The l3backend-testphase package
Additional backend PDF features
LaTeX PDF management testphase bundle

The LaTeX Project[*]

Version 0.95u, released 2023-02-07

1 l3backend-testphase Implementation

```
⟨drivers⟩\ProvidesExplFile
⟨*dvipdfmx⟩
  {l3backend-testphase-dvipdfmx.def}{2023-02-07}{}
  {LaTeX~PDF~management~testphase~bundle~backend~support: dvipdfmx}
⟨/dvipdfmx⟩
⟨*dvips⟩
  {l3backend-testphase-dvips.def}{2023-02-07}{}
  {LaTeX~PDF~management~testphase~bundle~backend~support: dvips}
⟨/dvips⟩
⟨*dvisvgm⟩
  {l3backend-testphase-dvisvgm.def}{2023-02-07}{}
  {LaTeX~PDF~management~testphase~bundle~backend~support: dvisvgm}
⟨/dvisvgm⟩
⟨*luatex⟩
  {l3backend-testphase-luatex.def}{2023-02-07}{}
  {LaTeX~PDF~management~testphase~bundle~backend~support: PDF output (LuaTeX)}
⟨/luatex⟩
⟨*pdftex⟩
  {l3backend-testphase-pdftex.def}{2023-02-07}{}
  {LaTeX~PDF~management~testphase~bundle~backend~support: PDF output (pdfTeX)}
⟨/pdftex⟩
⟨*xdvipdfmx⟩
  {l3backend-testphase-xetex.def}{2023-02-07}{}
  {LaTeX~PDF~management~testphase~bundle~backend~support: XeTeX}
⟨/xdvipdfmx⟩
```

1.1 Crossreferences

This uses the temporary l3ref-tmp.sty. It will will be replaced by kernel code later. It is only needed to get a reference for the absolute page counter. This uses the counter from the new lthooks/ltshipout package.

```
⟨@@=pdf⟩
⟨*drivers⟩
```

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

```
\RequirePackage{l3ref-tmp}
\cs_generate_variant:Nn \ref_label:nn {en}
\cs_generate_variant:Nn \ref_value:nn {en}
\cs_new_protected:Npn \__pdf_backend_ref_label:nn #1 #2
  {
     \@bsphack
     \ref_label:nn{#1}{abspage}
     \@esphack
  }
\cs_new:Npn \__pdf_backend_ref_value:nn #1 #2
  {
     \ref_value:nn{#1}{#2}
  }
\cs_generate_variant:Nn \__pdf_backend_ref_label:nn {en}
\cs_generate_variant:Nn \__pdf_backend_ref_value:nn {en}
⟨/drivers⟩
```

avoid that destinations names are optimized with xelatex/dvipdfmx see https://tug.org/pipermail/dvipdf May/000002.html

```
⟨*dvipdfmx | xdvipdfmx⟩
    \__kernel_backend_literal:x { dvipdfmx:config~C~ 0x0010 }
⟨/dvipdfmx | xdvipdfmx⟩
```

\g__pdf_tmpa_prop
\l__pdf_tmpa_tl
\l__pdf_backend_tmpa_box

Some scratch variables

```
⟨*drivers⟩
\prop_new:N \g__pdf_tmpa_prop
\tl_new:N   \l__pdf_tmpa_tl
\box_new:N  \l__pdf_backend_tmpa_box
\box_new:N  \l__pdf_backend_tmpb_box
⟨/drivers⟩
```

(*End definition for* `\g__pdf_tmpa_prop`*,* `\l__pdf_tmpa_tl`*, and* `\l__pdf_backend_tmpa_box`*.*)

\g__pdf_backend_resourceid_int
\g__pdf_backend_name_int
\g__pdf_backend_page_int

a counter to create labels for the resources, a counter to number properties in bdc marks, a counter for the `\pdfpageref` implementation.

```
⟨*drivers⟩
\int_new:N \g__pdf_backend_resourceid_int
\int_new:N \g__pdf_backend_name_int
\int_new:N \g__pdf_backend_page_int
⟨/drivers⟩
```

(*End definition for* `\g__pdf_backend_resourceid_int`*,* `\g__pdf_backend_name_int`*, and* `\g__pdf_backend_page_int`*.*)

1.2 luacode

Load the lua code.

```
⟨*luatex⟩
    \directlua { require("l3backend-testphase.lua") }
⟨/luatex⟩
```

1.3 Converting unicode strings to a pdfname

dvips needs a special function here, so we add this as backend function.

```
⟨*pdftex | luatex | dvipdfmx | xdvipdfmx | dvisvgm⟩
\cs_new:Npn \__kernel_pdf_name_from_unicode_e:n #1
  {
    / \str_convert_pdfname:e { \text_expand:n { #1 } }
  }
⟨/pdftex | luatex | dvipdfmx | xdvipdfmx | dvisvgm⟩
⟨*dvips⟩
\cs_new:Npn \__kernel_pdf_name_from_unicode_e:n #1
  {
    ~ ( \text_expand:n { #1 } ) ~ cvn
  }
⟨/dvips⟩
```

1.4 Hooks

1.4.1 Add the “end run” hooks

Here we add the end run hook to suitable end hooks.

```
⟨*pdftex | luatex⟩
% put in \@kernel@after@enddocument@afterlastpage
\tl_gput_right:Nn \@kernel@after@enddocument@afterlastpage
  {
    \g__kernel_pdfmanagement_end_run_code_tl
  }
⟨/pdftex | luatex⟩
⟨*dvipdfmx | xdvipdfmx⟩
% put in \@kernel@after@shipout@lastpage
\tl_gput_right:Nn \@kernel@after@shipout@lastpage
  {
    \g__kernel_pdfmanagement_end_run_code_tl
  }
⟨/dvipdfmx | xdvipdfmx⟩
⟨*dvips⟩
% put in \@kernel@after@shipout@lastpage
\tl_gput_right:Nn\@kernel@after@shipout@lastpage
  {
    \g__kernel_pdfmanagement_end_run_code_tl
  }
⟨/dvips⟩
```

1.4.2 Add the “shipout” hooks

Now we add to the shipout hooks the relevant token lists. We also push the page resources in shipout/firstpage (AtBeginDvi) as the backend code sets color stack there. The xetex driver needs a rule here. If it clashes on the first page, we will need a test ...

```
⟨*drivers⟩
\tl_if_exist:NTF \@kernel@after@shipout@background
  {
    \g@addto@macro \@kernel@before@shipout@background{\relax}
    \g@addto@macro \@kernel@after@shipout@background
```

```
      {
        \g__kernel_pdfmanagement_thispage_shipout_code_tl
      }
  }
  {
    \hook_gput_code:nnn{shipout/background}{pdf}
      {
        \g__kernel_pdfmanagement_thispage_shipout_code_tl
      }
  }

⟨/drivers⟩
```

1.5 The /Pages dictionary (pdfpagesattr)

__pdf_backend_Pages_primitive:n

This is the primitive command to add something to the /Pages dictionary. It works differently for the backends: pdftex and luatex overwrite existing content, dvips and dvipdfmx are additive. luatex sets it in lua. The higher level code has to take this into account.

```
⟨*pdftex⟩
\cs_new_protected:Npn \__pdf_backend_Pages_primitive:n #1
  {
    \tex_global:D \tex_pdfpagesattr:D { #1 }
  }
⟨/pdftex⟩
⟨*luatex⟩
%luatex: does it in lua
\sys_if_engine_luatex:T
  {
    \cs_new_protected:Npn \__pdf_backend_Pages_primitive:n #1
      {
        \tex_directlua:D
          {
            pdf.setpagesattributes( \__pdf_backend_luastring:n { #1 } )
          }
      }
  }
⟨/luatex⟩
⟨*dvips⟩
\cs_new_protected:Npx \__pdf_backend_Pages_primitive:n #1
  {
    \tex_special:D{ps:~[#1~/PAGES~pdfmark} %]
  }
⟨/dvips⟩
⟨*dvipdfmx | xdvipdfmx⟩
\cs_new_protected:Npn \__pdf_backend_Pages_primitive:n #1
  {
    \__pdf_backend:n{put~@pages~<<#1>>}
  }
⟨/dvipdfmx | xdvipdfmx⟩
⟨*dvisvgm⟩
\cs_new_protected:Npn \__pdf_backend_Pages_primitive:n #1
  {}
```

```
⟨/dvisvgm⟩
```

(*End definition for* `\__pdf_backend_Pages_primitive:n`*.*)

1.6 "Page" and "ThisPage" attributes (pdfpageattr)

__pdf_backend_Page_primitive:n
__pdf_backend_Page_gput:nn
__pdf_backend_Page_gremove:n
__pdf_backend_ThisPage_gput:nn
__pdf_backend_ThisPage_gpush:n

`\__pdf_backend_Page_primitive:n` is the primitive command to add something to the /Page dictionary. It works differently for the backends: pdftex and luatex overwrite existing content, dvips and dvipdfmx are additive. luatex sets it in lua. The higher level code has to take this into account. `\__pdf_backend_Page_gput:nn` stores default values. `\__pdf_backend_Page_gremove:n` allows to remove a value. `\__pdf_backend_ThisPage_gput:nn` adds a value to the current page. `\__pdf_backend_ThisPage_gpush:n` merges the default and the current page values and add them to the dictionary of the current page in `\g__pdf_backend_thispage_shipout_tl`.

```
%  backend commands
⟨*pdftex⟩
 %the primitive
  \cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
    {
      \tex_global:D \tex_pdfpageattr:D { #1 }
    }
% the command to store default values.
% Uses a prop with pdflatex + dvi,
% sets a lua table with lualatex
 \cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2 %key,value
   {
     \pdfdict_gput:nnn {g__pdf_Core/Page}{ #1 }{ #2 }
   }
% the command to remove a default value.
% Uses a prop with pdflatex + dvi,
% changes a lua table with lualatex
\cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
  {
    \pdfdict_gremove:nn  {g__pdf_Core/Page}{ #1 }
  }
% the command used in the document.
% direct call of the primitive special with dvips/dvipdfmx
% \latelua: fill a page related table with lualatex, merge it with the page
% table and push it directly
% write to aux and store in prop with pdflatex
\cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
  {
    %we need to know the page the resource should be added too.
    \int_gincr:N\g__pdf_backend_resourceid_int
    \__pdf_backend_ref_label:en { l3pdf\int_use:N\g__pdf_backend_resourceid_int }{abspage}
    \tl_set:Nx \l__pdf_tmpa_tl
      {
        \__pdf_backend_ref_value:en {l3pdf\int_use:N\g__pdf_backend_resourceid_int}{abspage}
      }
    \pdfdict_if_exist:nF { g__pdf_Core/backend_Page\l__pdf_tmpa_tl}
      {
        \pdfdict_new:n { g__pdf_Core/backend_Page\l__pdf_tmpa_tl}
      }
    %backend_Page has no handler.
    \pdfdict_gput:nnn {g__pdf_Core/backend_Page\l__pdf_tmpa_tl}{ #1 }{ #2 }
  }
%the code to push the values, used in shipout
%merges the two props and then fills the register in pdflatex
%merges the two tables and then fills (in lua) in luatex
%issues the values stored in the global prop with dvi
\cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
  {
    \prop_gset_eq:Nc \g__pdf_tmpa_prop { \__kernel_pdfdict_name:n { g__pdf_Core/Page } }
    \prop_if_exist:cT  { \__kernel_pdfdict_name:n { g__pdf_Core/backend_Page#1 } }
      {
        \prop_map_inline:cn { \__kernel_pdfdict_name:n { g__pdf_Core/backend_Page#1 } }
          {
            \prop_gput:Nnn \g__pdf_tmpa_prop { ##1 }{ ##2 }
          }
      }
    \exp_args:Nx \__pdf_backend_Page_primitive:n
      {
        \prop_map_function:NN \g__pdf_tmpa_prop \pdfdict_item:ne
      }
  }
⟨/pdftex⟩
⟨*luatex⟩
% do we need to use some escaping for the values?????
\cs_new:Npn \__pdf_backend_luastring:n #1
  {
    "\tex_luaescapestring:D { \tex_unexpanded:D { #1 } }"
  }
 %not used, only there for consistency
\cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
  {
    \tex_latelua:D
      {
        pdf.setpageattributes(\__pdf_backend_luastring:n { #1 })
      }
  }
  % the command to store default values.
  % Uses a prop with pdflatex + dvi,
  % sets a lua table with lualatex
\cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2
  {
    \tex_directlua:D
      {
        ltx.__pdf.backend_Page_gput
          (
            \__pdf_backend_luastring:n { #1 },
            \__pdf_backend_luastring:n { #2 }
          )
      }
  }
  % the command to remove a default value.
  % Uses a prop with pdflatex + dvi,
  % changes a lua table with lualatex
```

```
\cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
  {
    \tex_directlua:D
      {
        ltx.__pdf.backend_Page_gremove (\__pdf_backend_luastring:n { #1 })
      }
  }
 % the command used in the document.
 % direct call of the primitive special with dvips/dvipdfmx
 % \latelua: fill a page related table with lualatex, merge it with the page
 % table and push it directly
 % write to aux and store in prop with pdflatex
\cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
  {
    \tex_latelua:D
      {
        ltx.__pdf.backend_ThisPage_gput
          (
            tex.count["g_shipout_readonly_int"],
            \__pdf_backend_luastring:n { #1 },
            \__pdf_backend_luastring:n { #2 }
          )
        ltx.__pdf.backend_ThisPage_gpush (tex.count["g_shipout_readonly_int"])
      }
  }
  %the code to push the values, used in shipout
  %merges the two props and then fills the register in pdflatex
  %merges the two tables (the one is probably still empty) and then fills (in lua) in luatex
  %issues the values stored in the global prop with dvi
\cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
  {
    \tex_latelua:D
      {
        ltx.__pdf.backend_ThisPage_gpush (tex.count["g_shipout_readonly_int"])
      }
  }

⟨/luatex⟩
⟨*dvipdfmx | xdvipdfmx⟩
  %the primitive
\cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
  {
    \tex_special:D{pdf:~put~@thispage~<<#1>>}
  }
  % the command to store default values.
  % Uses a prop with pdflatex + dvi,
  % sets a lua table with lualatex
\cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2
  {
    \pdfdict_gput:nnn {g__pdf_Core/Page}{ #1 }{ #2 }
  }
  % the command to remove a default value.
  % Uses a prop with pdflatex + dvi,
  % changes a lua table with lualatex
```

```
\cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
   {
     \pdfdict_gremove:nn  {g__pdf_Core/Page}{ #1 }
   }
  % the command used in the document.
  % direct call of the primitive special with dvips/dvipdfmx
  % \latelua: fill a page related table with lualatex, merge it with the page
  % table and push it directly
  % write to aux and store in prop with pdflatex
\cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
  {
    \__pdf_backend_Page_primitive:n { /#1~#2 }
  }
  %the code to push the values, used in shipout
  %merges the two props and then fills the register in pdflatex
  %merges the two tables (the one is probably still empty)
  % and then fills (in lua) in luatex
  %issues the values stored in the global prop with dvi
\cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
  {
    \exp_args:Nx \__pdf_backend_Page_primitive:n
      { \pdfdict_use:n { g__pdf_Core/Page} }
  }
⟨/dvipdfmx | xdvipdfmx⟩
⟨*dvips⟩
\cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
   {
     \tex_special:D{ps:~[{ThisPage}<<#1>>~/PUT~pdfmark} %]
   }
  % the command to store default values.
  % Uses a prop with pdflatex + dvi,
  % sets a lua table with lualatex
\cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2
   {
     \pdfdict_gput:nnn {g__pdf_Core/Page}{ #1 }{ #2 }
   }
  % the command to remove a default value.
  % Uses a prop with pdflatex + dvi,
  % changes a lua table with lualatex
\cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
  {
    \pdfdict_gremove:nn  {g__pdf_Core/Page}{ #1 }
  }
  % the command used in the document.
  % direct call of the primitive special with dvips/dvipdfmx
  % \latelua: fill a page related table with lualatex, merge it with the page
  % table and push it directly
  % write to aux and store in prop with pdflatex
\cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
  {
    \__pdf_backend_Page_primitive:n { /#1~#2 }
  }
  %the code to push the values, used in shipout
  %merges the two props and then fills the register in pdflatex
  %merges the two tables (the one is probably still empty)
  %and then fills (in lua) in luatex
  %issues the values stored in the global prop with dvi
\cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
  {
    \exp_args:Nx \__pdf_backend_Page_primitive:n
          { \pdfdict_use:n { g__pdf_Core/Page} }
  }
⟨/dvips⟩
⟨*dvisvgm⟩
% mostly only dummies ...
\cs_new_protected:Npn \__pdf_backend_Page_primitive:n #1
   {}
  % Uses a prop with pdflatex + dvi,
\cs_new_protected:Npn \__pdf_backend_Page_gput:nn #1 #2
   {
     \pdfdict_gput:nnn {g__pdf_Core/Page}{ #1 }{ #2 }
   }
  % the command to remove a default value.
  % Uses a prop with pdflatex + dvi,
\cs_new_protected:Npn \__pdf_backend_Page_gremove:n #1
  {
    \pdfdict_gremove:nn  {g__pdf_Core/Page}{ #1 }
  }
  % the command used in the document.
\cs_new_protected:Npn \__pdf_backend_ThisPage_gput:nn #1 #2
  {}
  %the code to push the values, used in shipout
\cs_new_protected:Npn \__pdf_backend_ThisPage_gpush:n #1
  {}
⟨/dvisvgm⟩
```

(*End definition for* `\__pdf_backend_Page_primitive:n` *and others.*)

1.7 "Page/Resources": ExtGState, ColorSpace, Shading, Pattern

Path: Page/Resources/ExtGState etc. The actual output of the resources is handled together with the bdc/Properties. Here is only special code.

\c__pdf_backend_PageResources_clist

The names are quite often needed a similar list is now in l3pdfmanagement. Perhaps it should be merged.

```
⟨*drivers⟩
\clist_const:Nn \c__pdf_backend_PageResources_clist
  {
    ExtGState,
    ColorSpace,
    Pattern,
    Shading,
  }
⟨/drivers⟩
```

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

Now the backend commands the command to fill the register and to push the values.

__pdf_backend_PageResources_gput:nnn

stores values for the page resources.
#1 : name of the resource (ExtGState, ColorSpace, Shading, Pattern)
#2 : a pdf name without slash
#3 : value

__pdf_backend_PageResources_obj_gpush:

This pushes out the objects. It should be a no-op with xdvipdfmx and dvips as it currently issued in the end-of-run hook! create the backend objects:

```
⟨*pdftex | luatex⟩
\clist_map_inline:Nn \c__pdf_backend_PageResources_clist
  {
    \pdf_object_new:n {__pdf/Page/Resources/#1}
    \cs_if_exist:NT \tex_directlua:D
      {
        \tex_directlua:D
          {
            ltx.__pdf.object["__pdf/Page/Resources/#1"]
            =
            "\__pdf_backend_object_ref:n{__pdf/Page/Resources/#1}"
          }
      }
   }
⟨/pdftex | luatex⟩
```

values are only stored in a prop and will be output at end document. luatex must also trigger the lua side

```
⟨*luatex⟩
\cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
  {
    \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
    \tex_latelua:D{ltx.__pdf.Page.Resources.#1=true}
    \tex_latelua:D
      {
        ltx.pdf.Page_Resources_gpush(tex.count["g_shipout_readonly_int"])
      }
  }
⟨/luatex⟩
⟨*pdftex⟩
 \cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
   {
     \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
   }
⟨/pdftex⟩
```

code for end of document code

```
⟨*pdftex | luatex⟩
\cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush:
  {
    \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
      {
        \prop_if_empty:cF
          { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/##1} }
          {
            \pdf_object_write:nnx
              { __pdf/Page/Resources/##1 } { dict }
              { \pdfdict_use:n { g__pdf_Core/Page/Resources/##1} }
          }
     }
  }
⟨/pdftex | luatex⟩
```

xdvipdfmx doesn't work correctly with object names ... https://tug.org/pipermail/dvipdfmx/2019-August/000021.html, so we use this must be issued on every page! objects should not only be created but also `initialized` initialization should be done before anyone tries to write so we add rules for the backend. The push command should not be used as it is in the wrong end document hook. If needed a new command must be added.

```
⟨*dvipdfmx | xdvipdfmx⟩
⟨xdvipdfmx⟩\hook_gset_rule:nnnn{shipout/firstpage}{l3backend-xetex}{after}{pdf}
⟨dvipdfmx⟩\hook_gset_rule:nnnn{shipout/firstpage}{l3backend-dvipdfmx}{after}{pdf}
%
\clist_map_inline:Nn \c__pdf_backend_PageResources_clist
  {
    \pdf_object_new:n   { __pdf/Page/Resources/#1 }
    \hook_gput_code:nnn
      {shipout/firstpage}
      {pdf}
      {\pdf_object_write:nnn { __pdf/Page/Resources/#1 } { dict } {}}
  }
\cs_new_protected:Npn \__pdf_backend_PageResources:n #1
  {
    \__pdf_backend:n {put~@resources~<<#1>>}
  }
\cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
  {
   % this is not used for output, but there is a test if the resource is empty
   \exp_args:Nnx
   \prop_gput:cnn { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/#1} }
     { \str_convert_pdfname:n {#2} }{ #3 }
   %objects are not filled with \pdf_object_write as this is not additive!
    \__pdf_backend:x
      {
        put~\__pdf_backend_object_ref:n {__pdf/Page/Resources/#1}<</#2~#3>>
      }
  }

\cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush: {}
⟨/dvipdfmx | xdvipdfmx⟩
```

dvips unneeded, or no-op. The push command should not be used as it is in the wrong end document hook. If needed a new command must be added.

```
⟨*dvips⟩
\cs_new_protected:Npn \__pdf_backend_PageResources:n #1 {}
\cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
  { %only for the show command TEST!!
    \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
  }
\cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush: {}
⟨/dvips⟩
```

dvipsvgm unneeded, or no-op

```
⟨*dvisvgm⟩
\cs_new_protected:Npn \__pdf_backend_PageResources:n #1 {}
\cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
  { %only for the show command TEST!!
    \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
  }
\cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush: {}
⟨/dvisvgm⟩
```

(*End definition for* `\__pdf_backend_PageResources_gput:nnn` *and* `\__pdf_backend_PageResources_obj_gpush:`.)

1.7.1 Page resources /Properties + BDC operators

__pdf_backend_bdc:nn
__pdf_backend_bdcobject:nn
__pdf_backend_bdcobject:n
__pdf_backend_bmc:n
__pdf_backend_emc:
__pdf_backend_PageResources_gpush:n

`\__pdf_backend_bdc:nn`, `\__pdf_backend_bdcobject:nn`, `\__pdf_backend_bdcobject:n`, `\__pdf_backend_bmc:n` and `\__pdf_backend_emc:` are the backend command that create the bdc/emc marker and store the properties. `\__pdf_backend_PageResources_gpush:n` outputs the /Properties and/or the other resources for the current page.

```
% pdftex and luatex (and perhaps dvips ...) need to know if there are in a
% xform stream ...
⟨*drivers⟩
\bool_new:N \l__pdf_backend_xform_bool
⟨/drivers⟩
⟨*dvips⟩
% dvips is easy: create an object, and reference it in the bdc
% ghostscript will then automatically replace it by a name
% and add the name to the /Properties dict
% special variant von accsupp
% https://chat.stackexchange.com/transcript/message/50831812#50831812
%
\cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2 % #1 eg. Span, #2: dict_content
  {
    \__pdf_backend_pdfmark:x{/#1~<<#2>>~/BDC}
  }
\cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
  {
    \__pdf_backend_pdfmark:x{/#1~\__pdf_backend_object_ref:n{#2}~/BDC}
  }
\cs_set_protected:Npn \__pdf_backend_bdcobject:n #1  % #1 eg. Span,
  {
    \__pdf_backend_pdfmark:x{/#1~\__pdf_backend_object_last:~/BDC}
  }
\cs_set_protected:Npn \__pdf_backend_emc:
  {
    \__pdf_backend_pdfmark:n{/EMC} %
  }
\cs_set_protected:Npn \__pdf_backend_bmc:n #1
  {
    \__pdf_backend_pdfmark:n{/#1~/BMC} %
  }
\cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1 {}

⟨/dvips⟩
⟨*dvisvgm⟩
```

```
% dvisvgm should do nothing
%
\cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2 % #1 eg. Span, #2: dict_content
  {}
\cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
  {}
\cs_set_protected:Npn \__pdf_backend_bdcobject:n #1  % #1 eg. Span,
  {}
\cs_set_protected:Npn \__pdf_backend_emc:
  {}
\cs_set_protected:Npn \__pdf_backend_bmc:n #1
  {}
\cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1 {}

⟨/dvisvgm⟩

% xetex has to create the entries in the /Properties manually
% (like the other backends)
% use pdfbase special
% https://chat.stackexchange.com/transcript/message/50832016#50832016
% the property is added to xform resources automatically,
% no need to worry about it.
⟨*dvipdfmx | xdvipdfmx⟩
 \cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
   {
     \int_gincr:N \g__pdf_backend_name_int
     \__kernel_backend_literal:x
       {
         pdf:code~/#1/l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC
       }
     \__kernel_backend_literal:x
       {
         pdf:put~@resources~
           <<
             /Properties~
               <<
                 /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl
                 \__pdf_backend_object_ref:n { #2 }
               >>
           >>
       }
   }
 \cs_set_protected:Npn \__pdf_backend_bdcobject:n #1  % #1 eg. Span
   {
     \int_gincr:N \g__pdf_backend_name_int
     \__kernel_backend_literal:x
       {
         pdf:code~/#1/l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC
       }
     \__kernel_backend_literal:x
       {
         pdf:put~@resources~
           <<
             /Properties~
```

```
               <<
                 /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl
                 \__pdf_backend_object_last:
               >>
           >>
       }
   }
\cs_set_protected:Npn \__pdf_backend_bmc:n #1
   {
     \__kernel_backend_literal:n {pdf:code~/#1~BMC}  %pdfbase
   }

%this require management
\cs_set_protected:Npn \__pdf_backend_bdc_contobj:nn #1 #2
  {
    \pdf_object_unnamed_write:nn { dict }{ #2 }
    \__pdf_backend_bdcobject:n { #1 }
  }

\cs_set_protected:Npn \__pdf_backend_bdc_contstream:nn #1 #2
  {
    \__kernel_backend_literal:n {pdf:code~ /#1~<<#2>>~BDC }
  }

\cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2
  {
    \bool_if:NTF  \g__pdfmanagement_active_bool
      {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contobj:nn}
      {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contstream:nn}
      \__pdf_backend_bdc:nn {#1}{#2}
  }
\cs_set_protected:Npn \__pdf_backend_emc:
  {
    \__kernel_backend_literal:n {pdf:code~EMC}  %pdfbase
  }
  % properties are handled automatically, but the other resources should be added
  % at shipout
\cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1
  {
    \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
      {
        \prop_if_empty:cF { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/##1} }
          {
            \__kernel_backend_literal:x
              {
                pdf:put~@resources~
                  <</##1~\__pdf_backend_object_ref:n {__pdf/Page/Resources/##1}>>
              }
          }
      }
  }
⟨/dvipdfmx | xdvipdfmx⟩
% luatex + pdftex
⟨*luatex⟩
```

```
\cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
  {
    \int_gincr:N \g__pdf_backend_name_int
    \exp_args:Nx\__kernel_backend_literal_page:n
      { /#1 ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
    \bool_if:NTF \l__pdf_backend_xform_bool
      {
        \exp_args:Nnx\pdfdict_gput:nnn
          { g__pdf_Core/Xform/Resources/Properties }
          { l3pdf\int_use:N\g__pdf_backend_name_int }
          { \__pdf_backend_object_ref:n { #2 } }
      }
      {
        \exp_args:Nx \tex_latelua:D
          {
            ltx.pdf.Page_Resources_Properties_gput
              (
                tex.count["g_shipout_readonly_int"],
                "l3pdf\int_use:N\g__pdf_backend_name_int",
                "\__pdf_backend_object_ref:n { #2 }"
              )
          }
      }
  }
\cs_set_protected:Npn \__pdf_backend_bdcobject:n #1% #1 eg. Span
  {
    \int_gincr:N \g__pdf_backend_name_int
    \exp_args:Nx\__kernel_backend_literal_page:n
      { /#1 ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
    \bool_if:NTF \l__pdf_backend_xform_bool
      {
        \exp_args:Nnx\pdfdict_gput:nnn %no handler needed
          { g__pdf_Core/Xform/Resources/Properties }
          { l3pdf\int_use:N\g__pdf_backend_name_int }
          { \__pdf_backend_object_last: }
      }
      {
        \exp_args:Nx \tex_latelua:D
          {
            ltx.pdf.Page_Resources_Properties_gput
              (
                tex.count["g_shipout_readonly_int"],
                "l3pdf\int_use:N\g__pdf_backend_name_int",
                "\__pdf_backend_object_last:"
              )
          }
      }
  }
\cs_set_protected:Npn \__pdf_backend_bmc:n #1
   {
     \__kernel_backend_literal_page:n { /#1~BMC }
   }
\cs_set_protected:Npn \__pdf_backend_bdc_contobj:nn #1 #2
  {
    \pdf_object_unnamed_write:nn { dict } { #2 }
    \__pdf_backend_bdcobject:n { #1 }
  }
\cs_set_protected:Npn \__pdf_backend_bdc_contstream:nn #1 #2
  {
    \__kernel_backend_literal_page:n { /#1~<<#2>>~BDC }
  }
\cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2
  {
    \bool_if:NTF  \g__pdfmanagement_active_bool
      {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contobj:nn}
      {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contstream:nn}
      \__pdf_backend_bdc:nn {#1}{#2}
  }
\cs_set_protected:Npn \__pdf_backend_emc:
  {
    \__kernel_backend_literal_page:n { EMC }
  }

\cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1 {}
⟨/luatex⟩
⟨*pdftex⟩
% pdflatex is the most complicated as it has to go through the aux ...
% the push command is extended to take other resources too
\cs_set_protected:Npn \__pdf_backend_bdcobject:nn #1 #2 % #1 eg. Span, #2: object name
  {
    \int_gincr:N \g__pdf_backend_name_int
    \exp_args:Nx\__kernel_backend_literal_page:n
      { /#1 ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
    % code to set the property ....
    \int_gincr:N\g__pdf_backend_resourceid_int
    \bool_if:NTF \l__pdf_backend_xform_bool
      {
        \exp_args:Nnxx\pdfdict_gput:nnn %no handler needed
          { g__pdf_Core/Xform/Resources/Properties }
          { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
          { \__pdf_backend_object_ref:n { #2 } }
      }
      {
        \__pdf_backend_ref_label:en{l3pdf\int_use:N\g__pdf_backend_resourceid_int}{abspage}
        \tl_set:Nx \l__pdf_tmpa_tl
          {
            \__pdf_backend_ref_value:en{l3pdf\int_use:N\g__pdf_backend_resourceid_int}{abspa
          }
        \pdfdict_if_exist:nF { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties
          {
            \pdfdict_new:n { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
          }
        \exp_args:Nnxx\pdfdict_gput:nnn
          { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
          { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
          { \__pdf_backend_object_ref:n{#2} }
      }
  }
```

```
\cs_set_protected:Npn \__pdf_backend_bdcobject:n #1% #1 eg. Span
  {
    \int_gincr:N \g__pdf_backend_name_int
    \exp_args:Nx\__kernel_backend_literal_page:n
      { /#1 ~ /l3pdf\int_use:N\g__pdf_backend_name_int\c_space_tl BDC }
    % code to set the property ....
    \int_gincr:N\g__pdf_backend_resourceid_int
    \bool_if:NTF \l__pdf_backend_xform_bool
      {
        \exp_args:Nnxx\pdfdict_gput:nnn
          { g__pdf_Core/Xform/Resources/Properties }
          { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
          { \__pdf_backend_object_last: }
      }
      {
        \__pdf_backend_ref_label:en{l3pdf\int_use:N\g__pdf_backend_resourceid_int}{abspage}
        \tl_set:Nx \l__pdf_tmpa_tl
          {
            \__pdf_backend_ref_value:en{l3pdf\int_use:N\g__pdf_backend_resourceid_int}{abspa
          }
        \pdfdict_if_exist:nF { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties
          {
            \pdfdict_new:n { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
          }
        \exp_args:Nnxx\pdfdict_gput:nnn
          { g__pdf_Core/backend_Page\l__pdf_tmpa_tl/Resources/Properties }
          { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
          { \__pdf_backend_object_last: }
        %\pdfdict_show:n { g_backend_Page\l__pdf_tmpa_tl/Resources/Properties }
      }
  }
\cs_set_protected:Npn \__pdf_backend_bmc:n #1
  {
    \__kernel_backend_literal_page:n { /#1~BMC }
  }
\cs_set_protected:Npn \__pdf_backend_bdc_contobj:nn #1 #2
   {
     \pdf_object_unnamed_write:nn { dict } { #2 }
     \__pdf_backend_bdcobject:n { #1 }
   }
\cs_set_protected:Npn \__pdf_backend_bdc_contstream:nn #1 #2
   {
     \__kernel_backend_literal_page:n { /#1~<<#2>>~BDC }
   }
\cs_set_protected:Npn \__pdf_backend_bdc:nn #1 #2
  {
    \bool_if:NTF  \g__pdfmanagement_active_bool
      {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contobj:nn}
      {\cs_gset_eq:NN \__pdf_backend_bdc:nn \__pdf_backend_bdc_contstream:nn}
      \__pdf_backend_bdc:nn {#1}{#2}
  }
\cs_set_protected:Npn \__pdf_backend_emc:
  {
    \__kernel_backend_literal_page:n { EMC }
  }

\cs_new:Npn \__pdf_backend_PageResources_gpush_aux:n #1 %#1 ExtGState etc
  {
    \prop_if_empty:cF
      { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/#1} }
      {
        \pdfdict_item:ne { #1 }{ \pdf_object_ref:n {__pdf/Page/Resources/#1}}
      }
  }

\cs_new_protected:Npn \__pdf_backend_PageResources_gpush:n #1
  {
     \exp_args:NNx \tex_global:D \tex_pdfpageresources:D
       {
         \prop_if_exist:cT
           { \__kernel_pdfdict_name:n { g__pdf_Core/backend_Page#1/Resources/Properties } }
           {
             /Properties~
               <<
                 \prop_map_function:cN
                   { \__kernel_pdfdict_name:n { g__pdf_Core/backend_Page#1/Resources/Propert
                   \pdfdict_item:ne
               >>
           }
         %% add ExtGState etc
         \clist_map_function:NN
           \c__pdf_backend_PageResources_clist
           \__pdf_backend_PageResources_gpush_aux:n
       }
  }

⟨/pdftex⟩
```

(*End definition for* `\__pdf_backend_bdc:nn` *and others.*)

1.8 “Catalog” & subdirectories (pdfcatalog)

The backend command is already in the driver: `\__pdf_backend_catalog_gput:nn`

1.8.1 Special case: the /Names/EmbeddedFiles dictionary

Entries to /Names are handled differently, in part (/Desc) it is automatic, for other special commands like `\pdfnames` must be used. For EmbeddedFiles dvips wants code for every file and then creates the Name tree automatically. Other name trees are ignored. TODO: Currently the code for EmbeddedFiles is still a bit different but this should be merged, all name trees should be handled with the same code.

```
% pdflatex
⟨*pdftex⟩
\cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 %#1 name of name tree, #2 array co
  {
     \pdf_object_unnamed_write:nn {dict} {/Names [#2] }
     \tex_pdfnames:D {/#1~\pdf_object_ref_last:}
  }
```

```
⟨/pdftex⟩
⟨*luatex⟩
\cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 %#1 name of name tree, #2 array co
  {
     \pdf_object_unnamed_write:nn {dict} {/Names [#2] }
     \tex_pdfextension:D~names~ {/#1~\pdf_object_ref_last:}
  }
⟨/luatex⟩
⟨*dvipdfmx | xdvipdfmx⟩
\cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 %#1 name of name tree, #2 array co
  {
     \pdf_object_unnamed_write:nn {dict} {/Names [#2] }
     \__pdf_backend:x {put~@names~<</#1~\pdf_object_ref_last: >>}
  }
⟨/dvipdfmx | xdvipdfmx⟩

%dvips: noop
⟨*dvips⟩
\cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2  {}
⟨/dvips⟩
%dvisvgm: noop
⟨*dvisvgm⟩
\cs_new_protected:Npn \__pdf_backend_Names_gpush:nn #1 #2 {}
⟨/dvisvgm⟩
```

EmbeddedFiles is a bit special. For once we need backend commands for dvips. But we want also an option to create the name on the fly.

__pdf_backend_NamesEmbeddedFiles_add:nn

dvips need special backend code to create the name tree. With the other engines it does nothing.

```
⟨*pdftex | luatex | dvipdfmx | xdvipdfmx⟩
\cs_new_protected:Npn  \__pdf_backend_NamesEmbeddedFiles_add:nn #1 #2 {}
⟨/pdftex | luatex | dvipdfmx | xdvipdfmx⟩
⟨*dvips⟩
\cs_new_protected:Npn  \__pdf_backend_NamesEmbeddedFiles_add:nn #1 #2
      {
        \__pdf_backend_pdfmark:x
          {
            /Name~#1~
            /FS~#2~
            /EMBED
          }
      }
⟨/dvips⟩
⟨*dvisvgm⟩
%no op. Or is there any sensible use for it?
\cs_new_protected:Npn  \__pdf_backend_NamesEmbeddedFiles_add:nn #1 #2
      {}

⟨/dvisvgm⟩
```

(*End definition for* __pdf_backend_NamesEmbeddedFiles_add:nn.)

1.8.2 Additional annotation commands

Starting with texlive 2021 pdftex and luatex offer commands to interrupt a link. That can for example be used to exclude the header and footer from the link. We add here backend support for this.

```
⟨*drivers⟩
\cs_new_protected:Npn \__pdf_backend_link_off:{}
\cs_new_protected:Npn \__pdf_backend_link_on: {}
⟨/drivers⟩
⟨*pdftex⟩
\cs_if_exist:NT \pdfrunninglinkoff
 {
   \cs_set_protected:Npn \__pdf_backend_link_off:
    {
      \pdfrunninglinkoff
    }
   \cs_set_protected:Npn \__pdf_backend_link_on:
    {
      \pdfrunninglinkon
    }
 }
⟨/pdftex⟩
⟨*luatex⟩
\int_compare:nNnT {\tex_luatexversion:D } > {112}
 {
   \cs_set_protected:Npn \__pdf_backend_link_off:
    {
      \pdfextension linkstate 1
    }
   \cs_set_protected:Npn \__pdf_backend_link_on:
    {
      \pdfextension linkstate 0
    }
 }
⟨/luatex⟩
⟨*dvipdfmx | xdvipdfmx⟩
  \cs_set_protected:Npn \__pdf_backend_link_off:
    {
     \__pdf_backend:n { nolink }
    }
   \cs_set_protected:Npn \__pdf_backend_link_on:
    {
      \__pdf_backend:n { link }
    }
⟨/dvipdfmx | xdvipdfmx⟩
```

1.8.3 Form XObject / backend

__pdf_backend_xform_new:nnnn

#1 : name
#2 : attributes
#3 : resources needed?? or are all resources autogenerated?
#4 : content, this doesn't need to be a box!

__pdf_backend_xform_use:n
__pdf_backend_xform_ref:n

```
⟨*pdftex⟩
\cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4
% #1 name
% #2 attributes
% #3 resources
% #4 content, not necessarily a box!
  {
    \hbox_set:Nn \l__pdf_backend_tmpa_box
      {
        \bool_set_true:N \l__pdf_backend_xform_bool
        \prop_gclear:c {\__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties }}
        #4
      }
    %store the dimensions
    \tl_const:cx
      { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
      { \tex_the:D \box_wd:N \l__pdf_backend_tmpa_box }
    \tl_const:cx
      { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
      { \tex_the:D \box_ht:N \l__pdf_backend_tmpa_box }
    \tl_const:cx
      { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
      { \tex_the:D \box_dp:N \l__pdf_backend_tmpa_box }
    %% do we need to test if #2 and #3 are empty??
    \tex_immediate:D \tex_pdfxform:D
      ~  attr      ~ { #2 }
    %% which other resources should be default? Is an argument actually needed?
      ~  resources ~
      {
        #3
        \int_compare:nNnT
          { \prop_count:c { \__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Propertie
          >
          { 0 }
          {
            /Properties~
              <<
                \pdfdict_use:n { g__pdf_Core/Xform/Resources/Properties }
              >>
          }

        \prop_if_empty:cF
          { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/ExtGState } }
          {
            /ExtGState~ \pdf_object_ref:n { __pdf/Page/Resources/ExtGState }
          }
        \prop_if_empty:cF
          { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/Pattern } }
          {
            /Pattern~ \pdf_object_ref:n { __pdf/Page/Resources/Pattern }
          }
        \prop_if_empty:cF
          { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/Shading } }
          {
            /Shading~ \pdf_object_ref:n { __pdf/Page/Resources/Shading }
          }
        \prop_if_empty:cF
          { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/ColorSpace } }
          {
            /ColorSpace~ \pdf_object_ref:n { __pdf/Page/Resources/ColorSpace }
          }
      }
      \l__pdf_backend_tmpa_box
   \int_const:cn
     { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
     { \tex_pdflastxform:D }
  }

\cs_new_protected:Npn \__pdf_backend_xform_use:n #1
  {
    \tex_pdfrefxform:D
      \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
      \scan_stop:
  }

\cs_new:Npn \__pdf_backend_xform_ref:n #1
  {
    \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int } ~ 0 ~ R
  }
⟨/pdftex⟩
⟨*luatex⟩
%luatex
%nearly identical but not completely ...
\cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4
% #1 name
% #2 attributes
% #3 resources
% #4 content, not necessarily a box!
  {
    \hbox_set:Nn \l__pdf_backend_tmpa_box
      {
        \bool_set_true:N \l__pdf_backend_xform_bool
        \prop_gclear:c { \__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties }
        #4
      }
    \tl_const:cx
      { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
      { \tex_the:D \box_wd:N \l__pdf_backend_tmpa_box }
    \tl_const:cx
      { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
      { \tex_the:D \box_ht:N \l__pdf_backend_tmpa_box }
    \tl_const:cx
      { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
      { \tex_the:D \box_dp:N \l__pdf_backend_tmpa_box }
    %% do we need to test if #2 and #3 are empty??
    \tex_immediate:D \tex_pdfxform:D
      ~  attr      ~ { #2 }
      %% which resources should be default? Is an argument actually needed?
      ~  resources ~
      {
        #3
        \int_compare:nNnT
          {\prop_count:c { \__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties
          >
          { 0 }
          {
            /Properties~
              <<
                \pdfdict_use:n { g__pdf_Core/Xform/Resources/Properties }
              >>
          }
        \prop_if_empty:cF
          { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/ExtGState } }
          {
            /ExtGState~ \pdf_object_ref:n { __pdf/Page/Resources/ExtGState }
          }
        \prop_if_empty:cF
          { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/Pattern } }
          {
            /Pattern~ \pdf_object_ref:n { __pdf/Page/Resources/Pattern }
          }
        \prop_if_empty:cF
          { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/Shading } }
          {
            /Shading~ \pdf_object_ref:n { __pdf/Page/Resources/Shading }
          }
        \prop_if_empty:cF
          { \__kernel_pdfdict_name:n { g__pdf_Core/Page/Resources/ColorSpace } }
          {
            /ColorSpace~ \pdf_object_ref:n { __pdf/Page/Resources/ColorSpace }
          }
      }
      \l__pdf_backend_tmpa_box
      \int_const:cn
        { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
        { \tex_pdflastxform:D }
  }

\cs_new_protected:Npn \__pdf_backend_xform_use:n #1 %protected as with xelatex
  {
    \tex_pdfrefxform:D \int_use:c
      {
        c__pdf_backend_xform_ \tl_to_str:n {#1} _int
      }
      \scan_stop:
  }

\cs_new:Npn \__pdf_backend_xform_ref:n #1
  { \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int } ~ 0 ~ R }

⟨/luatex⟩
⟨*dvipdfmx | xdvipdfmx⟩
```

```
% xetex
   % it needs a bit testing if it really works to set the box to 0 before the special ...
   % does it disturb viewing the xobject?
   % what happens with the resources (bdc)? (should work as they are specials too)
   % xetex requires that the special is in horizontal mode. This means it affects
   % typesetting. But we can no delay the whole form code to shipout
   % as the object reference and the size is often wanted on the current page.
   % so we need to allocate a box - but probably they won't be thousands xform
   % in a document so it shouldn't matter.
    \cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4
    % #1 name
    % #2 attributes
    % #3 resources
    % #4 content, not necessarily a box!
      {
         \int_gincr:N \g__pdf_backend_object_int
         \int_const:cn
           { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
           { \g__pdf_backend_object_int }
         \box_new:c { g__pdf_backend_xform_#1_box }
         \hbox_gset:cn { g__pdf_backend_xform_#1_box }
           {
             \bool_set_true:N \l__pdf_backend_xform_bool
             #4
           }
         \tl_const:cx
           { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
           { \tex_the:D \box_wd:c { g__pdf_backend_xform_#1_box } }
         \tl_const:cx
           { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
           { \tex_the:D \box_ht:c { g__pdf_backend_xform_#1_box } }
         \tl_const:cx
           { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
           { \tex_the:D \box_dp:c { g__pdf_backend_xform_#1_box } }
         \box_set_dp:cn  { g__pdf_backend_xform_#1_box } { \c_zero_dim }
         \box_set_ht:cn  { g__pdf_backend_xform_#1_box } { \c_zero_dim }
         \box_set_wd:cn  { g__pdf_backend_xform_#1_box } { \c_zero_dim }
         \hook_gput_next_code:nn {shipout/background}
           {
             \mode_leave_vertical: %needed, the xform disappears without it.
             \__pdf_backend:x
               {
                 bxobj  ~ \__pdf_backend_xform_ref:n  { #1 }
                 \c_space_tl width  ~ \pdfxform_wd:n { #1 }
                 \c_space_tl height ~ \pdfxform_ht:n { #1 }
                 \c_space_tl depth  ~ \pdfxform_dp:n { #1 }
               }
             \box_use_drop:c { g__pdf_backend_xform_#1_box }
             \__pdf_backend:x {put ~ @resources ~<<#3>> }
             \__pdf_backend:x
               {
                 put~ @resources ~
                   <<
                     /ExtGState~ \pdf_object_ref:n { __pdf/Page/Resources/ExtGState }
```

```
          >>
        }
       \__pdf_backend:x
        {
         put~ @resources ~
         <<
          /Pattern~ \pdf_object_ref:n { __pdf/Page/Resources/Pattern }
         >>
        }
       \__pdf_backend:x
        {
         put~ @resources ~
         <<
          /Shading~ \pdf_object_ref:n { __pdf/Page/Resources/Shading }
         >>
        }
       \__pdf_backend:x
        {
         put~ @resources ~
         <<
          /ColorSpace~
          \pdf_object_ref:n { __pdf/Page/Resources/ColorSpace }
         >>
        }
       \exp_args:Nx
       \__pdf_backend:x {exobj ~<<#2>>}
      }
   }



  \cs_new:Npn \__pdf_backend_xform_ref:n #1
   {
    @pdf.xform \int_use:c { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
   }

  \cs_new_protected:Npn \__pdf_backend_xform_use:n #1
   {
    \hbox_set:Nn \l__pdf_backend_tmpa_box
     {
      \__pdf_backend:x
       {
        uxobj~ \__pdf_backend_xform_ref:n { #1 }
       }
     }
    \box_set_wd:Nn  \l__pdf_backend_tmpa_box { \pdfxform_wd:n { #1 } }
    \box_set_ht:Nn  \l__pdf_backend_tmpa_box { \pdfxform_ht:n { #1 } }
    \box_set_dp:Nn  \l__pdf_backend_tmpa_box { \pdfxform_dp:n { #1 } }
    \box_use_drop:N \l__pdf_backend_tmpa_box
   }
⟨/dvipdfmx | xdvipdfmx⟩
⟨*dvisvgm⟩
% unclear what it should do!!
\cs_new_protected:Npn \__pdf_backend_xform_new:nnnn #1 #2 #3 #4 {}
```

```
\cs_new_protected:Npn \__pdf_backend_xform_use:n #1 {}
\cs_new:Npn \__pdf_backend_xform_ref:n {}
⟨/dvisvgm⟩
```

The xform code for dvips is based on code from the attachfile2 package (in atfi-dvips), along with some ideas from pdfbase and has been corrected with the help of Alexander Grahn. Details like clipping and landscape will probably be corrected in the future. We need some temporary variables to store dimensions

```
⟨*dvips⟩
\tl_new:N \l__pdf_backend_xform_tmpwd_tl
\tl_new:N \l__pdf_backend_xform_tmpdp_tl
\tl_new:N \l__pdf_backend_xform_tmpht_tl

\cs_new_protected:Npn\__pdf_backend_xform_new:nnnn #1 #2 #3 #4 % #1 name, #2 attribute, #4
  {
    \int_gincr:N \g__pdf_backend_object_int
    \int_const:cn
      { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
      { \g__pdf_backend_object_int }

    \hbox_set:Nn \l__pdf_backend_tmpa_box
      {
        \bool_set_true:N \l__pdf_backend_xform_bool
        \prop_gclear:c {\__kernel_pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties }}
        #4
      }
   %store the dimensions
    \tl_const:cx
      { c__pdf_backend_xform_wd_ \tl_to_str:n {#1} _tl }
      { \tex_the:D \box_wd:N \l__pdf_backend_tmpa_box }
    \tl_const:cx
      { c__pdf_backend_xform_ht_ \tl_to_str:n {#1} _tl }
      { \tex_the:D \box_ht:N \l__pdf_backend_tmpa_box }
    \tl_const:cx
      { c__pdf_backend_xform_dp_ \tl_to_str:n {#1} _tl }
      { \tex_the:D \box_dp:N \l__pdf_backend_tmpa_box }
   %store content dimensions in DPI units (Dots) (code from issue 25)
    \tl_set:Nx\l__pdf_backend_xform_tmpwd_tl
      {
        \dim_to_decimal_in_sp:n{ \box_wd:N \l__pdf_backend_tmpa_box }~
        65536~div~72.27~div~DVImag~mul~Resolution~mul~
      }
    \tl_set:Nx\l__pdf_backend_xform_tmpht_tl
      {
        \dim_to_decimal_in_sp:n{ \box_ht:N \l__pdf_backend_tmpa_box }~
        65536~div~72.27~div~DVImag~mul~VResolution~mul~
      }
    \tl_set:Nx\l__pdf_backend_xform_tmpdp_tl
      {
        \dim_to_decimal_in_sp:n{ \box_dp:N \l__pdf_backend_tmpa_box }~
        65536~div~72.27~div~DVImag~mul~VResolution~mul~
      }
    % mirror the box
    %\box_scale:Nnn \l__pdf_backend_tmpa_box {1} {-1}
    \hbox_set:Nn\l__pdf_backend_tmpb_box
```

```
      {
        \__kernel_backend_postscript:x
          {
            gsave~currentpoint~
            initclip~ % restore default clipping path (page device/whole page)
            clippath~pathbbox~newpath~pop~pop~
            \tl_use:N\l__pdf_backend_xform_tmpdp_tl~add~translate~
            mark~
              /_objdef~{ pdf.obj \int_use:N\g__pdf_backend_object_int }\c_space_tl~
              /BBox[
                0~
                \tl_use:N\l__pdf_backend_xform_tmpht_tl~
                \tl_use:N\l__pdf_backend_xform_tmpwd_tl~
                \tl_use:N\l__pdf_backend_xform_tmpdp_tl~
                neg
              ]
              \str_if_eq:eeF{#1}{}
               {
                 product~(Distiller)~search~{pop~pop~pop~#2}{pop}ifelse~
               }
            /BP~pdfmark~1~-1~scale~neg~exch~neg~exch~translate
          }
        \box_use_drop:N\l__pdf_backend_tmpa_box
        \__kernel_backend_postscript:n
          {
            mark ~ /EP~pdfmark  ~ grestore
          }
       \str_if_eq:eeF{#1}{}
         {
           \__kernel_backend_postscript:x
             {
               product~(Ghostscript)~search~
                {
                  pop~pop~pop~
                  mark~
                  { pdf.obj \int_use:c{c__pdf_backend_xform_ \tl_to_str:n {#1} _int} }
                    ~<<#2>>~/PUT~pdfmark
                 }{pop}ifelse
              }
          }
      }
    \box_set_dp:Nn \l__pdf_backend_tmpb_box { \c_zero_dim }
    \box_set_ht:Nn \l__pdf_backend_tmpb_box { \c_zero_dim }
    \box_set_wd:Nn \l__pdf_backend_tmpb_box { \c_zero_dim }
    \hook_gput_code:nnn {begindocument/end}{pdfxform}
      {
        \mode_leave_vertical:
       \box_use:N\l__pdf_backend_tmpb_box
      }
  }


\cs_new_protected:Npn \__pdf_backend_xform_use:n #1
  {
    \hbox_set:Nn \l__pdf_backend_tmpa_box
      {
        \__kernel_backend_postscript:x
         {
           gsave~currentpoint~translate~1~-1~scale~
           mark~{ pdf.obj \int_use:c{c__pdf_backend_xform_ \tl_to_str:n {#1} _int }}~
          /SP~pdfmark ~ grestore
         }
      }
    \box_set_wd:Nn  \l__pdf_backend_tmpa_box { \pdfxform_wd:n { #1 } }
    \box_set_ht:Nn  \l__pdf_backend_tmpa_box { \pdfxform_ht:n { #1 } }
    \box_set_dp:Nn  \l__pdf_backend_tmpa_box { \pdfxform_dp:n { #1 } }
    \box_use_drop:N \l__pdf_backend_tmpa_box
  }
\cs_new:Npn \__pdf_backend_xform_ref:n #1
  {
    { pdf.obj \int_use:c{c__pdf_backend_xform_ \tl_to_str:n {#1} _int} }
  }

⟨/dvips⟩
⟨*drivers⟩
%% all
\prg_new_conditional:Npnn \__pdf_backend_xform_if_exist:n #1 { p , T , F , TF }
  {
    \int_if_exist:cTF { c__pdf_backend_xform_ \tl_to_str:n {#1} _int }
     { \prg_return_true: }
     { \prg_return_false:}
  }
\prg_new_eq_conditional:NNn \pdfxform_if_exist:n\__pdf_backend_xform_if_exist:n
  { TF , T , F , p }
⟨/drivers⟩
```

(*End definition for* `\__pdf_backend_xform_new:nnnn`*,* `\__pdf_backend_xform_use:n`*, and* `\__pdf_backend_xform_ref:n`*.*)

1.9 Structure Destinations

Standard destinations consist of a reference to a page in the pdf and instructions how to display it—typically they will put a specific location in the left top corner of the viewer and so give the impression that a link jumped to the word in this place. But in reality they are not connected to the content.

Starting with pdf 2.0 destinations can in a tagged PDF also point to a structure, to a /StructElem object. GoTo links can then additionally to the `/D` key pointing to a page destination also point to such a structure destination with an `/SD` key. Programs that e.g. convert such a PDF to html can then create better links. (According to the reference, PDF-viewer should prefer the structure destination over the page destination, but as far as it is known this isn't done yet.)

Currently structure destinations and GoTo links making use of it could natively only be created with the dvipdfmx backend. With pdftex and lualatex it was only possible to create a restricted type which used only the "Fit" mode. Starting with TEXlive 2022 (earlier in miktex) both engine will knew new keywords which allow to create structure destination easily.

The following backend code prepares the use of structure destinations. The general idea is that if structure destinations are used, they should be used always. So we define alternative commands which can be activated by mapping them to the standard backend commands.

\l_pdf_current_structure_destination_tl

This commands holds the name of the structure object to use in the next command which creates a destination. The code which activates structure destinations must also ensure that it has a sensible, expandable content. tagpdf for example will define it as

```
\tl_set:Nn \l_pdf_current_structure_destination_tl { __tag/struct/\g__tag_struct_stack_
```

```
⟨*drivers⟩
\tl_new:N   \l_pdf_current_structure_destination_tl
⟨/drivers⟩
```

(*End definition for* \l_pdf_current_structure_destination_tl*. This function is documented on page* ??*.*)

We will define alternatives for three backend commands:

```
\__pdf_backend_destination:nn       -> \__pdf_backend_structure_destination:nn
\__pdf_backend_destination:nnnn -> \__pdf_backend_structure_destination:nnnn
\__pdf_backend_link_begin_goto:nnw  -> \__pdf_backend_link_begin_structure_goto:nnw
```

Activating means mapping them onto the original commands. Be aware that not all engines and compilation routes support structure destinations, for them the command will be a no-op.

\pdf_activate_structure_destination:

```
⟨*drivers⟩
\cs_new_protected:Npn \pdf_activate_structure_destination:
 {
   \cs_gset_eq:NN \__pdf_backend_destination:nn \__pdf_backend_structure_destination:nn
   \cs_gset_eq:NN \__pdf_backend_destination:nnnn \__pdf_backend_structure_destination:nnnn
   \cs_gset_eq:NN \__pdf_backend_link_begin_goto:nnw  \__pdf_backend_link_begin_structure_go
 }
⟨/drivers⟩
```

(*End definition for* \pdf_activate_structure_destination:*. This function is documented on page* ??*.*)

Now the driver dependant parts. By default the new commands are simply copies of the original commands. We adapt them then for the engines and engine version which provide support for structure destinations.

```
⟨*drivers⟩
\cs_set_eq:NN \__pdf_backend_structure_destination:nn       \__pdf_backend_destination:nn
\cs_set_eq:NN \__pdf_backend_structure_destination:nnnn \__pdf_backend_destination:nnnn
\cs_set_eq:NN \__pdf_backend_link_begin_structure_goto:nnw  \__pdf_backend_link_begin_goto:n
⟨/drivers⟩
```

__pdf_backend_structure_destination:nn

This command is the backend command to create a destination. It should in parallel create also a structure destination. At first xetex/dvipdfmx. The structure destination is an array, so we use obj for it so that we can reference it:

```
⟨*xdvipdfmx | dvipdfmx⟩
\cs_set_protected:Npn \__pdf_backend_structure_destination:nn #1#2
  {
    \__pdf_backend:x
```

```
      {
        dest ~ ( \exp_not:n {#1} )
        [
          @thispage
          \str_case:nnF {#2}
            {
              { xyz }   { /XYZ ~ @xpos ~ @ypos ~ null }
              { fit }   { /Fit }
              { fitb }  { /FitB }
              { fitbh } { /FitBH }
              { fitbv } { /FitBV ~ @xpos }
              { fith }  { /FitH ~ @ypos }
              { fitv }  { /FitV ~ @xpos }
              { fitr }  { /Fit }
            }
            { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
        ]
      }
```

We test if the structure object exist. The object of the structure destination gets the name `@pdf.Sdest.`⟨*destname*⟩, where ⟨*destname*⟩ is the name of the standard destination so that we can reference it in the GoTo links.

```
    \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
      {
        \__pdf_backend:x
         {
          obj ~ @pdf.SDest.\exp_not:n{#1}
          [
            \exp_args:Ne \pdf_object_ref:n { \l_pdf_current_structure_destination_tl }
            \str_case:nnF {#2}
              {
                { xyz }   { /XYZ ~ @xpos ~ @ypos ~ null }
                { fit }   { /Fit }
                { fitb }  { /FitB }
                { fitbh } { /FitBH }
                { fitbv } { /FitBV ~ @xpos }
                { fith }  { /FitH ~ @ypos }
                { fitv }  { /FitV ~ @xpos }
                { fitr }  { /Fit }
              }
              { /XYZ ~ @xpos ~ @ypos ~ \fp_eval:n { (#2) / 100 } }
          ]
        }
      }
  }
```

The second destination command is for the boxed destination. Here we need to define an new auxiliary command:

```
\cs_new_protected:Npn \__pdf_backend_structure_destination_aux:nnnn #1#2#3#4
  {
    \vbox_to_zero:n
      {
        \__kernel_kern:n {#4}
        \hbox:n
          {
            \__pdf_backend:n { obj ~ @pdf_ #2 _llx ~ @xpos }
            \__pdf_backend:n { obj ~ @pdf_ #2 _lly ~ @ypos }
          }
        \tex_vss:D
      }
    \__kernel_kern:n {#1}
    \vbox_to_zero:n
      {
        \__kernel_kern:n { -#3 }
        \hbox:n
          {
            \__pdf_backend:n
              {
                dest ~ (#2)
                [
                  @thispage
                  /FitR ~
                    @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
                    @xpos ~ @ypos
                ]
              }
```

Here we add the structure destination to the same box

```
            \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
              {
                \__pdf_backend:x
                  {
                    obj ~ @pdf.SDest.\exp_not:n{#2}
                    [
                      \exp_args:Ne \pdf_object_ref:n { \l_pdf_current_structure_destination_
                      /FitR ~
                        @pdf_ #2 _llx ~ @pdf_ #2 _lly ~
                        @xpos ~ @ypos
                    ]
                  }
              }
          }
        \tex_vss:D
      }
    \__kernel_kern:n { -#1 }
  }
```

And now we redefine the destination command:

```
\cs_set_protected:Npn \__pdf_backend_structure_destination:nnnn #1#2#3#4
  {
    \exp_args:Ne \__pdf_backend_structure_destination_aux:nnnn
      { \dim_eval:n {#2} } {#1} {#3} {#4}
  }
```

At last the goto link.

```
\cs_set_protected:Npn \__pdf_backend_link_begin_structure_goto:nnw #1#2
  {
    \__pdf_backend_link_begin:n { #1 /Subtype /Link /A << /S /GoTo /D ( #2 ) /SD~@pdf.SDest.
  }
⟨/xdvipdfmx | dvipdfmx⟩
```

(*End definition for __pdf_backend_structure_destination:nn.*)

Now pdftex. We only redefine for version 1.40 revision 24 or later.

```
⟨*pdftex⟩
\bool_lazy_and:nnT
  { \int_compare_p:nNn {\tex_pdftexversion:D } > {139} }
  { \int_compare_p:nNn {\tex_pdftexrevision:D } > {23} }
  {
    \cs_set_protected:Npn \__pdf_backend_structure_destination:nn #1#2
      {
        \tex_pdfdest:D
           name {#1}
           \str_case:nnF {#2}
             {
               { xyz }   { xyz }
               { fit }   { fit }
               { fitb }  { fitb }
               { fitbh } { fitbh }
               { fitbv } { fitbv }
               { fith }  { fith }
               { fitv }  { fitv }
               { fitr }  { fitr }
             }
             { xyz ~ zoom \fp_eval:n { #2 * 10 } }
           \scan_stop:
        \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
          {
            \tex_pdfdest:D
               struct~
               \int_use:c
                 { c__pdf_backend_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structur
               name {#1}
               \str_case:nnF {#2}
                 {
                   { xyz }   { xyz }
                   { fit }   { fit }
                   { fitb }  { fitb }
                   { fitbh } { fitbh }
                   { fitbv } { fitbv }
                   { fith }  { fith }
                   { fitv }  { fitv }
                   { fitr }  { fitr }
                 }
                 { xyz ~ zoom \fp_eval:n { #2 * 10 } }
               \scan_stop:
          }
      }
    \cs_set_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
      {
       \tex_pdfdest:D
        name {#1}
        fitr ~
        width  \dim_eval:n {#2} ~
        height \dim_eval:n {#3} ~
        depth  \dim_eval:n {#4} \scan_stop:
```

```
       \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
         {
           \tex_pdfdest:D
             struct~
             \int_use:c
               { c__pdf_backend_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structure_
             name {#1}
             fitr ~
             width  \dim_eval:n {#2} ~
             height \dim_eval:n {#3} ~
             depth  \dim_eval:n {#4} \scan_stop:
         }
     }
    \cs_set_protected:Npn \__pdf_backend_link_begin_structure_goto:nnw #1#2
      {
        \__pdf_backend_link_begin:nnnw {#1} { goto~struct~name~{#2}~name } {#2}
      }
  }
⟨/pdftex⟩
```

luatex is quite similar to pdftex. Mostly the test for the version is different

```
⟨*luatex⟩
 \int_compare:nNnT {\directlua{tex.print(status.list()["development_id"])} } > {7468}
  {
    \cs_set_protected:Npn \__pdf_backend_structure_destination:nn #1#2
      {
        \tex_pdfextension:D dest
           name {#1}
           \str_case:nnF {#2}
             {
               { xyz }   { xyz }
               { fit }   { fit }
               { fitb }  { fitb }
               { fitbh } { fitbh }
               { fitbv } { fitbv }
               { fith }  { fith }
               { fitv }  { fitv }
               { fitr }  { fitr }
             }
             { xyz ~ zoom \fp_eval:n { #2 * 10 } }
           \scan_stop:
        \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
          {
            \tex_pdfextension:D dest
               struct~
               \int_use:c
                 { c__pdf_backend_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structur
               name {#1}
               \str_case:nnF {#2}
                 {
                   { xyz }   { xyz }
                   { fit }   { fit }
                   { fitb }  { fitb }
                   { fitbh } { fitbh }
                   { fitbv } { fitbv }
                   { fith }  { fith }
                   { fitv }  { fitv }
                   { fitr }  { fitr }
                 }
                 { xyz ~ zoom \fp_eval:n { #2 * 10 } }
               \scan_stop:
          }
      }
    \cs_set_protected:Npn \__pdf_backend_destination:nnnn #1#2#3#4
      {
       \tex_pdfextension:D dest
        name {#1}
        fitr ~
        width  \dim_eval:n {#2} ~
        height \dim_eval:n {#3} ~
        depth  \dim_eval:n {#4} \scan_stop:
       \exp_args:Ne \pdf_object_if_exist:nT { \l_pdf_current_structure_destination_tl }
         {
           \tex_pdfextension:D dest
             struct~
             \int_use:c
               { c__pdf_backend_object_ \exp_args:Ne \tl_to_str:n {\l_pdf_current_structure_
             name {#1}
             fitr ~
             width  \dim_eval:n {#2} ~
             height \dim_eval:n {#3} ~
             depth  \dim_eval:n {#4} \scan_stop:
         }
     }
    \cs_set_protected:Npn \__pdf_backend_link_begin_structure_goto:nnw #1#2
      {
        \__pdf_backend_link_begin:nnnw {#1} { goto~struct~name~{#2}~name } {#2}
      }
  }
⟨/luatex⟩
```

1.10 Settings for regression tests

When doing pdf based regression tests some meta data in the pdf should have fixed values to get identical pdf's. We define here the backend dependant part. The main command is then in l3pdfmeta

```
⟨*drivers⟩
\cs_new_protected:Npn \__pdf_backend_set_regression_data:
  {
    \sys_gset_rand_seed:n{1000}
    \pdfmanagement_add:nnn{Info}{Creator}{(TeX)}
⟨/drivers⟩
⟨*dvips⟩
    \__kernel_backend_literal:e{!~<</DocumentUUID~(DocumentUUID)>>~setpagedevice}
    \__kernel_backend_literal:e{!~<</InstanceUUID~(InstanceUUID)>>~setpagedevice}
⟨/dvips⟩
⟨*dvipdfmx⟩
    \pdfmanagement_add:nnn{Info}{Producer}{(dvipdfmx)}
    \__kernel_backend_literal:e
      {pdf:trailerid [~
      <00112233445566778899aabbccddeeff>~
      <00112233445566778899aabbccddeeff>~
          ]}
⟨/dvipdfmx⟩
⟨*xdvipdfmx⟩
    \pdfmanagement_add:nnn{Info}{Producer}{(xetex)}
    \__kernel_backend_literal:e
      {pdf:trailerid [~
      <00112233445566778899aabbccddeeff>~
      <00112233445566778899aabbccddeeff>~
          ]}
⟨/xdvipdfmx⟩
⟨*pdftex⟩
   \pdfmanagement_add:nnn{Info}{Producer}{(pdfTeX)}
   \tex_pdfsuppressptexinfo:D 7 \scan_stop:
   \pdftrailerid{2350CAD05F8A7AF0AA4058486855344F}
⟨/pdftex⟩
⟨*luatex⟩
   \pdfmanagement_add:nnn{Info}{Producer}{(LuaTeX)}
   \tex_pdfvariable:D suppressoptionalinfo 7\relax
   \tex_pdfvariable:D trailerid
     {[~
       <2350CAD05F8A7AF0AA4058486855344F>~
       <2350CAD05F8A7AF0AA4058486855344F>~
     ]}
⟨/luatex⟩
⟨*drivers⟩
    \pdfmanagement_add:nnn{Info}{CreationDate}{(D:20010101205959-00’00’)}
    \pdfmanagement_add:nnn{Info}{ModDate}{(D:20010101205959-00’00’)}
    \AddToDocumentProperties[document]{creationdate}{D:20010101205959-00’00’}
    \AddToDocumentProperties[document]{moddate}{D:20010101205959-00’00’}
    \AddToDocumentProperties[hyperref]{pdfmetadate}{D:20010101205959-00’00’}
    \AddToDocumentProperties[hyperref]{pdfdate}{D:20010101205959-00’00’}
    \AddToDocumentProperties[hyperref]{pdfinstanceid}{uuid:0a57c455-157a-4141-8c19-6237d832f
   }
⟨/drivers⟩
```

1.11 Uncompressed metadata object stream

The xmp metadata should be written “uncompressed” to pdf. It is not quite clear what exactly that means. Probably it only means that there should be no /Filter key in the stream, but packages like pdfx and hyperref try to suppress object compression too, so we add support for it too. With luatex this is possible by using the uncompressed key word. With pdftex one can change locally the compresslevel. (x)dvipdfmx does it automatically and doesn’t need some special command. No solution is known for the dvips route. We need it only once, so we make it special and probably no public interface is needed. It writes an unnamed object so should be referenced directly with \pdf_object_ref_last:

```
⟨*luatex⟩
\cs_new_protected:Npn \__pdf_backend_metadata_stream:n #1
  {
    \tex_immediate:D \tex_pdfextension:D obj ~uncompressed~
```

```
      \__pdf_backend_object_write:nn {stream} {{/Type~/Metadata~/Subtype~/XML}{#1}}
  }
⟨/luatex⟩
⟨*pdftex⟩
\cs_new_protected:Npn \__pdf_backend_metadata_stream:n #1
  {
    \group_begin:
     \tex_pdfcompresslevel:D  0 \scan_stop:
     \tex_immediate:D \tex_pdfobj:D
     \__pdf_backend_object_write:nn {stream} {{/Type~/Metadata~/Subtype~/XML}{#1}}
    \group_end:
  }
⟨/pdftex⟩
⟨*xdvipdfmx | dvipdfmx | dvips | dvisvgm⟩
\cs_new_protected:Npn \__pdf_backend_metadata_stream:n #1
  {
    \pdf_object_unnamed_write:nn {stream}{{/Type~/Metadata~/Subtype~/XML}{#1}}
  }
⟨/xdvipdfmx | dvipdfmx | dvips | dvisvgm⟩
```

1.12 lua code for lualatex

```
⟨*lua⟩
ltx= ltx or {}
ltx.__pdf      = ltx.__pdf or {}
ltx.__pdf.Page = ltx.__pdf.Page or {}
ltx.__pdf.Page.dflt = ltx.__pdf.Page.dflt or {}
ltx.__pdf.Page.Resources = ltx.__pdf.Resources or {}
ltx.__pdf.Page.Resources.Properties = ltx.__pdf.Page.Resources.Properties or {}
ltx.__pdf.Page.Resources.List={"ExtGState","ColorSpace","Pattern","Shading"}
ltx.__pdf.object = ltx.__pdf.object or {}

ltx.pdf= ltx.pdf or {} -- for "public" functions

local __pdf = ltx.__pdf
local pdf = pdf

local function __pdf_backend_Page_gput (name,value)
 __pdf.Page.dflt[name]=value
end

local function __pdf_backend_Page_gremove (name)
 __pdf.Page.dflt[name]=nil
end

local function __pdf_backend_Page_gclear ()
 __pdf.Page.dflt={}
end

local function __pdf_backend_ThisPage_gput (page,name,value)
 __pdf.Page[page] = __pdf.Page[page] or {}
 __pdf.Page[page][name]=value
end

```

```
local function __pdf_backend_ThisPage_gpush (page)
 local token=""
 local t = {}
 local tkeys= {}
 for name,value in pairs(__pdf.Page.dflt) do
   t[name]=value
 end
 if __pdf.Page[page] then
  for name,value in pairs(__pdf.Page[page]) do
   t[name] = value
  end
 end
 -- sort the table to get reliable test files.
 for name,value in pairs(t) do
  table.insert(tkeys,name)
 end
 table.sort(tkeys)
 for _,name in ipairs(tkeys) do
   token = token .. "/"..name.." "..t[name]
 end
 return token
end

function ltx.__pdf.backend_ThisPage_gput (page,name,value) -- tex.count["g_shipout_readonly_
 __pdf_backend_ThisPage_gput (page,name,value)
end

function ltx.__pdf.backend_ThisPage_gpush (page)
  pdf.setpageattributes(__pdf_backend_ThisPage_gpush (page))
end

function ltx.__pdf.backend_Page_gput (name,value)
  __pdf_backend_Page_gput (name,value)
end

function ltx.__pdf.backend_Page_gremove (name)
  __pdf_backend_Page_gremove (name)
end

function ltx.__pdf.backend_Page_gclear ()
  __pdf_backend_Page_gclear ()
end


local Properties  = ltx.__pdf.Page.Resources.Properties
local ResourceList= ltx.__pdf.Page.Resources.List
local function __pdf_backend_PageResources_gpush (page)
 local token=""
 if Properties[page] then
-- we sort the table, so that the pdf test works
  local t = {}
  for name,value in pairs  (Properties[page]) do
   table.insert (t,name)
  end
  table.sort (t)
  for _,name in ipairs(t) do
   token = token .. "/"..name.." ".. Properties[page][name]
  end
  token = "/Properties <<"..token..">>"
 end
  for i,name in ipairs(ResourceList) do
   if ltx.__pdf.Page.Resources[name] then
   token = token .. "/"..name.." "..ltx.pdf.object_ref("__pdf/Page/Resources/"..name)
   end
  end
 return token
end

-- the function is public, as I probably need it in tagpdf too ...
function ltx.pdf.Page_Resources_Properties_gput (page,name,value) -- tex.count["g_shipout_re
 Properties[page] = Properties[page] or {}
 Properties[page][name]=value
 pdf.setpageresources(__pdf_backend_PageResources_gpush (page))
end

function ltx.pdf.Page_Resources_gpush(page)
 pdf.setpageresources(__pdf_backend_PageResources_gpush (page))
end

function ltx.pdf.object_ref (objname)
 if ltx.__pdf.object[objname] then
  local ref= ltx.__pdf.object[objname]
  return ref
 else
  return "false"
 end
end
⟨/lua⟩
```

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.