1628     \bool_if:NT\g__pdfmeta_xmp_bool
1629       {
1630         \str_if_exist:NTF\c_sys_timestamp_str
1631           {
1632             \tl_set_eq:NN \l__pdfmeta_xmp_currentdate_tl \c_sys_timestamp_str
1633           }
1634           {
1635             \file_get_timestamp:nN{\jobname.log}\l__pdfmeta_xmp_currentdate_tl
1636           }
1637         \__pdfmeta_xmp_date_split:VN\l__pdfmeta_xmp_currentdate_tl\l__pdfmeta_xmp_currentdate
1638         \hook_use:n { pdfmeta/xmp }
1639         \__pdfmeta_xmp_build_packet:
1640         \exp_args:No
1641         \__pdf_backend_metadata_stream:n {\g__pdfmeta_xmp_packet_tl}
1642          \pdfmanagement_add:nne {Catalog} {Metadata}{\pdf_object_ref_last:}
1643         \bool_if:NT \g__pdfmeta_xmp_export_bool
1644           {
1645             \iow_open:Nn\g_tmpa_iow{\g__pdfmeta_xmp_export_str.xmpi}
1646             \exp_args:NNo\iow_now:Nn\g_tmpa_iow{\g__pdfmeta_xmp_packet_tl}
1647             \iow_close:N\g_tmpa_iow
1648           }
1649       }
1650   }
```

4.17 User commands

\pdfmeta_xmp_add:n

```
1651 \cs_new_protected:Npn \pdfmeta_xmp_add:n #1
1652   {
1653     \tl_gput_right:Nn \g__pdfmeta_xmp_user_packet_tl
1654       {
1655         \__pdfmeta_xmp_add_packet_chunk:n { #1 }
1656       }
1657   }
```

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

\pdfmeta_xmp_xmlns_new:nn

```
1658 \cs_new_protected:Npn \pdfmeta_xmp_xmlns_new:nn #1 #2
1659   {
1660     \prop_if_in:NnTF \g__pdfmeta_xmp_xmlns_prop {#1}
1661       {\msg_warning:nnnn{pdfmeta}{xmp-defined}{xmlns~namespace}{#1}}
1662       {\__pdfmeta_xmp_xmlns_new:nn {#1}{#2}}
1663   }
```

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

`\pdfmeta_xmp_schema_new:nnn`

```
1664 \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 10.)*

`\pdfmeta_xmp_property_new:nnnnn`

```
1665 \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 10.)*

`\pdfmeta_xmp_add_declaration:n`
`\pdfmeta_xmp_add_declaration:e`

```
1666 \cs_new_protected:Npn \pdfmeta_xmp_add_declaration:n #1 %conformsTo uri
1667   {
1668     \__pdfmeta_xmp_schema_enable_pdfd:
1669     \prop_gput:Nnn\g__pdfmeta_xmp_pdfd_data_prop{#1}{}
1670   }
1671 \cs_generate_variant:Nn \pdfmeta_xmp_add_declaration:n {e}
```

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

`\pdfmeta_xmp_add_declaration:nnnnn`
`\pdfmeta_xmp_add_declaration:ennnn`

```
1672 \cs_new_protected:Npn \pdfmeta_xmp_add_declaration:nnnnn #1#2#3#4#5
1673 %#1=conformsTo uri, #2 claimBy, #3 claimDate #4 claimCredentials #4 claimReport
1674   {
1675     \__pdfmeta_xmp_schema_enable_pdfd:
1676     \tl_set:Nn \l__pdfmeta_tmpa_tl
1677       {
1678         \__pdfmeta_xmp_add_packet_open_attr:nnn{rdf}{li}{rdf:parseType="Resource"}
1679         \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimBy}{#2}
1680         \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimDate}{#3}
1681         \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimCredentials}{#4}
1682         \__pdfmeta_xmp_add_packet_line:nnn{pdfd}{claimReport}{#5}
1683         \__pdfmeta_xmp_add_packet_close:nn{rdf}{li}
1684       }
1685     \prop_get:NnNT \g__pdfmeta_xmp_pdfd_data_prop {#1}\l__pdfmeta_tmpb_tl
1686       {
1687         \tl_concat:NNN \l__pdfmeta_tmpa_tl \l__pdfmeta_tmpa_tl \l__pdfmeta_tmpb_tl
1688       }
1689     \prop_gput:Nno\g__pdfmeta_xmp_pdfd_data_prop{#1}
1690         {
1691           \l__pdfmeta_tmpa_tl
1692         }
1693   }
1694 \cs_generate_variant:Nn\pdfmeta_xmp_add_declaration:nnnnn {e,eee}
```

(End of definition for `\pdfmeta_xmp_add_declaration:nnnnn`*. This function is documented on page 9.)*

4.18 Default declarations

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

```
1695 \cs_new:Npn \__pdfmeta_xmp_iso_today:
1696   {
1697     \int_use:N\c_sys_year_int-
1698     \int_compare:nNnT {\c_sys_month_int} < {10}{0} \int_use:N\c_sys_month_int -
1699     \int_compare:nNnT {\c_sys_day_int}   < {10}{0} \int_use:N\c_sys_day_int
1700   }
1701 \cs_new_protected:Npn \__pdfmeta_xmp_wtpdf_reuse_declaration:
1702  {
1703    \pdfmeta_xmp_add_declaration:eeenn
1704        {http://pdfa.org/declarations/wtpdf\c_hash_str reuse1.0}
1705        {LaTeX~Project}
1706        {\__pdfmeta_xmp_iso_today:}{}{}
1707  }
1708 \cs_new_protected:Npn \__pdfmeta_xmp_wtpdf_accessibility_declaration:
1709  {
1710    \pdfmeta_xmp_add_declaration:ennnn
1711        {http://pdfa.org/declarations/wtpdf\c_hash_str accessibility1.0}
1712        {LaTeX~Project}
1713        {\__pdfmeta_xmp_iso_today:}{}{}
1714  }
```

(*End of definition for* __pdfmeta_xmp_wtpdf_reuse_declaration: *and* __pdfmeta_xmp_wtpdf_-
accessibility_declaration:.)

```
1715 ⟨/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.

58

60