The l3backend-pdf-extra package
Additional backend PDF features

The LaTeX3 Project[*]

Released XXXX-XX-XX

1 l3backend-pdf-extra Implementation

1.1 Crossreferences

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

```
⟨@@=pdf⟩
⟨*drivers⟩
\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⟩
```

```
⟨*dvipdfmx | xdvipdfmx⟩
% avoid that destinations names are optimized.
% is this still needed??
% see https://tug.org/pipermail/dvipdfmx/2019-May/000002.html
    \__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
```

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

```
\box_new:N  \l__pdf_backend_tmpa_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-pdf.lua") }
⟨/luatex⟩
```

1.3 Hooks

1.3.1 Add the "end run" hooks

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

```
⟨*pdftex | luatex⟩
\hook_gput_code:nnn {enddocument/afterlastpage}
  {pdf/endrun}
  {
    \hook_use:n {pdf/management/end_run}
  }
⟨/pdftex | luatex⟩
⟨*dvipdfmx | xdvipdfmx⟩
\hook_gput_code:nnn {shipout/lastpage}
  {pdf/endrun}
  {
    \hook_use:n {pdf/management/end_run}
  }
\hook_gset_rule:nnnn {shipout/lastpage}{pdf}{before}{pdf/endrun}
⟨/dvipdfmx | xdvipdfmx⟩
⟨*dvips⟩
\hook_gput_code:nnn {shipout/lastpage}
  {pdf/endrun}
  {
    \hook_use:n {pdf/management/end_run}
  }
\hook_gset_rule:nnnn {shipout/lastpage}{pdf}{before}{pdf/endrun}
⟨/dvips⟩
```

1.3.2 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
      {
        \hook_use:n {pdf/management/thispage_shipout}
      }
    \g@addto@macro \@kernel@after@shipout@lastpage
      {
        \hook_use:n {pdf/management/lastpage_shipout}
      }
  }
  {
    \hook_gput_code:nnn{shipout/background}{pdf}
      {
        \hook_use:n {pdf/management/thispage_shipout}
      }
    \hook_gput_code:nnn {shipout/lastpage} {pdf}
      {
        \hook_use:n {pdf/management/lastpage_shipout}
      }
  }

⟨/drivers⟩
```

1.4 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.)

1.5 “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
    %\zref@labelbylist {l3pdf\int_use:N\g__pdf_backend_resourceid_int} {l3pdf}
    %\ref_label:en{l3pdf\int_use:N\g__pdf_backend_resourceid_int}{abspage}
     \__pdf_backend_ref_label:en { l3pdf\int_use:N\g__pdf_backend_resourceid_int }{abspage}
    \tl_set:Nx \l__pdf_tmpa_tl
      {
        %\zref@extractdefault
%          {l3pdf\int_use:N\g__pdf_backend_resourceid_int}
%          {pdf@abspage}
%          {0}
%        \ref_value:en{l3pdf\int_use:N\g__pdf_backend_resourceid_int}{abspage}
        \__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 { \pdfdict_name:n { g__pdf_Core/Page } }
    \prop_if_exist:cT  { \pdfdict_name:n { g__pdf_Core/backend_Page#1 } }
      {
        \prop_map_inline:cn { \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.*)

1.6 "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.

__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

This pushes out the objects. It is a no-op with xdvipdfmx and dvips.

__pdf_backend_PageResources_obj_gpush:

```
% backend commands the command to fill the register
% and to push the values.
%
% 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⟩
% pdftex and luatex
⟨*pdftex | luatex⟩
 %create the backend objects:
\clist_map_inline:Nn \c__pdf_backend_PageResources_clist
  {
    \__pdf_backend_object_new:nn {Page/Resources/#1} {dict}
    \cs_if_exist:NT \tex_directlua:D
      {
        \tex_directlua:D
          {
            ltx.__pdf.object["Page/Resources/#1"]
            =
            "\__pdf_backend_object_ref:n{Page/Resources/#1}"
          }
      }
   }
⟨/pdftex | luatex⟩
⟨*luatex⟩
%values are only stored in a prop and will be output at end document.
\cs_new_protected:Npn \__pdf_backend_PageResources_gput:nnn #1 #2 #3
  {
    \pdfdict_gput:nnn {g__pdf_Core/Page/Resources/#1} { #2 }{ #3 }
    % luatex must also trigger the lua side
    \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⟩
⟨*pdftex | luatex⟩
%code for end of document code
\cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush:
  {
    \clist_map_inline:Nn \c__pdf_backend_PageResources_clist
      {
        \prop_if_empty:cF
          { \pdfdict_name:n { g__pdf_Core/Page/Resources/##1} }
          {
            \__pdf_backend_object_write:nx
              { Page/Resources/##1 }
              { \pdfdict_use:n { g__pdf_Core/Page/Resources/##1} }
          }
     }
  }
⟨/pdftex | luatex⟩
% xdvipdfmx
% \special{pdf:pageresources<<#1>>} doesn’t work correctly with object names ...
% https://tug.org/pipermail/dvipdfmx/2019-August/000021.html,
% so we use \special{pdf:put @resources}
% this must be issued on every page!
⟨*dvipdfmx | xdvipdfmx⟩
%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.
⟨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_backend_object_new:nn   { Page/Resources/#1 } { dict }
    \hook_gput_code:nnn{shipout/firstpage}{pdf}{\__pdf_backend_object_write:nn { Page/Resour
  }
\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
   \prop_gput:cnn { \pdfdict_name:n { g__pdf_Core/Page/Resources/#1} } { #2 }{ #3 }
   %objects are not filled with \pdf_object_write as this is not additive!
    \__pdf_backend:x
      {
        put~\__pdf_backend_object_ref:n {Page/Resources/#1}<</#2~#3>>
      }
  }

\cs_new_protected:Npn \__pdf_backend_PageResources_obj_gpush: {}
⟨/dvipdfmx | xdvipdfmx⟩
⟨*dvips⟩
% dvips unneeded, or no-op
\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⟩
⟨*dvisvgm⟩
% dvips unneeded, or no-op
\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:`.)

1.6.1 Page resources /Properties + BDC operators

__pdf_backend_bdc:nn
__pdf_backend_bdc_obj:nn
__pdf_backend_bdc_obj:n
__pdf_backend_bmc:n
__pdf_backend_emc:
__pdf_backend_PageResources_gpush:n

`\__pdf_backend_bdc:nn`, `\__pdf_backend_bdc_obj:nn`, `\__pdf_backend_bdc_obj: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_bdc_obj: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_bdc_obj: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_bdc_obj:nn #1 #2 % #1 eg. Span, #2: object name
  {}
\cs_set_protected:Npn \__pdf_backend_bdc_obj: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_bdc_obj: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_bdc_obj: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_now:nn { dict }{ #2 }
    \__pdf_backend_bdc_obj: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 { \pdfdict_name:n { g__pdf_Core/Page/Resources/##1} }
          {
            \__kernel_backend_literal:x
              {
                pdf:put~@resources~
                  <</##1~\__pdf_backend_object_ref:n {Page/Resources/##1}>>
              }
          }
      }
  }
⟨/dvipdfmx | xdvipdfmx⟩
% luatex + pdftex
⟨*luatex⟩
\cs_set_protected:Npn \__pdf_backend_bdc_obj: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_bdc_obj: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_now:nn { dict } { #2 }
    \__pdf_backend_bdc_obj: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_bdc_obj: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 } }
      }
      {
        %\zref@labelbylist
%          { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
%          { l3pdf }
%        \ref_label:en{l3pdf\int_use:N\g__pdf_backend_resourceid_int}{abspage}
        \__pdf_backend_ref_label:en{l3pdf\int_use:N\g__pdf_backend_resourceid_int}{abspage}
        \tl_set:Nx \l__pdf_tmpa_tl
          {
            %\zref@extractdefault
%              { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
%              {pdf@abspage}
%              {0}
            %\ref_value:en{l3pdf\int_use:N\g__pdf_backend_resourceid_int}{abspage}
            \__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_bdc_obj: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: }
      }
      {
        %\zref@labelbylist
%          { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
%          { l3pdf }
       %\ref_label:en{l3pdf\int_use:N\g__pdf_backend_resourceid_int}{abspage}
        \__pdf_backend_ref_label:en{l3pdf\int_use:N\g__pdf_backend_resourceid_int}{abspage}
        \tl_set:Nx \l__pdf_tmpa_tl
          {
            %\zref@extractdefault
%              { l3pdf\int_use:N\g__pdf_backend_resourceid_int }
%              {pdf@abspage}
%              {0}
```

```
         %   \ref_value:en{l3pdf\int_use:N\g__pdf_backend_resourceid_int}{abspage}
            \__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_now:nn { dict } { #2 }
     \__pdf_backend_bdc_obj: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
      { \pdfdict_name:n { g__pdf_Core/Page/Resources/#1} }
      {
        \pdfdict_item:ne { #1 }{ \pdf_object_ref:n {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
           { \pdfdict_name:n { g__pdf_Core/backend_Page#1/Resources/Properties } }
           {
             /Properties~
               <<
                 \prop_map_function:cN
                   { \pdfdict_name:n { g__pdf_Core/backend_Page#1/Resources/Properties } }
                   \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.*)

1.7 “Catalog” & subdirectories (pdfcatalog)

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

1.7.1 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 we need some code to push the tree if files have been added. dvips wants code for every file and then creates the Name tree automatically.

```
% pdflatex
⟨*pdftex⟩
\cs_new_protected:Npn \__pdf_backend_NamesEmbeddedFiles_gpush:n #1 %array content
  {
     \pdf_object_now:nn {dict} {/Names [#1] }
      \tex_pdfnames:D {/EmbeddedFiles~\pdf_object_last:}
  }
⟨/pdftex⟩
⟨*luatex⟩
\cs_new_protected:Npn \__pdf_backend_NamesEmbeddedFiles_gpush:n #1 %array content
  {
    \pdf_object_now:nn {dict} {/Names [#1] }
    \tex_pdfextension:D~names~{/EmbeddedFiles~\pdf_object_last: }
  }
⟨/luatex⟩
⟨*dvipdfmx | xdvipdfmx⟩
\cs_new_protected:Npn \__pdf_backend_NamesEmbeddedFiles_gpush:n #1 %array content
  {
    \pdf_object_now:nn {dict} { /Names [#1] }
    %n or x?
    \__pdf_backend:x {put~@names~<</EmbeddedFiles~\pdf_object_last: >>}
  }
⟨/dvipdfmx | xdvipdfmx⟩

%dvips: noop
```

```
⟨*dvips⟩
\cs_new_protected:Npn \__pdf_backend_NamesEmbeddedFiles_gpush:n #1 {}
⟨/dvips⟩
%dvisvgm: noop
⟨*dvisvgm⟩
\cs_new_protected:Npn \__pdf_backend_NamesEmbeddedFiles_gpush:n #1 {}
⟨/dvisvgm⟩

```

Names in the EmbeddedFiles name tree must sorted alphabetically, so we need commands to create this names. And we need a sequence to store the names and the objects. We use the prefix l3ef, and we assume that at most 9999 files will be used.

\g__pdf_backend_EmbeddedFiles_int

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

__pdf_backend_EmbeddedFiles_name:

```
⟨*drivers⟩
\int_new:N \g__pdf_backend_EmbeddedFiles_int
\cs_new:Npn \__pdf_backend_EmbeddedFiles_name:
 {
   (
    l3ef
    \int_compare:nNnT {\g__pdf_backend_EmbeddedFiles_int} < {10}
     {0}
    \int_compare:nNnT {\g__pdf_backend_EmbeddedFiles_int} < {100}
     {0}
    \int_compare:nNnT {\g__pdf_backend_EmbeddedFiles_int} < {1000}
     {0}
    \int_use:N \g__pdf_backend_EmbeddedFiles_int
   )
 }
⟨/drivers⟩
```

(*End definition for* `\__pdf_backend_EmbeddedFiles_name:`.)

\g__pdf_backend_EmbeddedFiles_seq
\g__pdf_backend_EmbeddedFiles_named_prop

The sequence will hold the content of the array that is pushed out at then end (not with dvips), the prop holds the obj names-names relation.

(*End definition for* `\g__pdf_backend_EmbeddedFiles_seq` *and* `\g__pdf_backend_EmbeddedFiles_named_prop`.)

```
⟨*drivers⟩
\seq_new:N \g__pdf_backend_EmbeddedFiles_seq
\prop_new:N \g__pdf_backend_EmbeddedFiles_named_prop
⟨/drivers⟩
```

__pdf_backend_NamesEmbeddedFiles_add:n

This command saves an object reference of a filespec dictionary in the EmbeddedFiles name tree. We define a prop to store the relation between object name and name in the name tree.

```
⟨*pdftex | luatex | dvipdfmx | xdvipdfmx⟩
\cs_new_protected:Npn  \__pdf_backend_NamesEmbeddedFiles_add:n #1
    %#1 object ref
    {
      \int_gincr:N \g__pdf_backend_EmbeddedFiles_int
      \prop_gput:Nnx \g__pdf_backend_EmbeddedFiles_named_prop
        { #1 }
        { \__pdf_backend_EmbeddedFiles_name: }
      \seq_gput_right:Nx \g__pdf_backend_EmbeddedFiles_seq
        { \__pdf_backend_EmbeddedFiles_name: \c_space_tl #1 }
    }

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

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

1.7.2 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 {\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 { \pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties } } }
     >
     { 0 }
     {
      /Properties~
       <<
        \pdfdict_use:n { g__pdf_Core/Xform/Resources/Properties }
       >>
     }

    \prop_if_empty:cF
     { \pdfdict_name:n { g__pdf_Core/Page/Resources/ExtGState } }
     {
      /ExtGState~ \pdf_object_ref:n { Page/Resources/ExtGState }
     }
    \prop_if_empty:cF
     { \pdfdict_name:n { g__pdf_Core/Page/Resources/Pattern } }
     {
      /Pattern~ \pdf_object_ref:n { Page/Resources/Pattern }
     }
    \prop_if_empty:cF
     { \pdfdict_name:n { g__pdf_Core/Page/Resources/Shading } }
     {
      /Shading~ \pdf_object_ref:n { Page/Resources/Shading }
     }
    \prop_if_empty:cF
     { \pdfdict_name:n { g__pdf_Core/Page/Resources/ColorSpace } }
     {
      /ColorSpace~ \pdf_object_ref:n { 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 { \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 { \pdfdict_name:n { g__pdf_Core/Xform/Resources/Properties } }}
          >
          { 0 }
          {
            /Properties~
              <<
                \pdfdict_use:n { g__pdf_Core/Xform/Resources/Properties }
              >>
          }
        \prop_if_empty:cF
          { \pdfdict_name:n { g__pdf_Core/Page/Resources/ExtGState } }
          {
            /ExtGState~ \pdf_object_ref:n { Page/Resources/ExtGState }
          }
        \prop_if_empty:cF
          { \pdfdict_name:n { g__pdf_Core/Page/Resources/Pattern } }
          {
            /Pattern~ \pdf_object_ref:n { Page/Resources/Pattern }
          }
        \prop_if_empty:cF
          { \pdfdict_name:n { g__pdf_Core/Page/Resources/Shading } }
          {
            /Shading~ \pdf_object_ref:n { Page/Resources/Shading }
          }
        \prop_if_empty:cF
          { \pdfdict_name:n { g__pdf_Core/Page/Resources/ColorSpace } }
          {
            /ColorSpace~ \pdf_object_ref:n { 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  ~ \pdf_xform_wd:n { #1 }
                \c_space_tl height ~ \pdf_xform_ht:n { #1 }
                \c_space_tl depth  ~ \pdf_xform_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 { Page/Resources/ExtGState }
                  >>
              }
            \__pdf_backend:x
              {
                put~ @resources ~
                <<
                  /Pattern~ \pdf_object_ref:n { Page/Resources/Pattern }
                >>
              }
            \__pdf_backend:x
              {
                put~ @resources ~
                <<
                  /Shading~ \pdf_object_ref:n { Page/Resources/Shading }
                >>
```

```
               }
             \__pdf_backend:x
               {
                 put~ @resources ~
                 <<
                   /ColorSpace~
                   \pdf_object_ref:n { 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 { \pdf_xform_wd:n { #1 } }
       \box_set_ht:Nn  \l__pdf_backend_tmpa_box { \pdf_xform_ht:n { #1 } }
       \box_set_dp:Nn  \l__pdf_backend_tmpa_box { \pdf_xform_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⟩
⟨*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 \pdf_xform_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.)

1.8 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("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.