The pdfmanagement-firstaid package – temporary patches and package replacements LaTeX PDF management testphase bundle

The LaTeX Project[*]

Version 0.96l, released 2024-09-13

1 pdfmanagement-firstaid documentation

This code is temporary! It tries to patch commands of other packages or even replace package which are incompatible with the pdfmanagement, to remove clashes and test if everything works as expected. This code should disappear when packages adapt to the central interfaces.

The package contains an number of sections for various packages. Every section can be disabled in (the first) `\DocumentMetadata` with `debug={firstaidoff={name1,name2,...},...}`.

```
⟨*package⟩
\ProvidesExplPackage{pdfmanagement-firstaid}{2024-09-13}{0.96l}
  {LaTeX PDF management testphase bundle / firstaid-patches}

⟨@@=pdfmanagement⟩
\clist_map_inline:nn
  {pgf,xmp,xcolor,color,output,colorspace,luacolor}
  {
    \bool_new:c       { g__pdfmanagement_firstaid_#1_bool }
    \bool_gset_true:c { g__pdfmanagement_firstaid_#1_bool }
  }
\clist_map_inline:Nn \g__pdfmanagement_firstaidoff_clist
  {
    \bool_if_exist:cT { g__pdfmanagement_firstaid_#1_bool }
      {
        \bool_gset_false:c { g__pdfmanagement_firstaid_#1_bool }
      }
  }
\msg_new:nnn  { pdfmanagement } { firstaid }
              { loading~pdfmanagement~firstaid~code~for~#1 }
\msg_new:nnn  { pdfmanagement } { firstaid-changed }
              { package~#1~has~changed.~Check~if~patch~is~still~valid! }
\msg_new:nnn  { pdfmanagement } { firstaid-disabled }
              { The~loading~of~package~#1~is~disabled.\\
                It~is~not~compatible~with~the~PDF~management. }
\msg_new:nnn  { pdfmanagement } { firstaid-too-old }
```

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

```
              { Package~#1~is~too~old~and~not~compatible.\\
                Get~at~least~version~#2.}

```

1.1 color

color is not incompatible, but the new hyperref driver makes use of l3color to set the colors. It is therefore necessary to patch some internal color commands, so that colors defined with its `\definecolor` command are known to l3color and so hyperref. This only supports the color models from l3color (which covers all standard model of the color package). The `named` model is mapped to `\color_set:nn`.

This patch serves also as test to check if this change can be safely added to color later.

```
\bool_if:NT \g__pdfmanagement_firstaid_color_bool
  {
    \declare@file@substitution{color.sty}{color-ltx.sty}
  }
```

1.2 xcolor

xcolor is not incompatible, but the new hyperref driver makes use of l3color to set the colors. It is therefore necessary to patch xcolor, so that colors defined with its `\definecolor` command are known to l3color and so hyperref. This only supports the color model from l3color. Colors defined with the models `cmy` and `tHsb` are silently ignored.

The `named` model is mapped to `\color_set:nn`.

```
\bool_if:NT \g__pdfmanagement_firstaid_xcolor_bool
  {
    \AddToHook
       {
         package/xcolor/after
       }
       {\RequirePackage{xcolor-patches-tmp-ltx}}
```

The patch must before color definitions are loaded, which will happen in hooks in a newer xcolor:

```
    \DeclareHookRule{package/xcolor/after}{pdfmanagement-firstaid}{before}{xcolor}
  }
```

1.3 luacolor

The luacolor package doesn't take colors from l3color into account. We add a fix, but only for pdf mode. luacolor can disable the code by clearing the hook if needed.

```
\bool_lazy_all:nT
  {
    {\sys_if_engine_luatex_p:}
    {\g__pdfmanagement_firstaid_luacolor_bool}
    {\sys_if_output_pdf_p:}
  }
  {
    \AddToHook{package/luacolor/after}
     {
```

```
     \cs_set_protected:Npn \__color_backend_select:nn #1#2
      {
        \tl_set:Nn \l__color_backend_fill_tl {#1}
        \tl_set:Nn \l__color_backend_stroke_tl {#2}
        \LuaCol@setattribute\LuaCol@Attribute
          {
            \directlua
              {
                oberdiek.luacolor.get("\luaescapestring{#1~#2}")
              }
          }
       }
     \cs_set_protected:Npn \__color_backend_fill:n #1
       {
        \tl_set:Nn \l__color_backend_fill_tl {#1}
        \LuaCol@setattribute\LuaCol@Attribute
          {
            \directlua
              {
                oberdiek.luacolor.get("\luaescapestring{#1}")
              }
          }
       }
     \cs_set_protected:Npn \__color_backend_stroke:n #1
       {
         \tl_set:Nn \l__color_backend_stroke_tl {#1}
         \LuaCol@setattribute\LuaCol@Attribute
           {
             \directlua
               {
                 oberdiek.luacolor.get("\luaescapestring{#1}")
               }
           }
       }
     \cs_set_protected:Npn \__color_backend_reset:  {}
     \cs_set_eq:NN \__color_backend_fill_reset:   \__color_backend_reset:
     \cs_set_eq:NN \__color_backend_stroke_reset: \__color_backend_reset:
   }
}
```

1.4 pgf

In pgf, resource management is set up in the file `pgfutil-common.tex`. This then provides three functions for adding to the resources, all of which are objects:

- `\pgfutil@addpdfresource@extgs`: Extended graphics state
- `\pgfutil@addpdfresource@colorspaces`: Color spaces
- `\pgfutil@addpdfresource@patterns`: Patterns

These resource dictionaries are used by adding entries in a cumulative sense; the macro layer deals with ensuring that each entry is only given once. Note that the objects themselves must be given only once for each page.

To support these functions, there are a series of set-up macros which install these resources. That has to take place for every page: the exact route therefore depends on the driver.

For the pdfmanagement project we need to avoid that pgf interferes in ExtGState, ColorSpace and Pattern (Shadings are added to the xform resources and so probably unproblematic for now). The actual patch is in a file hook guarded by the boolean, the rest of the code is always defined.

```

\bool_if:NT \g__pdfmanagement_firstaid_pgf_bool
  {
    \msg_info:nnn{pdfmanagement }{firstaid}{pgf}
    \AddToHook
      {
        file/pgfrcs.sty/after
      }
      {
        \cs_set_eq:NN
          \__pdfmanagement_pgfori_pgfutil@setuppdfresources
          \pgfutil@setuppdfresources
        \def\pgfutil@setuppdfresources
          {
            \pdfmanagement_if_active:TF
              {
                \__pdfmanagement_pgf_sys_setuppdfresources_plain:
              }
              {
                \__pdfmanagement_pgfori_pgfutil@setuppdfresources
              }
         }
     }
  }
%\def\pgfutil@addpdfresource@extgs#1{\pgf@sys@addpdfresource@extgs@plain{#1}}
%\def\pgfutil@addpdfresource@colorspaces#1{\pgf@sys@addpdfresource@colorspaces@plain{#1}}
%\def\pgfutil@addpdfresource@patterns#1{\pgf@sys@addpdfresource@patterns@plain{#1}}
%\def\pgfutil@setuppdfresources{\pgf@sys@setuppdfresources@plain}
% \pgf@sys@pdf@possible@resources %used in xform
%Trying to patch pgf ..
\cs_new_protected:Npn \__pdfmanagement_pgf_sys_setuppdfresources_plain:
  {
   %objects are already created ...
    \def\pgf@sys@pdf@possible@resources
      {
        /ColorSpace~\pdf_object_ref:n {__pdf/Page/Resources/ColorSpace}
        /Pattern   ~\pdf_object_ref:n {__pdf/Page/Resources/Pattern}
        /ExtGState ~\pdf_object_ref:n {__pdf/Page/Resources/ExtGState}
      }
    \let\pgf@sys@pdf@check@resources=\relax%
    %not sure if needed, but perhaps the lists are used somewhere else ...
    \let\pgf@sys@pgf@resource@list@extgs=\pgfutil@empty%
    \let\pgf@sys@pgf@resource@list@patterns=\pgfutil@empty%
    \let\pgf@sys@pgf@resource@list@colorspaces=\pgfutil@empty%
    % the commands to add page resources
    \def\pgf@sys@addpdfresource@extgs@plain##1
```

```
      {
        %\exp_after:wN %for transparent which passes a command
          \__pdfmanagement_patch_pgfextgs:w ##1\q_stop
      }
    \def\pgf@sys@addpdfresource@patterns@plain##1
      {
        \__pdfmanagement_patch_pgfpatterns:w ##1\q_stop
      }
   \def\pgf@sys@addpdfresource@colorspaces@plain##1
      {
        \__pdfmanagement_patch_pgfcolorspaces:w ##1\q_stop
      }
  }

%\AtEndPreamble{\pgfutil@setuppdfresources}
% helper commands as pgf doesn’t pass resources as two arguments
% code to add to the resources existing stuff in the format "/name value":
\cs_new:Npn \__pdfmanagement_split_dict_entry_aux:NNw #1 #2 /#3~#4\q_stop
  {
    \tl_set:Nn #1 {#3}
    \tl_set:Nn #2 {#4}
  }

\cs_generate_variant:Nn \tl_trim_spaces:n{V}
\cs_generate_variant:Nn \pdfmanagement_add:nnn {nee}
\cs_new:Npn \__pdfmanagement_patch_pgfextgs:w  #1/#2<<#3>>#4\q_stop
  {
    \tl_set:Ne\l_tmpa_tl{#2}
      \pdfmanagement_add:nee
        {Page/Resources/ExtGState}{\tl_trim_spaces:V\l_tmpa_tl}{<<#3 #4>>}
  }
\cs_new:Npn \__pdfmanagement_patch_pgfpatterns:w  #1/#2\space#3\q_stop
  {
     \pdfmanagement_add:nee
        {Page/Resources/Pattern}{\tl_trim_spaces:n{#2}}{#3}
  }
\cs_new:Npn \__pdfmanagement_patch_pgfcolorspaces:w  #1/#2[#3]#4\q_stop
  {
    \pdfmanagement_add:nee
      {Page/Resources/ColorSpace}{\tl_trim_spaces:n{#2}}{[#3]}
  }

```

1.5 xmp

This handles the new xmp code.

```
\bool_if:NT \g__pdfmanagement_firstaid_xmp_bool
 {
   \disable@package@load{hyperxmp}{\msg_warning:nnn{pdfmanagement}{firstaid-disabled}{hyperxm
   \disable@package@load{pdfx}    {\msg_warning:nnn{pdfmanagement}{firstaid-disabled}{pdfx}}
   \AddToHook{package/doclicense/after}
    {
      \AtBeginDocument
        {
          \hypersetup
            {
              pdfcopyright  = {\doclicenseLongTextForHyperref},
              pdflicenseurl = {\doclicenseURL},
            }
        }
    }
 }
⟨/package⟩
```

1.6 colorspace

This is rather difficult as no real places to inject patches at first a try to avoid that its ExtGState is missing: it can not be avoided to recreate the objects (and so to get duplicates) as colorspace uses temporary macros whose contents is lost.

```
⟨*package⟩
\bool_if:NT \g__pdfmanagement_firstaid_colorspace_bool
 {
    \AddToHook
      {
       package/colorspace/after
      }
      {\RequirePackage{colorspace-patches-tmp-ltx}}
 }
⟨/package⟩
```

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.