The l3pdfmeta package
pdf-standards and XMP-metadata

The LaTeX3 Project*

Released XXXX-XX-XX

1 l3pdfmeta documentation

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

In future is will probably also contain code to setup XMP-metadata. Until then XMP-metadata can be added by one of two mutual incompatible packages: hyperxmp and pdfx. Both packages are still incompatible with the PDF resource management, but for hyperxmp some patches are provided, so the basic functions works.

1.1 Verifying requirements of PDF standards

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

The l3pdfmeta packages 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

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

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

small number of requirements it is also needed to test a user value against a standard value. For example, `named_actions` restricts the allowed named actions in an annotation of subtype `/Named`, in this case it is needed to check not only if the requirement is in the standard but also if the user value is in the allowed list.

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

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

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

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

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

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

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

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

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

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

1.1.1 Simple tests without handler

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

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

`no_encryption` don't encrypt

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

`no_embed_content` no `/EF` key in filespec, no `/Type/EmbeddedFiles`. *This will be checked in future by l3pdffiles for the files it embeds.* The restrictment is set for only PDF/A-1b. PDF/A-2b and PDF/A3-b lifted this restriction: PDF/A-2b allows to embed other PDF documents conforming to either PDF/A-1 or PDF/A-2, and PDF/A-3 allows any embedded files. I don't see a way to test the PDF/A-2b requirement so currently it will simply allow everything. Perhaps a test for at least the PDF-format will be added in future.

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

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

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

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

1.1.2 Tests with values and special handlers

`min_pdf_version` stores the minimal PDF version. It should be checked against the current PDF version (`\pdf_version:`). A failure means that the version should be changed. This check is done by l3pdfmeta when the version is set with `\DeclareDocumentMetadata` so 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.

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

The interface looks like this

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

  }
```

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

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

```
\DeclareDocumentMetadata
  {
    %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.

2 l3pdfmeta implementation

```
⟨*package⟩
⟨@@=pdfmeta⟩
\ProvidesExplPackage {l3pdfmeta} {2021-02-19} {0.3}
  {XMP-Metadata and PDF-Standards}
```

Message for unknown standards

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

\l__pdfmeta_tmpa_tl
\l__pdfmeta_tmpb_tl
\l__pdfmeta_tmpa_str

```
\tl_new:N\l__pdfmeta_tmpa_tl
\tl_new:N\l__pdfmeta_tmpb_tl
\str_new:N \l__pdfmeta_tmpa_str
```

[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.

(*End definition for* `\l__pdfmeta_tmpa_tl`, `\l__pdfmeta_tmpb_tl`, *and* `\l__pdfmeta_tmpa_str`.)

3 Standards (work in progress)

3.1 Tools and tests

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

`\g__pdfmeta_standard_prop`

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

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

3.2 Functions to check a requirement

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

`\pdfmeta_standard_item:n`

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

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

`\pdfmeta_standard_get:nN`

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

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

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

`\pdfmeta_standard_verify_p:n`
`\pdfmeta_standard_verify:nTF`

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

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

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

`\pdfmeta_standard_verify:nnTF`

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

```
\prg_new_protected_conditional:Npnn \pdfmeta_standard_verify:nn #1 #2  {T,F,TF}
  {
    \prop_if_in:NnTF \g__pdfmeta_standard_prop {#1}
      {
        \cs_if_exist:cTF {__pdfmeta_standard_verify_handler_#1:nn}
```

```
          {
            \exp_args:Nnnx
            \use:c
              {__pdfmeta_standard_verify_handler_#1:nn}
              { #2 }
              { \prop_item:Nn \g__pdfmeta_standard_prop {#1} }
          }
          {
            \prg_return_false:
          }
      }
      {
        \prg_return_true:
      }
  }
```

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

Now we setup a number of handlers.

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

_standard_verify_handler_min_pdf_version:nn

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

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

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

ta_standard_verify_handler_named_actions:nn

```

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

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

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

a_standard_verify_handler_annot_action_A:nn

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

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

This check is probably not needed, but for completeness

dard_verify_handler_outputintent_subtype:nn

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

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

3.3 Enforcing requirements

A number of requirements can sensibly be enforced by us.

3.3.1 Annot flags

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

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

At begin document this should be checked:

```
\hook_gput_code:nnn {begindocument} {pdf}
  {
    \pdfmeta_standard_verify:nF { annot_flags }
     { \__pdfmeta_verify_pdfa_annot_flags: }
  }
```

3.4 pdf/A

We use global properties so that follow up standards can be copied and then adjusted. Some note about requirements for more standard can be found in info/pdfstandard.tex.

\g__pdfmeta_standard_pdf/A-1B_prop
\g__pdfmeta_standard_pdf/A-2B_prop
\g__pdfmeta_standard_pdf/A-3B_prop

```
\prop_new:c { g__pdfmeta_standard_pdf/A-1B_prop }
\prop_set_from_keyval:cn { g__pdfmeta_standard_pdf/A-1B_prop }
  {
    ,name             = pdf/A-1B
    ,type             = A
    ,year             = 2005
    ,min_pdf_version  = 1.4        %minimum
```

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

%A-2b ==============
\prop_new:c { g__pdfmeta_standard_pdf/A-2B_prop }
\prop_gset_eq:cc
  { g__pdfmeta_standard_pdf/A-2B_prop }
  { g__pdfmeta_standard_pdf/A-1B_prop }
\prop_gput:cnn
```

```
  { g__pdfmeta_standard_pdf/A-2B_prop }{name}{pdf/A-2B}
\prop_gput:cnn
  { g__pdfmeta_standard_pdf/A-2B_prop }{year}{2011}
% embedding files is allowed (with restrictions)
\prop_gremove:cn
  { g__pdfmeta_standard_pdf/A-2B_prop }
  { embed_content}

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

(*End definition for* `\g__pdfmeta_standard_pdf/A-1B_prop`, `\g__pdfmeta_standard_pdf/A-2B_prop`, *and* `\g__pdfmeta_standard_pdf/A-3B_prop`.)

3.5 Colorprofiles and Outputintents

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

Adding a profile and an intent is technically easy:

1. Embed the profile as stream with

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

2. Write a `/OutputIntent` dictionary for this

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

3. Reference the dictionary in the catalog:

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

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

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

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

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

Some keys to fill the property.

```
\keys_define:nn { document / metadata }
 {
   colorprofiles .code:n =
    {
      \keys_set:nn { document / metadata / colorprofiles }{#1}
    }
 }
\keys_define:nn { document / metadata / colorprofiles }
 {
   ,A .code:n =
      {
        \tl_if_blank:nF {#1}
          {
            \prop_gput:Nnn \g__pdfmeta_outputintents_prop
             { GTS_PDFA1  } {#1}
          }
      }
   ,a .code:n =
      {
        \tl_if_blank:nF {#1}
          {
            \prop_gput:Nnn \g__pdfmeta_outputintents_prop
              { GTS_PDFA1  } {#1}
          }
      }
   ,X .code:n =
      {
        \tl_if_blank:nF {#1}
          {
             \prop_gput:Nnn \g__pdfmeta_outputintents_prop
              { GTS_PDFX  } {#1}
          }
      }
   ,x .code:n =
      {
        \tl_if_blank:nF {#1}
          {
            \prop_gput:Nnn \g__pdfmeta_outputintents_prop
              { GTS_PDFX  } {#1}
          }
      }
   ,unknown .code:n =
     {
       \tl_if_blank:nF {#1}
          {
           \exp_args:NNo
            \prop_gput:Nnn \g__pdfmeta_outputintents_prop
              { \l_keys_key_str  } {#1}
```

```
          }
     }
 }
```

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

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

__pdfmeta_embed_colorprofile:n
__pdfmeta_write_outputintent:nn

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

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

\cs_new_protected:Npn \__pdfmeta_write_outputintent:nn #1 #2 %#1 file name, #2 subtype
  {
    \group_begin:
     \pdfdict_put:nnx {l_pdfmeta/outputintent}{S}{/\str_convert_pdfname:n{#2}}
     \pdfdict_put:nnx {l_pdfmeta/outputintent}
       {DestOutputProfile}
       {\pdf_object_ref:n{__pdfmeta_colorprofile_#1}}
     \clist_map_inline:nn { OutputConditionIdentifier, Info, RegistryName }
       {
```

```
          \prop_get:cnNT
           { c__pdfmeta_colorprofile_#1}
           { ##1 }
           \l__pdfmeta_tmpa_tl
           {
             \pdf_string_from_unicode:nVN {utf8/string}\l__pdfmeta_tmpa_tl\l__pdfmeta_tmpa_str
             \pdfdict_put:nnx
               {l_pdfmeta/outputintent}{##1}{\l__pdfmeta_tmpa_str}
           }
       }
     \pdf_object_unnamed_write:nx {dict}{\pdfdict_use:n {l_pdfmeta/outputintent} }
     \pdfmanagement_add:nnx {Catalog}{OutputIntents}{\pdf_object_ref_last:}
    \group_end:
  }
```

(*End definition for* `\__pdfmeta_embed_colorprofile:n` *and* `\__pdfmeta_write_outputintent:nn`.)

Now the verifying code. If no requirement is set we simply loop over the property

```

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

If an output intent is required for pdf/A we need to ensure, that the key of default subtype has a value, as default we take sRGB.icc. Then we loop but take always the same profile.

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

```
316             {
317               \exp_args:NV
318               \__pdfmeta_write_outputintent:nn
319                 \l__pdfmeta_tmpb_tl
320                 { #1 }
321             }
322         }
323     }
```

324 ⟨/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.