

The **l3pdfmanagement** module

Managing central PDF resources

L^AT_EX PDF management testphase bundle

The L^AT_EX Project*

Version 0.95g, released 2021-07-21

1 **l3pdfmanagement** documentation

When creating a pdf a number of objects, dictionaries and entries to central “core” dictionaries must be created.

The commands in this module offer interfaces to this core PDF dictionaries. They unify a number of primitives like the pdftex registers and commands `\pdfcatalog`, `\pdfpageattr`, `\pdfpagesattr`, `\pdfinfo`, `\pdfpageresources` and similar commands of the other backends in a backend independent way.

The supported backends are pdf_latex, lua_latex, (x)dvipdfmx (lat_lex, xel_lat_lex and—starting in texlive 2021—lua_lat_lex) and dvips with ps2pdf (not completely yet). dvips with distiller could work too but is untested.

That the interfaces are backend independent doesn’t mean that the results and even the compilation behavior is identical. The backends are too different to allow this. Some backends expand arguments e.g. in a `\special` while other don’t. Some backends can insert a resource at the first compilation, while another uses the aux-file and a label and so needs at least two. Some backends create and manage resources automatically which must be managed manually by other backends.

The dictionaries and resources handled by this module are inserted only once in a PDF or only once per page. Examples are the Catalog dictionary, the Info dictionary, the page resources. For these dictionaries and resources management by the L^AT_EX kernel is necessary to avoid that packages overwrite settings from other packages which would lead to clashes and incompatibilities. It is therefore necessary that *all* packages which want to add content to these dictionaries and resources use the interface provided by this module.

As these dictionaries and resources are so central for the PDF format values to these dictionaries are always added globally. Through the interface values can be added (and in many cases also removed) by users and packages, but the actually writing of the dictionary entries and resources to the PDF is handled by the kernel code.

The interface uses as main name to address the resources *Paths* which follow the names and structure described in the PDF reference. This should make it easy to identify the names needed to insert a specific PDF resources with the new interfaces. All *Paths* have names starting with an uppercase letter.

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

The following tabular summarize the *Paths* and which pdftex primitive they replace:

Info	<code>\pdfinfo</code>
Catalog & various subdictionaries	<code>\pdfcatalog</code>
Pages	<code>\pdfpagesattr</code>
Page, ThisPage	<code>\pdfpageattr</code>
Page/Resources/ExtGState	<code>\pdfpageresources</code>
Page/Resources/Shading	<code>\pdfpageresources</code>
Page/Resources/Pattern	<code>\pdfpageresources</code>
Page/Resources/ColorSpace	<code>\pdfpageresources</code>

There is no `Page/Resources/Properties` dictionary in the list, because this dictionary is not filled directly, but managed through side effects when setting BDC-marks.

1.1 User Commands

To avoid problems with older documents the resource management of this module is not activated unconditionally. The values are pushed out to the dictionaries only if a boolean has been set to true. The state can be tested with a conditional.

```
\pdfmanagement_if_active_p: ★
\pdfmanagement_if_active:TF ★
```

New: 2020-07-04

This conditional tests if the resource management code is active.

```
\pdfmanagement_add:nnn    \pdfmanagement_add:nnn {\resource path} {\name} {\value}
\pdfmanagement_add:(nnx|nxx)
```

New: 2020-04-06

This function puts `{\name}` `{\value}` in the PDF resource described by the symbolic name `{\resource path}`. Technically it stores it globally in an internal property lists and writes it later into the right PDF dictionary¹. Which values for `{\resource path}` exist is described in the following. `{\name}` should be a PDF Name without the starting slash. Like with all keys used in PDF dictionaries (see the `l3pdfdict` module) the name is escaped with `\str_convert_pdfname:n` when stored. `{\value}` should be a valid PDF value for this Name in the target dictionary.

The code works with all major engines but not necessarily in the same way. Most importantly

- The expansion behaviour of the backends can differ. Some backends expand a value always fully when writing to the PDF, with other backends command names could end as strings in the PDF. So one should neither rely on `{\name}` `{\value}` to be expanded nor not expanded by the backend commands.
- The number of compilations needed can differ between the engines and backends. Some engines have to use labels and the aux-file to setup the dictionaries and so need at least two compilations to put everything in place.

¹Currently all resources are PDF dictionaries, so resource and dictionary mean the same.

<code>\pdfmanagement_show:n</code>	<code>\pdfmanagement_show:n {<resource path>}</code>
------------------------------------	--

New: 2020-04-08

This shows the content of the dictionary targetted by `{<resource path>}` in the log and on the terminal if possible.

It is not reliable for page resources as these are filled at shipout.

It also doesn't show necessarily all the content. For example most backends add automatically entries to the Info dictionary.

<code>\pdfmanagement_remove:nn</code>	<code>\pdfmanagement_remove:nn {<resource path>} {<name>}</code>
---------------------------------------	--

New: 2020-04-07

Removes `/<name>` and its associated `<value>` from the dictionary described with `{<resource path>}`. The removal is global. If `<name>` is not found no change occurs, *i.e.* there is no need to test for the existence of a name before trying to remove it. Values from the special Catalog entries where the values are collected in arrays can't be removed (but should ever a use case appear it could be added).

1.2 Description of the resource pathes

1.2.1 Info: The Info dictionary



If the primitive commands of the engines are used too there will be double entries in the pdf (at least with the backend pdftex and luatex). How pdf viewer handles this is unpredictable.

<code>pdfmanagement: Info</code>	<code>\pdfmanagement_add:nnn {Info} {<name>} {<value>}</code>
----------------------------------	---

Adds `/<name>` and the `<value>` to the Info dictionary. `<name>` should be a PDF name without the leading slash, Like with all keys used in PDF dictionaries (see the `l3pdfdict` module) the name is escaped with `\str_convert_pdfname:n` when stored. `<value>` should be a valid pdf value. Any escaping or (re)encoding must be done explicitly. If a `<name>` is used twice, only the last `<value>` set will be used. The Info dictionary is written at the end of the compilation, so values can be set at any time. The Info dictionary expects utf16be in the strings, so a conversion like this is normally sensible:

```
\str_set_convert:Nnnn \l_tmpa_str { Grüße }{ default } {utf16/string}
\pdfmanagement_add:nnx {Info} {Title}{(\l_tmpa_str)}
```

The entries in Info dictionary are rather special as the engines/backends adds some core entries, and changing or removing these entries is not always possible.

The special entries are

Producer Added by all engines and backends. Removing the entry is only possible with luatex with `\pdfvariable suppressoptionalinfo 128`. Changing is possible with `\pdfmanagement_add:nnn` with the exception of dvips/ps2pdf where the entry is always something like `GPL Ghostscript 9.53.3`.

Creator Added by all engines and backends. Removal only possible in luatex by adding 16 to the bitset. Changing is possible with the management command.

CreationDate Added by all engines and backends. With the exception of dvips/ps2pdf `SOURCE_DATE_EPOCH` is honored. With pdftex it is possible to suppress it with `\pdfinfoomitdate = 1`, and in luatex by adding 32 to the bitset. Changing is possible with the management command and will overwrite an epoch setting.

ModDate Added by all engines and backends with the exception of xdvipdfmx. With the exception of dvips/ps2pdf SOURCE_DATE_EPOCH is honored. Suppressing it is possible in pdftex with \pdfinfoomitdate = 1, and in luatex by adding 64 to the bitset. Changing is possible with the management command.

Trapped Added by pdftex and luatex. Removal only possible in luatex by adding 256 to the bitset. Changing (and adding in the other backends) is possible with the management command.

PTEX.Fullbanner Added by pdftex and luatex. Removal possible in pdftex with \pdfsuppressptexinfo-1, in luatex by adding 512 to the bitset. Changing is not possible.

Title Added by dvips/ps2pdf and set to filename.dvi. Removal is probably not possible, but it can be overwritten with the management command.

1.2.2 Pages: The “Pages” dictionary



As the content of this dictionary is written at the end it will in pdftex and luatex overwrite values added with the primitive commands (e.g. \pdfpagesattr. Package authors should use the management commands instead.

By using this path with the pdfmanagement interface, values can be added to the /Pages object. This replaces for example \pdfpagesattr.

pdfmanagement: Pages \pdfmanagement_add:nnn {Pages} {<name>} {<value>}

Adds /<name> <value> to the /Pages dictionary. It is always stored globally. The content is written to the pdf at the end of the compilation, so values can be added, changed or removed until then. <name> should be a valid pdf name without the leading slash, <value> should be a valid pdf value. Any escaping or (re)encoding must be done explicitly. Some backends expand the value but this should not be relied on. If a <name> is used twice, only the last <value> set will be used.

1.2.3 “Page” and “ThisPage”

pdfmanagement: Page \pdfmanagement_add:nnn {Page} {<name>} {<value>}

New: 2020-04-12

Values added with the path **Page** are added to the page dictionary of the current page and the following pages. The current page means the page on which the command is *executed*. <name> should be a valid pdf name without the leading slash. Typical names used here are e.g. **Rotate** and **CropBox**. <value> should be a valid pdf value. Any escaping or (re)encoding must be done explicitly. Some backends expand the value but this should not be relied on. To avoid problems with the asynchronous page breaking the command should be used after \newpage or in the header. It should not be used in a float, as it will then quite probably be executed on the wrong page. The value is assigned directly and is always stored globally. If a <name> is used twice, only the last <value> set will be used. Names set with \pdfmanagement_add:nnn{ThisPage} will overwrite names set with \pdfmanagement_add:nnn{Page} if there is a clash. Values can be removed again with \pdfmanagement_remove:nn. This replaces \pdfpageattr.

```
pdfmanagement: ThisPage \pdfmanagement_add:nnn {ThisPage} {\langle name \rangle} {\langle value \rangle}
```

New: 2020-04-12

Adds `/\langle name \rangle \langle value \rangle` at *shipout* to the page dictionary of the current page. Current page means here the *shipout* page. It is always stored globally. If `{\langle name \rangle}` has already a value set in the **Page** dictionary it will be overwritten for this page. `\langle name \rangle` should be a valid pdf name without the leading slash, `\langle value \rangle` should be a valid pdf value. Any escaping or (re)encoding must be done explicitly. If a `\langle name \rangle` is used twice, only the last `\langle value \rangle` set will be used. With the engine pdf_latex (at least) a second compilation is needed. Values added to **ThisPage** can not be removed. It is not possible to show the content of this dictionary with `\pdfmanagement_show:n`.

1.2.4 “Page/Resources”: ExtGState, ColorSpace, Shading, Pattern

```
pdfmanagement: Page/Resources/ExtGState \pdfmanagement_add:nnn {Page/Resources/\langle resource \rangle} {\langle name \rangle}
pdfmanagement: Page/Resources/ColorSpace {\langle value \rangle}
pdfmanagement: Page/Resources/Shading
pdfmanagement: Page/Resources/Pattern
```

Updated: 2020-04-10

Adds `/\langle name \rangle \langle value \rangle` to the page resource `\langle resource \rangle`. `\langle resource \rangle` can be **ExtGState**, **ColorSpace**, **Pattern** oder **Shading**. The values are always stored globally. The content is written to the pdf at the end of the compilation, so values can be added until then. `\langle name \rangle` should be a valid pdf name without the leading slash, `\langle value \rangle` should be a valid pdf value for the resource. Any escaping or (re)encoding must be done explicitly. If a `\langle name \rangle` is used twice, only the last `\langle value \rangle` set will be used.

With the dvips backend the command does nothing: these resources are managed by ghostscript or the distiller if e. g. transparency is used.

The resources are added to all pages starting with the first where something has been added to a resources. That means that for example all ExtGState resources are combined in one dictionary object and every page with a ExtGState resource refer to this object ².



The primitive commands (e.g. `\pdfpagemresources`) to set the resources should not be used together with this code as the calls will overwrite each other and values will be lost. This means that currently there are clashes with the packages tikz, transparent and colorspace.

1.2.5 “Catalog” & subdirectories

The catalog is a central dictionary in a PDF with a number of subdictionaries. Entries to the top level of the catalog can be added with

```
\pdfmanagement_add:nnn {Catalog}{\langle Name \rangle}{\langle Value \rangle}
```

Entries to subdictionaries by using in the first argument one of the pathes described later. The entries in the catalog have varying requirements regarding the PDF management. Some entries (like **/Lang**) are simple values where new values should overwrite existing values, other like for example **/OutputIntents** can contain a number of values and can be filled from more than one source. In some cases the values that needs to be added are not at the top-level but in some subsubdictionary or are actually part of an array. To handle the pdf management uses a variety of internal, special handlers.

²This is similar to how pgf handles this resources



In some cases entries are added implicitly. For example entries to the name tree of the `/EmbeddedFiles` key in the `/Names` directory are added with the commands of the `l3pdfxfile` module. This clashes with e.g. the `embedfile` package which should not be used!

Entries at the top level of the catalog The Names in the following tabular are entries that are added to the top level of the catalog.

If $\langle Name \rangle$ gets assigned a value more than once the last one wins. There is no check that the values have the correct type and format. It is up to the user to ensure that the value does what is intended.

The required PDF version is only mentioned if it is larger than 1.5.

Example: `\pdfmanagement_add:nnn {Catalog}{PageMode}{/UseNone}`

Name	Value	Remark
Collection	objref or dict	the content should be build by external packages (see eg <code>embedfile</code>)
DPartRoot	objref or dict	PDF 2.0
Lang	string	e.g. <code>(de-DE)</code>
Legal	objref or dict	
Metadata	objref or stream	
NeedsRendering	boolean	PDF 1.7
OpenAction	array (dest) or dict (action)	
PageLabels	objref or dict	number tree
PageLayout	name	one of <code>/SinglePage</code> , <code>/OneColumn</code> , <code>/TwoColumnLeft</code> , <code>/TwoColumnRight</code> , <code>/TwoPageLeft</code> , <code>/TwoPageRight</code>
PageMode	name	one of <code>/UseNone</code> , <code>/UseOutlines</code> , <code>/UseThumbs</code> , <code>/UseOC</code> , <code>/UseAttachments</code> (PDF 1.6)
Perms	objref or dict	permissions
PieceInfo	objref or dict	
SpiderInfo	objref or dict	
StructTreeRoot	objref or dict	
Threads	objref to an array	
URI	objref or dict	
Version	name	eg. <code>/1.7</code>
$\langle unknown \rangle$		an unknown $\langle name \rangle$ will be inserted without a warning.

Simple entries in subdictionaries of the catalog The following resource pathes have been predeclared and allow to add values to the respective subdictionaries of the catalog. The names of the dictionaries follow the naming and location of the dictionaries in the PDF reference. If $\langle Name \rangle$ gets assigned two values the last one wins.

Example: `\pdfmanagement_add:nnn {Catalog/MarkInfo}{Marked}{true}`

Path/dictionary	Names	Value	Remark
Catalog/AA	WC, WS, DS, WP,DP	all dict	
Catalog/AcroForm	NeedAppearances	boolean	In pdf 2.0 NeedAppearances is deprecated, it is then required that every widget has an appearance streams.
	SigFlags	Integer	
	DA	String	
	Q	Integer	
Catalog/AcroForm/DR	XFA <i><name></i>	stream or array	pdf 1.5 probably unneeded
Catalog/AcroForm/DR/Font	<i><name></i>	dict	
Catalog/MarkInfo	Marked	boolean	
	UserProperties	boolean	
	Suspects	boolean	
Catalog/ViewerPreferences	HideToolbar	boolean	
	Direction	/R2L or /L2R	
	...		many more, see the reference

Catalog entries with multiple values in arrays The following entries are special: Their values are arrays and it must be possible to append to such arrays. This means that a new call to set this value doesn't replace the value but appends it. The value is an object reference. It is sensible to declare the object first. E.g.

```
\pdf_object_new:nn {module/intent}{dict}
\pdf_object_write:nn {module/intent}{...}
\pdfmanagement_add:nxx {Catalog} {OutputIntents}{\pdf_object_ref:n {module/intent}}
```

or

```
\pdf_object_unnamed_write:nn {dict} { ... }
\pdfmanagement_add:nxx {Catalog} {OutputIntents}{\pdf_object_ref_last:}
```

Path/dictionary	Name	Value	Remark
Catalog/AcroForm	Fields	object reference	
Catalog/AcroForm	CO	object reference	
Catalog	AF	object reference	PDF 2.0, associated files
Catalog/OCProperties	OCGs	object reference	if there are OCProperties, OCGs and D are required.
Catalog/OCProperties	Configs	object reference	
Catalog/OCProperties	D	object reference	This is actually a single value as there can be only one default. If the value is set twice, the second wins, and the first is added to OCProperties/Configs.
Catalog	OutputIntents	object reference	
Catalog	Requirements	object reference	PDF 1.7
Catalog/Names	EmbeddedFiles	object reference	This should reference a filespec dictionary. It will attach the file to the file panel.

2 l3pdfmanagement implementation

```

1 <@@=pdfmanagement>
2 <*header>
3 %
4 \ProvidesExplPackage{l3pdfmanagement}{2021-07-21}{0.95g}
5 {Management of core PDF dictionaries (LaTeX PDF management testphase bundle)}
6 </header>

```

2.1 Messages

```

7 <*package>
8 \msg_new:nnn { pdfmanagement } { unknown-dict }
9 { The~PDF~management~resource~'#1'~is~unknown. }
10
11 \msg_new:nnn { pdfmanagement } { empty-value }
12 { The~value~for~#1~is~empty~and~will~be~ignored }
13
14 \msg_new:nnn { pdfmanagement } { no-removal }
15 { It~is~not~possible~to~remove~values~from~'#1'~.}
16
17 \msg_new:nnn { pdfmanagement } { no-show }
18 { It~is~not~possible~to~show~the~content~of~'#1'~.}
19
20 \msg_new:nnn { pdfmanagement } { show-dict }
21 {
22   The~PDF~resource~'#1'~
23   \tl_if_empty:nTF {#2}
24   { is~empty \>~ . }

```

```

25     { contains~the~pairs~(without~outer~braces): #2 . }
26   }
27   \msg_new:nnn { pdfmanagement } { dict-already-defined }
28   {
29     The~path~'#1'~is~already~defined.
30   }
31   \msg_new:nnn { pdfmanagement } { inactive }
32   {
33     The~PDF~resources~management~is~not~active\\
34     command~'#1'~ignored.
35   }

```

`\g__pdfmanagement_active_bool` This boolean will control the activation of the management code. It is used in the hooks, and in some backend files. `\DeclareDocumentMetadata` should set it to true

```

36 \bool_new:N \g__pdfmanagement_active_bool

```

(End definition for `\g__pdfmanagement_active_bool`.)

A user predicate to test if the management code is active

```

37 \prg_new_conditional:Npnn \__pdfmanagement_if_active: { p , T , F , TF }
38 {
39   \bool_if:NTF \g__pdfmanagement_active_bool
40   { \prg_return_true: }
41   { \prg_return_false: }
42 }
43 \prg_set_eq_conditional:NNn
44 \pdfmanagement_if_active: \__pdfmanagement_if_active: { p , T , F , TF }
45

```

We use a hook, to collect value added before the backend is ready.

```

46 \hook_new:n {pdfmanagement/add}
47 \cs_new_protected:Npn \pdfmanagement_add:nnn #1 #2 #3
48 {
49   \__pdfmanagement_if_active:TF
50   {
51     \pdfdict_if_exist:nTF { g__pdf_Core/#1 }
52     {
53       \hook_gput_code:nnn
54       {pdfmanagement/add}
55       {pdfmanagement}
56       {
57         \__pdfmanagement_handler_gput:nnn { #1 } { #2 } { #3 }
58       }
59     }
60     {
61       \msg_error:nnn{pdfmanagement}{unknown-dict}{#1}
62     }
63   }
64   {
65     \msg_warning:nnx {pdfmanagement}{inactive}
66     {\tl_to_str:n {\pdfmanagement_add:nnn}}
67   }
68 }
69
70 \cs_generate_variant:Nn \pdfmanagement_add:nnn {nnx,nxx}

```

2.2 Hooks – shipout and end of run code

Code is executed in three places: At shipout of every page, at shipout of the last page, at the end of the document (after the last clearpage). Due to backend differences the code in the three places (and the exact timing) can be different: pdf_latex/lualatex can execute code after the last `\clearpage` which the dvi-based drivers have to add on a shipout page.

`\g__kernel_pdfmanagement_end_run_code_tl`

This variables contain the code run in the three places.

```

71 \tl_new:N \g__kernel_pdfmanagement_thispage_shipout_code_tl
72 \tl_new:N \g__kernel_pdfmanagement_lastpage_shipout_code_tl
73 \tl_new:N \g__kernel_pdfmanagement_end_run_code_tl

(End definition for \g__kernel_pdfmanagement_thispage_shipout_code_tl \g__kernel_pdfmanagement_
lastpage_shipout_code_tl \g__kernel_pdfmanagement_end_run_code_tl.)

74 \tl_gset:Nn \g__kernel_pdfmanagement_thispage_shipout_code_tl
75 {
76   \bool_if:NT \g__pdfmanagement_active_bool
77   {
78     \exp_args:NV \__pdf_backend_ThisPage_gpush:n { \g_shipout_readonly_int }
79     \exp_args:NV \__pdf_backend_PageResources_gpush:n { \g_shipout_readonly_int }
80   }
81 }
82
83 \tl_gset:Nn \g__kernel_pdfmanagement_end_run_code_tl
84 {
85   \bool_if:NT \g__pdfmanagement_active_bool
86   {
87     \__pdf_backend_PageResources_obj_gpush: %ExtGState etc
88     \__pdfmanagement_Pages_gpush: %pagesattr
89     \__pdfmanagement_Info_gpush: %pdfinfo
90     \__pdfmanagement_Catalog_gpush:
91   }
92 }
```

2.3 Naming convention

Currently the following names are used: All have internally additionally a `Core` before the slash, to hide the real name a bit.

<code>/Info</code>	%	<code>(\pdfinfo)</code>
<code>/Catalog</code>	%	<code>(\pdfcatalog)</code>
<code>/Catalog/AA</code>	%	
<code>/Catalog/AcroForm</code>		
<code>/Catalog/OCProperties</code>		
<code>/Catalog/OutputIntents</code>		
<code>/Catalog/AcroForm/DR</code>		
<code>/Catalog/AcroForm/DR/Font</code>		
<code>/Catalog/MarkInfo</code>		
<code>/Catalog/ViewerPreferences</code>		
<code>/Pages</code>	%	<code>(\pagesattr)</code>
<code>/Page</code>	%	<code>(\pageattr)</code>
<code>/ThisPage</code>	%	<code>(\pageattr)</code>

```

/backend_PageN/Resources/Properties % this is only internal.
/Page/Resources/ExtGState
/Page/Resources/ColorSpace
/Page/Resources/Pattern
/Page/Resources/Shading
/Page/Resources/Properties
/Xform/Resources/Properties

```

```

    \_pdfmanagement_handler_gput:nnn \_pdfmanagement_handler_gput:nnn is the main command to fill the dictionaries. In
    \_pdfmanagement_get:nnN simple cases it directly fill the property list, but if a handler exists this is called. It is
    \_pdfmanagement_gremove:nn important to use it only in places where this make sense.
    \_pdfmanagement_show:n

93
94 %global
95 \cs_new_protected:Npn \_pdfmanagement_handler_gput:nnn #1 #2 #3 % #1 dict, #2 name, #3 value
96 {
97     \tl_if_empty:nTF { #3 }
98     {
99         \msg_none:nnn { pdfmanagement } { empty-value } { /#1/#2 }
100     }
101     {
102         \pdfdict_if_exist:nTF { g__pdf_Core/#1 }
103         {
104             \cs_if_exist:cTF
105             { __pdfmanagement_handler/#1/?_gput:nn } %general, name independant handler
106             { \use:c {__pdfmanagement_handler/#1/?_gput:nn} {#2} {#3} }
107             {
108                 \cs_if_exist:cTF
109                 { __pdfmanagement_handler/#1/#2_gput:n }
110                 { \use:c {__pdfmanagement_handler/#1/#2_gput:n} {#3} } %special handler
111                 {
112                     \exp_args:Nnx
113                     \prop_gput:cnn
114                     { \_kernel_pdfdict_name:n { g__pdf_Core/#1 } }
115                     { \str_convert_pdfname:n { #2 } }
116                     { #3 }
117                 }
118             }
119         }
120         {
121             \msg_error:nnn { pdfmanagement } { unknown-dict } { #1 }
122         }
123     }
124 }
125
126
127 \cs_generate_variant:Nn \_pdfmanagement_handler_gput:nnn {nxx}
128
129 \cs_new_protected:Npn \_pdfmanagement_get:nnN #1 #2 #3 %path,key,macro
130 {
131     \exp_args:Nnx
132     \prop_get:cnN
133     { \_kernel_pdfdict_name:n { g__pdf_Core/#1 } }
134     { \str_convert_pdfname:n {#2} } #3

```

```

135     }
136
137
138 \cs_new_protected:Npn \__pdfmanagement_handler_gremove:nn #1 #2 %path,key
139 {
140     \pdfdict_if_exist:nTF { g__pdf_Core/#1 }
141     {
142         \cs_if_exist:cTF
143         { __pdfmanagement_handler/#1/?_gremove:n } %general, name independant handler
144         { \use:c {__pdfmanagement_handler/#1/?_gremove:n} {#2} }
145         {
146             \cs_if_exist:cTF
147             { __pdfmanagement_handler/#1/#2_gremove: }
148             { \use:c {__pdfmanagement_handler/#1/#2_gremove:} } %special handler
149             {
150                 \exp_args:Nnx
151                 \prop_gremove:cn
152                 { \__kernel_pdfdict_name:n { g__pdf_Core/#1 } }
153                 { \str_convert_pdfname:n {#2} }
154             }
155         }
156     }
157     {
158         \msg_error:nnn { pdfmanagement } { unknown-dict } { #1 }
159     }
160 }
161
162 \cs_new_protected:Npn \__pdfmanagement_gremove:nn #1 #2 %path,key
163 {
164     \pdfdict_if_exist:nTF { g__pdf_Core/#1 }
165     {
166         \exp_args:Nnx
167         \prop_gremove:cn
168         { \__kernel_pdfdict_name:n { g__pdf_Core/#1 } }
169         { \str_convert_pdfname:n{#2} }
170     }
171     {
172         \msg_error:nnn { pdfmanagement } { unknown-dict } { #1 }
173     }
174 }
175
176
177 \cs_new_protected:Npn \__pdfmanagement_show:Nn #1#2
178 {
179     \cs_if_exist:cTF
180     { __pdfmanagement_handler/#2/?_show: } %general, name independant handler
181     { \use:c {__pdfmanagement_handler/#2/?_show:} }
182     {
183         \prop_if_exist:cTF { \__kernel_pdfdict_name:n { g__pdf_Core/#2 } }
184         {
185             #1
186             { pdfmanagement } { show-dict }
187             { \tl_to_str:n {#2} }
188             {

```

```

189         \prop_map_function:cN
190         {\__kernel_pdfdict_name:n { g__pdf_Core/#2 }}
191         \msg_show_item:nn
192     }
193     { } { }
194 }
195 {
196     #1 { pdfmanagement } { unknown-dict } {#2}{ }{ }{ }
197 }
198 }
199 }
200
201 \cs_new_protected:Npn \__pdfmanagement_show:n #1 %path
202 {
203     \prop_show:c { \__kernel_pdfdict_name:n { g__pdf_Core/#1 } }
204 }

```

(End definition for __pdfmanagement_handler_gput:nnn and others.)

```

205 \cs_new_protected:Npn \pdfmanagement_show:n #1
206 {
207     \__pdfmanagement_show:Nn \msg_show:nnxxxx {#1}
208 }
209
210 \cs_new_protected:Npn \pdfmanagement_remove:nn #1 #2
211 {
212     \pdfdict_if_exist:nTF { g__pdf_Core/#1 }
213     {
214         \__pdfmanagement_handler_gremove:nn { #1 }{ #2 }
215     }
216     {
217         \msg_error:nnn{pdfmanagement}{unknown-dict}{#1}
218     }
219 }
220
221 \cs_new_protected:Npn \pdfmanagement_get:nnN #1 #2 #3
222 {
223     \pdfdict_if_exist:nTF { g__pdf_Core/#1 }
224     {
225         \__pdfmanagement_get:nnN { #1 }{ #2 } #3
226     }
227     {
228         \msg_error:nnn{pdfmanagement}{unknown-dict}{#1}
229     }
230 }

```

2.4 The Info dictionary

Initialization of the dictionary:

```

229 \pdfdict_new:n { g__pdf_Core/Info}

```

`__pdfmanagement_Info_gpush:` `__pdfmanagement_Info_gpush:` is the command that outputs the info dictionary (currently in the end-of-run hooks).

```

230 % push to the register command / issue the special
231 \cs_new_protected:Npn \__pdfmanagement_Info_gpush:

```

```

232 {
233   \prop_map_function:cN
234   { \__kernel_pdffdict_name:n { g__pdf_Core/Info} }
235   \__pdf_backend_info_gput:nn
236   \prop_gclear:c { \__kernel_pdffdict_name:n { g__pdf_Core/Info} }
237 }

```

(End definition for __pdfmanagement_Info_gpush:.)

2.5 The Pages dictionary code

At first the initialisation

```

238 \pdffdict_new:n { g__pdf_Core/Pages}

```

__pdfmanagement_Pages_gpush: This is the command that outputs the Pages dictionary. It is used at the end of the document in \g__pdf_backend_end_run_tl

```

239 % push to the register command / issue the special
240 \cs_new_protected:Npn \__pdfmanagement_Pages_gpush:
241 {
242   \pdffdict_if_empty:nF { g__pdf_Core/Pages}
243   {
244     \exp_args:Nx \__pdf_backend_Pages_primitive:n
245     {
246       \pdffdict_use:n { g__pdf_Core/Pages}
247     }
248   }
249 }
250

```

(End definition for __pdfmanagement_Pages_gpush:.)

2.6 The Page and ThisPage dictionary

At first the initialisation.

```

251 \pdffdict_new:n { g__pdf_Core/Page }
252 \pdffdict_new:n { g__pdf_Core/ThisPage }
253
254 %handler for pdfmanagement
255 \cs_new_protected:cpn { __pdfmanagement_handler/Page/?_gput:nn } #1 #2
256 {
257   \__pdf_backend_Page_gput:nn { #1 } { #2 }
258 }
259 % remove:
260 \cs_new_protected:cpn { __pdfmanagement_handler/Page/?_gremove:n } #1
261 {
262   \__pdf_backend_Page_gremove:n { #1 }
263 }
264
265 % handler for pdfmanagement
266 \cs_new_protected:cpn { __pdfmanagement_handler/ThisPage/?_gput:nn } #1 #2
267 {
268   \prop_gput:cnn { \__kernel_pdffdict_name:n { g__pdf_Core/ThisPage } } { #1 } { #2 }
269   \bool_if:NT \g__pdfmanagement_active_bool
270   {

```

```

271         \__pdf_backend_ThisPage_gput:nn { #1 }{ #2 }
272     }
273 }
274
275 \cs_new_protected:cpn { __pdfmanagement_handler/ThisPage/?_gremove:n } #1
276 {
277     \msg_warning:nnn { pdfmanagement } { no-removal }{ThisPage}
278 }
279
280 \cs_new_protected:cpn { __pdfmanagement_handler/ThisPage/?_show: }
281 {
282     \msg_warning:nnn { pdfmanagement } { no-show }{ThisPage}
283 }
284

```

2.6.1 “Page/Resources”: ExtGState, ColorSpace, Shading, Pattern

```

285 \clist_const:Nn \c__pdfmanagement_PageResources_clist
286 {
287     ExtGState,
288     ColorSpace,
289     Pattern,
290     Shading,
291 }
292
293 \clist_map_inline:Nn \c__pdfmanagement_PageResources_clist
294 {
295     \pdfdict_new:n { g__pdf_Core/Page/Resources/#1}
296 }
297 %
298 % setter: #1 is the name of the resource
299 \cs_new_protected:cpn { __pdfmanagement_handler/Page/Resources/ExtGState/?_gput:nn } #1 #2
300 {
301     \__pdf_backend_PageResources_gput:nnn {ExtGState} { #1 }{ #2 }
302 }
303
304 \cs_new_protected:cpn { __pdfmanagement_handler/Page/Resources/ColorSpace/?_gput:nn } #1 #2
305 {
306     \__pdf_backend_PageResources_gput:nnn {ColorSpace} { #1 }{ #2 }
307 }
308
309 \cs_new_protected:cpn { __pdfmanagement_handler/Page/Resources/Shading/?_gput:nn } #1 #2
310 {
311     \__pdf_backend_PageResources_gput:nnn {Shading} { #1 }{ #2 }
312 }
313
314 \cs_new_protected:cpn { __pdfmanagement_handler/Page/Resources/Pattern/?_gput:nn } #1 #2
315 {
316     \__pdf_backend_PageResources_gput:nnn {Pattern} { #1 }{ #2 }
317 }

```

2.6.2 “Catalog”

The catalog has mixed entries: toplevel, subdictionaries, and entries which must build arrays.

\c__pdfmanagement_Catalog_toplevel_clist \c__pdfmanagement_Catalog_sub_clist \c__pdfmanagement_Catalog_seq_clist _pdfmanagement_catalog_XX_gput:n	<p>This variables hold the list of the various types of entries. With it the various <code>_gput</code> commands are generated.</p> <p><i>(End definition for \c__pdfmanagement_Catalog_toplevel_clist, \c__pdfmanagement_Catalog_sub_clist, and \c__pdfmanagement_Catalog_seq_clist.)</i></p> <p>Various commands to handle subentries and special cases.</p> <pre> 318 \pdfdict_new:n { g__pdf_Core/Catalog} 319 320 \clist_const:Nn \c__pdfmanagement_Catalog_toplevel_clist 321 { 322 Collection, 323 DPartRoot, 324 Lang, 325 Legal, 326 Metadata, 327 NeedsRendering, 328 OCPProperties/D, 329 OpenAction, 330 PageLabels, 331 PageLayout, 332 PageMode, 333 Perms, 334 PieceInfo, 335 SpiderInfo, 336 StructTreeRoot, 337 Threads, 338 URI, 339 Version 340 } 341 342 \clist_const:Nn \c__pdfmanagement_Catalog_sub_clist 343 { 344 AA, 345 AcroForm, 346 AcroForm/DR, 347 AcroForm/DR/Font, 348 MarkInfo, 349 ViewerPreferences, 350 OCPProperties 351 } 352 353 \clist_map_inline:Nn \c__pdfmanagement_Catalog_sub_clist 354 { 355 \pdfdict_new:n { g__pdf_Core/Catalog/#1} 356 } 357 358 359 \clist_const:Nn \c__pdfmanagement_Catalog_seq_clist 360 { 361 AF, 362 OCPProperties/OCGs, 363 OCPProperties/Configs, 364 OutputIntents, 365 Requirements, </pre>
---	---

```

366     AcroForm/Fields,
367     AcroForm/CO
368 }
369
370
371
372 \clist_map_inline:Nn \c__pdfmanagement_Catalog_seq_clist
373 {
374     \seq_new:c { g__pdfmanagement_/Catalog/#1_seq } % new name later
375     \cs_new_protected:cpn { __pdfmanagement_handler/Catalog/#1_gput:n } ##1
376     {
377         \seq_gput_right:cn { g__pdfmanagement_/Catalog/#1_seq } { ##1 }
378     }
379 }
380
381 \cs_new_protected:cpn { __pdfmanagement_handler/Catalog/OCProperties/D_gput:n } #1
382 {
383     \seq_gput_left:cn
384     { g__pdfmanagement_/Catalog/OCProperties/Configs_seq }
385     { #1 }
386 }

```

(End definition for __pdfmanagement_catalog_XX_gput:n.)

Building the catalog: Push order

__pdfmanagement_Catalog_gpush:

```

387 \cs_new_protected:Npn \__pdfmanagement_Catalog_gpush:
388 {
389     \use:c { __pdfmanagement_/Catalog/AA_gpush: }
390     \use:c { __pdfmanagement_/Catalog/AcroForm_gpush: }
391     \use:c { __pdfmanagement_/Catalog/AF_gpush: }
392     \use:c { __pdfmanagement_/Catalog/MarkInfo_gpush: }
393     \pdfmeta_standard_verify:nT {Catalog_no_OCProperties}
394     {
395         \use:c { __pdfmanagement_/Catalog/OCProperties_gpush: }
396     }
397     \use:c { __pdfmanagement_/Catalog/OutputIntents_gpush: }
398     \use:c { __pdfmanagement_/Catalog/Requirements_gpush: }
399     \use:c { __pdfmanagement_/Catalog/ViewerPreferences_gpush: }
400     % output the single values:
401     \prop_map_function:cn
402     { \__kernel_pdfdict_name:n { g__pdf_Core/Catalog} }
403     \__pdf_backend_catalog_gput:nn
404     % output names tree:
405     \use:c { __pdfmanagement_/Catalog/Names/EmbeddedFiles_gpush: }
406 }

```

(End definition for __pdfmanagement_Catalog_gpush:.)

Building catalog entries: AA

__pdfmanagement_/Catalog/AA_gpush:

```

407 \cs_new_protected:cpn { __pdfmanagement_/Catalog/AA_gpush: }

```

```

408 {
409   \prop_if_empty:cF
410   { \__kernel_pdffdict_name:n { g__pdf_Core/Catalog/AA } }
411   {
412     \__pdf_backend_object_new:nn { __pdfmanagement/Catalog/AA } { dict }
413     \__pdf_backend_object_write:nx
414       { __pdfmanagement/Catalog/AA }
415       { \pdffdict_use:n { g__pdf_Core/Catalog/AA } }
416     \exp_args:Nnx
417     \__pdf_backend_catalog_gput:nn
418       {AA}
419       {
420         \__pdf_backend_object_ref:n { __pdfmanagement/Catalog/AA }
421       }
422   }
423 }

```

(End definition for __pdfmanagement_/Catalog/AA_gpush:.)

Building catalog entries: AcroForm This is the most complicated case. The entries is build from /Catalog/AcroForm/Fields (array), /Catalog/AcroForm/CO (array), /Catalog/AcroForm/DR/Font (dict), /Catalog/AcroForm/DR (dict), /Catalog/AcroForm

__pdfmanagement_/Catalog/AcroForm_gpush:

```

424 \cs_new_protected:cpn { __pdfmanagement_/Catalog/AcroForm_gpush: }
425 {
426   \seq_if_empty:cF { g__pdfmanagement_/Catalog/AcroForm/Fields_seq }
427   {
428     \__pdf_backend_object_new:nn { __pdfmanagement/Catalog/AcroForm/Fields } { array }
429     \__pdf_backend_object_write:nx
430       { __pdfmanagement/Catalog/AcroForm/Fields }
431       { \seq_use:cn { g__pdfmanagement_/Catalog/AcroForm/Fields_seq } {-} }
432     \exp_args:Nnnx
433     \prop_gput:cnn %we have to use \prop here to avoid the handler ...
434     { \__kernel_pdffdict_name:n { g__pdf_Core/Catalog/AcroForm } }
435     { Fields }
436     { \__pdf_backend_object_ref:n { __pdfmanagement/Catalog/AcroForm/Fields } }
437   }
438   \seq_if_empty:cF { g__pdfmanagement_/Catalog/AcroForm/CO_seq }
439   {
440     \__pdf_backend_object_new:nn { __pdfmanagement/Catalog/AcroForm/CO } { array }
441     \exp_args:Nnx
442     \__pdf_backend_object_write:nn
443       { __pdfmanagement/Catalog/AcroForm/CO }
444       { \seq_use:cn { g__pdfmanagement_/Catalog/AcroForm/CO_seq } {-} }
445     \exp_args:Nnnx
446     \prop_gput:cnn %we have to use \prop here to avoid the handler ...
447     { \__kernel_pdffdict_name:n { g__pdf_Core/Catalog/AcroForm } }
448     { CO }
449     { \__pdf_backend_object_ref:n { __pdfmanagement/Catalog/AcroForm/CO } }
450   }
451   \prop_if_empty:cF { \__kernel_pdffdict_name:n { g__pdf_Core/Catalog/AcroForm/DR/Font}}
452   {
453     \__pdf_backend_object_new:nn { __pdfmanagement/Catalog/AcroForm/DR/Font } {dict}

```

```

454     \exp_args:Nnx
455     \__pdf_backend_object_write:nn
456     { __pdfmanagement/Catalog/AcroForm/DR/Font }
457     { \pdfdict_use:n { g__pdf_Core/Catalog/AcroForm/DR/Font } }
458     \exp_args:Nnnx
459     \prop_gput:cnn %we have to use \prop here to avoid the handler ...
460     { \__kernel_pdfdict_name:n { g__pdf_Core/Catalog/AcroForm/DR } }
461     { Font }
462     { \__pdf_backend_object_ref:n { __pdfmanagement/Catalog/AcroForm/DR/Font } }
463   }
464   \prop_if_empty:cF { \__kernel_pdfdict_name:n { g__pdf_Core/Catalog/AcroForm/DR}}
465   {
466     \__pdf_backend_object_new:nn { __pdfmanagement/Catalog/AcroForm/DR } {dict}
467     \exp_args:Nnx
468     \__pdf_backend_object_write:nn
469     { __pdfmanagement/Catalog/AcroForm/DR }
470     { \pdfdict_use:n { g__pdf_Core/Catalog/AcroForm/DR } }
471     \exp_args:Nnnx
472     \prop_gput:cnn %we have to use \prop here to avoid the handler ...
473     { \__kernel_pdfdict_name:n { g__pdf_Core/Catalog/AcroForm } }
474     { DR }
475     { \__pdf_backend_object_ref:n { __pdfmanagement/Catalog/AcroForm/DR } }
476   }
477   \prop_if_empty:cF { \__kernel_pdfdict_name:n { g__pdf_Core/Catalog/AcroForm} }
478   {
479     \__pdf_backend_object_new:nn { __pdfmanagement/Catalog/AcroForm } {dict}
480     \exp_args:Nnx
481     \__pdf_backend_object_write:nn
482     { __pdfmanagement/Catalog/AcroForm }
483     { \pdfdict_use:n { g__pdf_Core/Catalog/AcroForm } }
484     \exp_args:Nnnx
485     \__pdfmanagement_handler_gput:nnn
486     { Catalog }
487     { AcroForm }
488     { \__pdf_backend_object_ref:n { __pdfmanagement/Catalog/AcroForm } }
489   }
490 }
491

```

(End definition for __pdfmanagement_/Catalog/AcroForm_gpush:.)

Building catalog entries: AF AF is an array.

__pdfmanagement_/Catalog/AF_gpush:

```

492 \cs_new_protected:cpn { __pdfmanagement_/Catalog/AF_gpush: }
493 {
494   \seq_if_empty:cF
495   { g__pdfmanagement_/Catalog/AF_seq }
496   {
497     \__pdf_backend_object_new:nn { __pdfmanagement/Catalog/AF } { array }
498     \exp_args:Nnx
499     \__pdf_backend_object_write:nn
500     { __pdfmanagement/Catalog/AF }
501     { \seq_use:cn { g__pdfmanagement_/Catalog/AF_seq } {~} }

```

```

502     \exp_args:Nnx
503     \__pdf_backend_catalog_gput:nn
504     {AF}
505     {
506       \__pdf_backend_object_ref:n {__pdfmanagement/Catalog/AF}
507     }
508   }
509 }

```

(End definition for __pdfmanagement_/Catalog/AF_gpush:.)

Building catalog entries: MarkInfo

__pdfmanagement_/Catalog/MarkInfo_gpush:

```

510 \cs_new_protected:cpn { __pdfmanagement_/Catalog/MarkInfo_gpush: }
511 {
512   \prop_if_empty:cF
513   { \__kernel_pdffdict_name:n { g__pdf_Core/Catalog/MarkInfo } }
514   {
515     \__pdf_backend_object_new:nn { __pdfmanagement_/Catalog/MarkInfo } { dict }
516     \exp_args:Nnx
517     \__pdf_backend_object_write:nn
518     { __pdfmanagement_/Catalog/MarkInfo }
519     { \pdffdict_use:n { g__pdf_Core/Catalog/MarkInfo } }
520     \exp_args:Nnx
521     \__pdf_backend_catalog_gput:nn
522     {MarkInfo}
523     {
524       \__pdf_backend_object_ref:n {__pdfmanagement_/Catalog/MarkInfo}
525     }
526   }
527 }

```

(End definition for __pdfmanagement_/Catalog/MarkInfo_gpush:.)

Building catalog entries: OCProperties This is a dictionary with three entries:

/OCGs (required) An array of indirect references, access needed for more than one package.

/D (required) a dict (given as an object name) to the default configuration

/Configs (optional) an array of indirect references to more configurations.

The /D entry is also a config, it is the first of the seq. The overall structure is nested: a dict with arrays.

__pdfmanagement_/Catalog/OCProperties_gpush:

```

528 % Catalog/OCProperties: OCGs + D is required
529 \cs_new_protected:cpn { __pdfmanagement_/Catalog/OCProperties_gpush: }
530 {
531   \int_compare:nNnT
532   {
533     ( \seq_count:c { g__pdfmanagement_/Catalog/OCProperties/OCGs_seq } ) *
534     ( \seq_count:c { g__pdfmanagement_/Catalog/OCProperties/Configs_seq } )

```

```

535     }
536     >
537     { 0 }
538     {
539         \__pdf_backend_object_new:nn { __pdfmanagement/Catalog/OCProperties } { dict }
540         \seq_gpop_left:cn { g__pdfmanagement_/Catalog/OCProperties/Configs_seq } \l_tmpa_tl
541         \exp_args:Nnx
542         \__pdf_backend_object_write:nn {__pdfmanagement/Catalog/OCProperties}
543         {
544             /OCGs~ [ \seq_use:cn { g__pdfmanagement_/Catalog/OCProperties/OCGs_seq } {~} ]
545             /D~\l_tmpa_tl~
546             \seq_if_empty:cF { g__pdfmanagement_/Catalog/OCProperties/Configs_seq }
547             {
548                 /Configs~
549                 [ \seq_use:cn { g__pdfmanagement_/Catalog/OCProperties/Configs_seq } {~} ]
550             }
551         }
552         \exp_args:Nnx
553         \__pdf_backend_catalog_gput:nn
554         { OCProperties }
555         { \__pdf_backend_object_ref:n {__pdfmanagement/Catalog/OCProperties} }
556     }
557 }

```

(End definition for __pdfmanagement_/Catalog/OCProperties_gpush:.)

Building catalog entries: OutputIntents OutputIntents is an array.

__pdfmanagement_/Catalog/OutputIntents_gpush:

```

558 \cs_new_protected:cpn { __pdfmanagement_/Catalog/OutputIntents_gpush: }
559 {
560     \seq_if_empty:cF
561     { g__pdfmanagement_/Catalog/OutputIntents_seq }
562     {
563         \__pdf_backend_object_new:nn { __pdfmanagement/Catalog/OutputIntents } { array }
564         \exp_args:Nnx
565         \__pdf_backend_object_write:nn
566         { __pdfmanagement/Catalog/OutputIntents }
567         { \seq_use:cn { g__pdfmanagement_/Catalog/OutputIntents_seq } {~} }
568         \exp_args:Nnx
569         \__pdf_backend_catalog_gput:nn
570         {OutputIntents}
571         {
572             \__pdf_backend_object_ref:n {__pdfmanagement/Catalog/OutputIntents}
573         }
574     }
575 }

```

(End definition for __pdfmanagement_/Catalog/OutputIntents_gpush:.)

Building catalog entries: Requirements Requirements is an array.

__pdfmanagement_/Catalog/Requirements_gpush:

```

576 \cs_new_protected:cpn { __pdfmanagement_/Catalog/Requirements_gpush: }

```

```

577 {
578   \seq_if_empty:cF
579   { g__pdfmanagement_/Catalog/Requirements_seq }
580   {
581     \__pdf_backend_object_new:nn { __pdfmanagement/Catalog/Requirements } { array }
582     \exp_args:Nnx
583     \__pdf_backend_object_write:nn
584       { __pdfmanagement/Catalog/Requirements }
585       { \seq_use:cn { g__pdfmanagement_/Catalog/Requirements_seq } {~} }
586     \exp_args:Nnx
587     \__pdf_backend_catalog_gput:nn
588       {Requirements}
589     {
590       \__pdf_backend_object_ref:n { __pdfmanagement/Catalog/Requirements }
591     }
592   }
593 }

```

(End definition for __pdfmanagement_/Catalog/Requirements_gpush:.)

Building catalog entries: ViewerPreferences

__pdfmanagement_/Catalog/ViewerPreferences_gpush:

```

594 \cs_new_protected:cpn { __pdfmanagement_/Catalog/ViewerPreferences_gpush: }
595 {
596   \prop_if_empty:cF
597   { \__kernel_pdffdict_name:n { g__pdf_Core/Catalog/ViewerPreferences } }
598   {
599     \__pdf_backend_object_new:nn { __pdfmanagement/Catalog/ViewerPreferences } { dict }
600     \exp_args:Nnx
601     \__pdf_backend_object_write:nn
602       { __pdfmanagement/Catalog/ViewerPreferences }
603       { \pdffdict_use:n { g__pdf_Core/Catalog/ViewerPreferences } }
604     \exp_args:Nnx
605     \__pdf_backend_catalog_gput:nn
606       {ViewerPreferences}
607     {
608       \__pdf_backend_object_ref:n {__pdfmanagement/Catalog/ViewerPreferences}
609     }
610   }
611 }

```

(End definition for __pdfmanagement_/Catalog/ViewerPreferences_gpush:.)

Building catalog entries: Names/EmbeddedFiles

Handler EmbeddedFiles is an array and needs a special handler to add values.

```

612 \pdffdict_new:n { g__pdf_Core/Catalog/Names }
613
614 \cs_new_protected:cpn { __pdfmanagement_handler/Catalog/Names/EmbeddedFiles_gput:n } #1
615 {
616   \__pdf_backend_NamesEmbeddedFiles_add:n { #1 }
617 }

```

(End definition for Handler. This function is documented on page ??.)

The entry should only be added if there are actually embedded files. This can be tested by checking the names_seq

agement_/Catalog/Names/EmbeddedFiles_gpush:

```

618 %
619 \cs_new_protected:cpn { __pdfmanagement_/Catalog/Names/EmbeddedFiles_gpush: }
620 {
621   \seq_if_empty:NF \g__pdf_backend_EmbeddedFiles_seq
622   {
623     \exp_args:Nx \__pdf_backend_NamesEmbeddedFiles_gpush:n
624     {
625       \seq_use:Nn \g__pdf_backend_EmbeddedFiles_seq {~}
626     }
627   }
628 }

```

(End definition for __pdfmanagement_/Catalog/Names/EmbeddedFiles_gpush:.)

__pdfmanagement_handler/Catalog/?_show:

A handler to show the catalog.

```

629 \cs_new_protected:cpn { __pdfmanagement_handler/Catalog/?_show: }
630 {
631   \iow_term:x
632   {
633     \iow_newline:
634     The~Catalog~contains~in~the~top~level~the~single~value~entries
635     \prop_map_function:cN
636     { \__kernel_pdffdict_name:n { g__pdf_Core/Catalog } }
637     \msg_show_item:nn
638   }
639   \clist_map_inline:Nn \c__pdfmanagement_Catalog_seq_clist
640   {
641     \seq_if_empty:cF { g__pdfmanagement_/Catalog/##1_seq }
642     {
643       \iow_term:x
644       {
645         The~'##1'~array~contains~the~entries
646         \seq_map_function:cN { g__pdfmanagement_/Catalog/##1_seq } \msg_show_item:n
647       }
648     }
649   }
650   \clist_map_inline:Nn \c__pdfmanagement_Catalog_sub_clist
651   {
652     \prop_if_empty:cF { \__kernel_pdffdict_name:n { g__pdf_Core/Catalog/##1 } }
653     {
654       \iow_term:x
655       {
656         The~Catalog~subdirectory~'##1'~contains~the~single~value~entries
657         \prop_map_function:cN
658         { \__kernel_pdffdict_name:n { g__pdf_Core/Catalog/##1 } }
659         \msg_show_item:nn
660       }
661     }
662   }
}

```

```

663   \tl_show:x {\tl_to_str:n{\pdfmanagement_show:n{Catalog}}}  

664   }

```

(End definition for `__pdfmanagement_handler/Catalog/?_show:.`)

2.7 xform / Properties

```

665 \pdfdict_new:n { g__pdf_Core/Xform/Resources/Properties}  

666 \</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.

Symbols		H	
<code>\\</code>	24, 33	Handler	<u>612</u>
B		hook commands:	
bool commands:		<code>\hook_gput_code:nnn</code>	53
<code>\bool_if:NTF</code>	39, 76, 85, 269	<code>\hook_new:n</code>	46
<code>\bool_new:N</code>	36	I	
C		int commands:	
<code>\clearpage</code>	<u>10</u>	<code>\int_compare:nNnTF</code>	531
clist commands:		iow commands:	
<code>\clist_const:Nn</code> ...	285, 320, 342, 359	<code>\iow_newline:</code>	633
<code>\clist_map_inline:Nn</code>		<code>\iow_term:n</code>	631, 643, 654
.....	293, 353, 372, 639, 650	K	
cs commands:		kernel internal commands:	
<code>\cs_generate_variant:Nn</code>	70, 127	<code>__kernel_pdffdict_name:n</code> 114, 133,	
<code>\cs_if_exist:NTF</code> 104, 108, 142, 146, 179		152, 168, 183, 190, 203, 234, 236,	
<code>\cs_new_protected:Npn</code>		268, 402, 410, 434, 447, 451, 460,	
.....	47, 95, 129, 138, 162, 177,	464, 473, 477, 513, 597, 636, 652, 658	
201, 205, 209, 219, 231, 240, 255,		<code>\g__kernel_pdfmanagement_end_-</code>	
260, 266, 275, 280, 299, 304, 309,		<code>run_code_tl</code>	73, 83
314, 375, 381, 387, 407, 424, 492,		<code>\g__kernel_pdfmanagement_-</code>	
510, 529, 558, 576, 594, 614, 619, 629		<code>lastpage_shipout_code_tl</code>	72
D		<code>\g__kernel_pdfmanagement_-</code>	
<code>\DeclareDocumentMetadata</code>	<u>9</u>	<code>thispage_shipout_code_tl</code> ..	71, 74
E		<code>\g__kernel_pdfmanagement_-</code>	
exp commands:		<code>thispage_shipout_code_-</code>	
<code>\exp_args:Nnnx</code> 432, 445, 458, 471, 484		<code>tl\g__kernel_pdfmanagement_-</code>	
<code>\exp_args:Nnx</code>		<code>lastpage_shipout_code_-</code>	
.....	112, 131, 150, 166, 416, 441,	<code>tl\g__kernel_pdfmanagement_-</code>	
454, 467, 480, 498, 502, 516, 520,		<code>end_run_code_tl</code>	<u>71</u>
541, 552, 564, 568, 582, 586, 600, 604		M	
<code>\exp_args:NV</code>	78, 79	msg commands:	
<code>\exp_args:Nx</code>	244, 623	<code>\msg_error:nnn</code>	
		61, 121, 158, 172, 216, 226
		<code>\msg_new:nnn</code> ..	8, 11, 14, 17, 20, 27, 31

\msg_none:nnn	99	pdfmanagement:Page/Resources/ColorSpace	5
\msg_show:nnnnnn	207	pdfmanagement:Page/Resources/ExtGState	5
\msg_show_item:n	646	pdfmanagement:Page/Resources/Pattern	5
\msg_show_item:nn	191, 637, 659	pdfmanagement:Page/Resources/Shading	5
\msg_warning:nnn	65, 277, 282	pdfmanagement:Pages	4
N		pdfmanagement:ThisPage	5
\newpage	4	\pdfmanagement_add:nnn	2-5, 47, 66, 70
P		\pdfmanagement_get:nnN	219
pdf internal commands:		\pdfmanagement_if_active:	44
__pdf_backend_catalog_gput:nn	403, 417, 503, 521, 553, 569, 587, 605	\pdfmanagement_if_active:TF	2
\g__pdf_backend_EmbeddedFiles_-seq	621, 625	\pdfmanagement_if_active_p:	2
\g__pdf_backend_end_run_tl	14	\pdfmanagement_remove:nn	3, 4, 209
__pdf_backend_info_gput:nn	235	\pdfmanagement_show:n	3, 5, 205, 663
__pdf_backend_NamesEmbeddedFiles_-add:n	616	pdfmanagement internal commands:	
__pdf_backend_NamesEmbeddedFiles_-gpush:n	623	__pdfmanagement_/Catalog/AA_-gpush:	407
__pdf_backend_object_new:nn	412, 428, 440, 453, 466, 479, 497, 515, 539, 563, 581, 599	__pdfmanagement_/Catalog/AcroForm_-gpush:	424
__pdf_backend_object_ref:n	420, 436, 449, 462, 475, 488, 506, 524, 555, 572, 590, 608	__pdfmanagement_/Catalog/AF_-gpush:	492
__pdf_backend_object_write:nn	413, 429, 442, 455, 468, 481, 499, 517, 542, 565, 583, 601	__pdfmanagement_/Catalog/MarkInfo_-gpush:	510
__pdf_backend_Page_gput:nn	257	__pdfmanagement_/Catalog/Names/EmbeddedFiles_-gpush:	618
__pdf_backend_Page_gremove:n	262	__pdfmanagement_/Catalog/OCProperties_-gpush:	528
__pdf_backend_PageResources_-gpush:n	79	__pdfmanagement_/Catalog/OutputIntents_-gpush:	558
__pdf_backend_PageResources_-gput:nnn	301, 306, 311, 316	__pdfmanagement_/Catalog/Requirements_-gpush:	576
__pdf_backend_PageResources_-obj_gpush:	87	__pdfmanagement_/Catalog/ViewerPreferences_-gpush:	594
__pdf_backend_Pages_primitive:n	244	\g__pdfmanagement_active_bool	36, 39, 76, 85, 269
__pdf_backend_ThisPage_gpush:n	78	__pdfmanagement_Catalog_gpush:	90, 387, 387
__pdf_backend_ThisPage_gput:nn	271	\c__pdfmanagement_Catalog_seq_-clist	318, 359, 372, 639
\pdfcatalog	1, 2	\c__pdfmanagement_Catalog_sub_-clist	318, 342, 353, 650
pdfdict commands:		\c__pdfmanagement_Catalog_-toplevel_clist	318, 320
\pdfdict_if_empty:nTF	242	__pdfmanagement_catalog_XX_-gput:n	318
\pdfdict_if_exist:nTF	51, 102, 140, 164, 211, 221	__pdfmanagement_get:nnN	93, 129, 223
\pdfdict_new:n	229, 238, 251, 252, 295, 318, 355, 612, 665	__pdfmanagement_gremove:nn	93, 162
\pdfdict_use:n	246, 415, 457, 470, 483, 519, 603	__pdfmanagement_handler/Catalog/?_-show:	629
\pdfinfo	1, 2	__pdfmanagement_handler_-gput:nnn	11, 57, 93, 95, 127, 485
pdfmanagement commands:			
pdfmanagement:Info	3		
pdfmanagement:Page	4		

_pdfmanagement_handler_- gremove:nn	138, 213	\prop_show:N	203
_pdfmanagement_if_active:	37, 44	\ProvidesExplPackage	4
_pdfmanagement_if_active:TF	49		
_pdfmanagement_Info_gpush:		S	
.....	13, 89, 230, 231	seq commands:	
\c_pdfmanagement_PageResources_- clist	285, 293	\seq_count:N	533, 534
_pdfmanagement_Pages_gpush:		\seq_gpop_left:NN	540
.....	88, 239, 240	\seq_gput_left:Nn	383
_pdfmanagement_show:n	93, 201	\seq_gput_right:Nn	377
_pdfmanagement_show:Nn	177, 207	\seq_if_empty:NTF	
pdfmeta commands:		426, 438, 494, 546, 560, 578, 621, 641	
\pdfmeta_standard_verify:nTF	393	\seq_map_function:NN	646
\pdfpageattr	1, 2, 4	\seq_new:N	374
\pdfpageresources	1, 2, 5	\seq_use:Nn	
\pdfpagesattr	1, 2, 4	431, 444, 501, 544, 549, 567, 585, 625	
prg commands:		shipout commands:	
\prg_new_conditional:Npnn	37	\g_shipout_readonly_int	78, 79
\prg_return_false:	41	\special	1
\prg_return_true:	40	str commands:	
\prg_set_eq_conditional:NNn	43	\str_convert_pdfname:n	
\prop	433, 446, 459, 472	2, 3, 115, 134, 153, 169
prop commands:			
\prop_gclear:N	236	T	
\prop_get:NnN	132	tl commands:	
\prop_gput:Nnn		\tl_gset:Nn	74, 83
.....	113, 268, 433, 446, 459, 472	\tl_if_empty:nTF	23, 97
\prop_gremove:Nn	151, 167	\tl_new:N	71, 72, 73
\prop_if_empty:NTF		\tl_show:n	663
....	409, 451, 464, 477, 512, 596, 652	\tl_to_str:n	66, 187, 663
\prop_if_exist:NTF	183	\l_tmpa_tl	540, 545
\prop_map_function:NN			
.....	189, 233, 401, 635, 657	U	
		use commands:	
		\use:N	106, 110, 144, 148, 181, 389,
			390, 391, 392, 395, 397, 398, 399, 405